ESCUELA SUPERIOR POLITECNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

“Diseño e Implementación de una Tarjeta de Monitoreo y Control en Forma Remota a través de la Internet, utilizando la Tecnología del Microcontrolador PIC18F97J60”

TESIS DE GRADO

Previo a la obtención del Título de:

INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

Presentada por:
Efrén Arturo Montenegro Viera.
Eduardo Patricio Sandoya Tinoco.

GUAYAQUIL – ECUADOR
2008

AGRADECIMIENTO

Agradecemos a Dios por ayudarnos a concluir este trabajo a pesar de todas las adversidades, a nuestros padres y a todas las personas que de manera directa o indirecta contribuyeron a la realización de este tema de tesis; de manera especial queremos agradecer a Ing. Ronald Ponguillo, Director de Tesis por su ayuda, paciencia y colaboración a lo largo del proyecto.

DEDICATORIA

A Dios antes que todo, seguido de mi Padre Efrén, mi Madre Adriana, mis hermanos, a mi familia en general que estuvo involucrada en este proceso; por empujarme en los momentos que me quedaba y alentarme a seguir hasta el final.

Efrén Montenegro Viera.

DEDICATORIA

A Dios por ser mi mayor motivación para seguir adelante; a mi hermano Iván, mi Madre Marina y mi Padre Eduardo que con paciencia supieron guiarme y estuvieron siempre para apoyarme incondicionalmente a lo largo de mi carrera y en mi vida personal.

Patricio Sandoya Tinoco.

TRIBUNAL DE GRADUACION

Ing. Jorge Aragundi R.
SUB DECANO DE LA FIEC
PRESIDENTE

Ing. Ronald Ponguillo I.
DIRECTOR DE TESIS

Ing. Carlos Valdivieso A.
VOCAL PRINCIPAL

Ing. Pedro Vargas G.
VOCAL PRINCIPAL

DECLARACIÓN EXPRESA

“La responsabilidad del contenido de esta Tesis de Grado, me corresponde exclusivamente, y el patrimonio intelectual de la misma a la ESCUELA SUPERIOR POLITECNICA DEL LITORAL”.

(Reglamento de Graduación de la ESPOL).

Efrén Arturo Montenegro Viera.

Eduardo Patricio Sandoya Tinoco.

RESUMEN

El trabajo presentado en este proyecto de tesis, fue desarrollado para demostrar la aplicación de tecnologías, como el Microcontrolador y la Domótica, y consiste en el “Diseño e Implementación de una Tarjeta de Monitoreo y Control en Forma Remota a través de la Internet, utilizando la Tecnología del Microcontrolador PIC18F97J60”.

El primer capítulo, comprende una explicación teórica de la Domótica, empezando por la introducción y definición de términos, los dispositivos que están involucrados en este sistema de control con su respectiva arquitectura, así también se incluyen los diferentes medios de transmisión, las tecnologías y protocolos utilizados.

En el segundo capítulo se describe los fundamentos teóricos de la tecnología Ethernet, comenzando por su historia, estudiando las capas que presentan los modelos de referencia OSI y TCP/IP respectivamente; así como, los diferentes protocolos utilizados en el modelo TCP/IP.

En el tercer capítulo constan los principios teóricos de la familia PIC18, en especial del Microcontrolador PIC18F97J60; así como, su descripción y utilización del mismo. También se detalla acerca del compilador MPLAB, sus

herramientas C18 y C30. Además se explica sobre la configuración y utilización del programador ICD2, así como del Bus Serial Universal (USB).

El cuarto capítulo contiene los detalles del diseño de los módulos de la tesis como la descripción técnica, diagramas de bloques y flujo, la programación, diseño de la página web y sus herramientas, elementos utilizados en el diseño y desarrollo, tanto de la tarjeta como del adaptador.

El quinto capítulo se analiza el funcionamiento del sistema, con los resultados obtenidos de las pruebas realizadas durante la transmisión y recepción de datos.

El último capítulo se presenta un análisis de costos de los elementos utilizados en las tarjetas de control y del adaptador respectivamente, así como el diseño y montaje de las mismas.

Finalmente se anexa la descripción de los componentes electrónicos utilizados para el desarrollo del proyecto, en las recomendaciones y conclusiones damos a conocer las sugerencias a las que llegamos, a partir de las pruebas y resultados obtenidos en el proceso.

ÍNDICE GENERAL

Anexos

ABREVIATURAS

A	Amperio
bit	Unidad de medida de información equivalente a la elección entre dos posibilidades igualmente probables
bps	bits por segundo
DC	Corriente directa
dsPIC	Microcontrolador de Microchip con soporte para procesamiento de señales
ICSP	Programación serial en circuito
Kbytes	Kilobytes
Kbps	Kilobits por segundo
KB/s	Kilobytes por segundo
kHz	Kilohertz
kOhm	Kiloohmio
m	metro
MAC	Control de Acceso al Medio
MHz	Mega hertz
Mbytes	Megabytes
ohm	Ohmio
PIC	Microcontrolador de Microchip
RAM	Memoria de Acceso Aleatorio
TTL	Lógica transistor transistor
USB	Bus serial universal
UTP	Cable trenzado
V	Voltio
VI	Instrumento virtual
Vac	Voltio de corriente alterna
Vpp	Voltio pico a pico
Watt	Vatio
WDT	Temporizador de vigía extendido

ÍNDICE DE IMÁGENES

ÍNDICE DE TABLA

INTRODUCCIÓN

En la actualidad, la automatización de viviendas está en auge y son un paso hacia el futuro. Existen diferentes maneras de implementar la domótica, motivo por el cual, hemos decidido optar por una nueva tecnología como lo es la de los PICs con sistemas embebidos para el desarrollo de una aplicación que reduzca en costo y tamaño lo que hoy por hoy podemos encontrar en los mercados.

Con la ayuda de Microchip y sus diferentes elementos, desarrollamos un software que nos permitirá el monitoreo y control de las luces de una vivienda, recalcando que el mismo con las respectivas modificaciones podrá ser utilizado para diferentes aplicaciones eléctricas.

Nuestra aplicación consta de una tarjeta de control, en la que estará nuestro principal componente que es el PIC18F97J60 con módulo Ethernet embebido programado con el software MPLAB IDE utilizando el programador serial ICD2, ambos proveídos por Microchip.

A más de nuestra tarjeta de control, contaremos con una tarjeta de potencia conectada en una etapa posterior, con la finalidad de manejar los niveles de voltaje requeridos para el buen funcionamiento de nuestro proyecto.

Todo el análisis y funcionamiento del proyecto, será demostrado a escala en una pequeña vivienda con su respectiva instalación eléctrica.

CAPÍTULO 1

1. INTRODUCCIÓN

1.1 Antecedentes y Justificativos

Antiguamente, una vivienda era un lugar de resguardo para vivir con un mínimo de instalaciones; en la actualidad, se busca la comodidad. Gracias a las nuevas tecnologías, se puede conseguir controlar las ventanas y las luces de una casa desde la Internet, logrando encender la calefacción con una llamada desde un móvil, etc. Con esta automatización de las viviendas, se consigue una mayor comodidad con un menor esfuerzo.

El concepto de domótica, nace en EEUU a finales de los 70 y principios de los 80, gracias al auge de las telecomunicaciones y un período próspero en la construcción de edificios inteligentes (cuando va aplicado a edificios), edificios pre-cableados (cuando incorporan una red de comunicaciones voz/datos estructurada y universal), edificios de altas tecnologías (capaces de utilizar tecnologías avanzadas de comunicación e información), edificios automatizados (cuando incorporan instalaciones de control de servicios técnicos y seguridad) u otros hogares automatizados (cuando se refiere a la vivienda).

Los franceses, con una clasificación más general de la domótica, indican que ésta se refiere a la vivienda, y la inmótica cuando se relaciona con la edificación no residencial (hospitales, hoteles, estaciones, plantas industriales, etc.).

La domótica, es el conjunto de servicios de una vivienda garantizado por los sistemas que realizan diversas funciones, las cuales pueden estar conectadas entre las a las redes interiores (redes locales) y exteriores de comunicación (Internet). Gracias a estos sistemas se obtiene un notable ahorro energético, una gestión eficaz de la casa, una buena comunicación con el exterior y un gran nivel de seguridad.

1.2 Introducción a la Domótica

1.2.1 Definición del Término Domótica

Actualmente, los sistemas para el control del hogar integran automatización, informática y nuevas tecnologías de la información. Para sintetizar esta nueva filosofía aplicada al sector doméstico, se ha acuñado un nuevo neologismo, domótica: "tecnología aplicada al hogar", formado por la raíz latina domus (casa), define todas las funciones y servicios proporcionados por una vivienda inteligente. En francés se utiliza un término similar, ‘domotique’ formado por “domus” y “robotique” (robótica), y en inglés se utiliza la expresión ‘nome systems’ o ‘smart house’.

Según el diccionario “Larousse” de la Real Academia de la Lengua Francesa, la domótica es el “conjunto de servicios proporcionados por sistemas tecnológicos integrados, como el mejor medio para satisfacer estas necesidades básicas de seguridad, comunicación, gestión energética y confort, del hombre y de su entorno más cercano”.

Así pues, se puede decir que la domótica, es el resultado de la integración de los sistemas de gestión de seguridad, comunicaciones, gestión del confort y control energético conforme lo indica la siguiente figura y detalle:

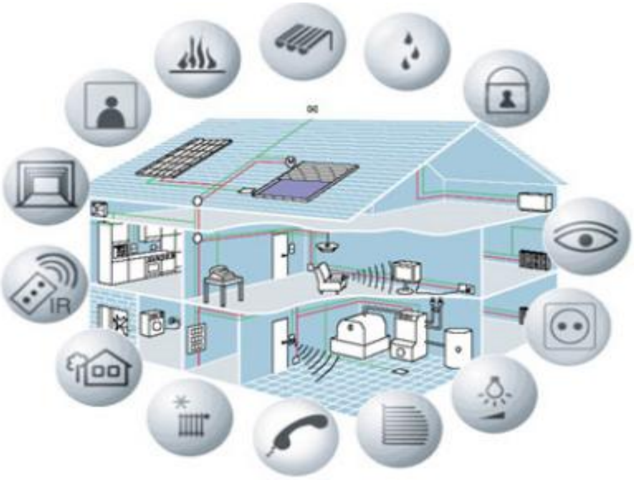


Figura 1.1 Integración de sistemas en vivienda inteligente

- SEGURIDAD: De las personas y los bienes materiales delante de las agresiones externas e intrusiones.
- COMUNICACIONES: Integración de sistemas de enlace entre el interior (red local) y el exterior de la casa (Internet).
- GESTION: Incluye todos aquellos automatismos que no se puede clasificar en ningún otro lugar, como por ejemplo ventanas, luces, audio, video, etc.
- ENERGÍA: Control y distribución adecuada y eficiente de la energía.

1.2.2 Los Dispositivos

La amplitud de una solución de domótica puede variar, desde un único dispositivo que realiza una sola acción, hasta amplios sistemas que controlan prácticamente todas las instalaciones dentro de la vivienda. Los distintos dispositivos de los sistemas de domótica se pueden clasificar en los siguientes grupos:

- **Controlador:** Los controladores son los dispositivos que gestionan el sistema según la programación y la información que reciben. Puede haber un solo controlador, o varios distribuidos por el sistema.

- **Actuador:** El actuador es un dispositivo capaz de ejecutar y/o recibir una orden del controlador y realizar una acción sobre un aparato o sistema (encendido/apagado, subida/bajada, apertura/cierre, etc.).
- **Sensor**: El sensor es el dispositivo que monitoriza el entorno, captando así información que transmite al sistema (sensores de agua, gas, humo, temperatura, viento, humedad, lluvia, iluminación, etc.).
- **Bus:** Es el medio de transmisión que transporta la información entre los distintos dispositivos por un cableado propio, por la redes de otros sistemas (red eléctrica, red telefónica, red de datos) o de forma inalámbrica.
- **Interfaz:** Las interfaces se refiere a los dispositivos (pantallas, móvil, Internet, conectores) y los formatos (binario, audio) en que se muestra la información del

sistema para los usuarios (u otros sistemas) y donde los mismos pueden interactuar con el sistema.

Es preciso, destacar que todos los dispositivos del sistema de domótica no tienen que estar físicamente separados; sino, varias funcionalidades pueden estar combinadas en un equipo. Por ejemplo, un equipo de Central de Domótica puede estar compuesto por un controlador, actuadores, sensores y varias interfaces.

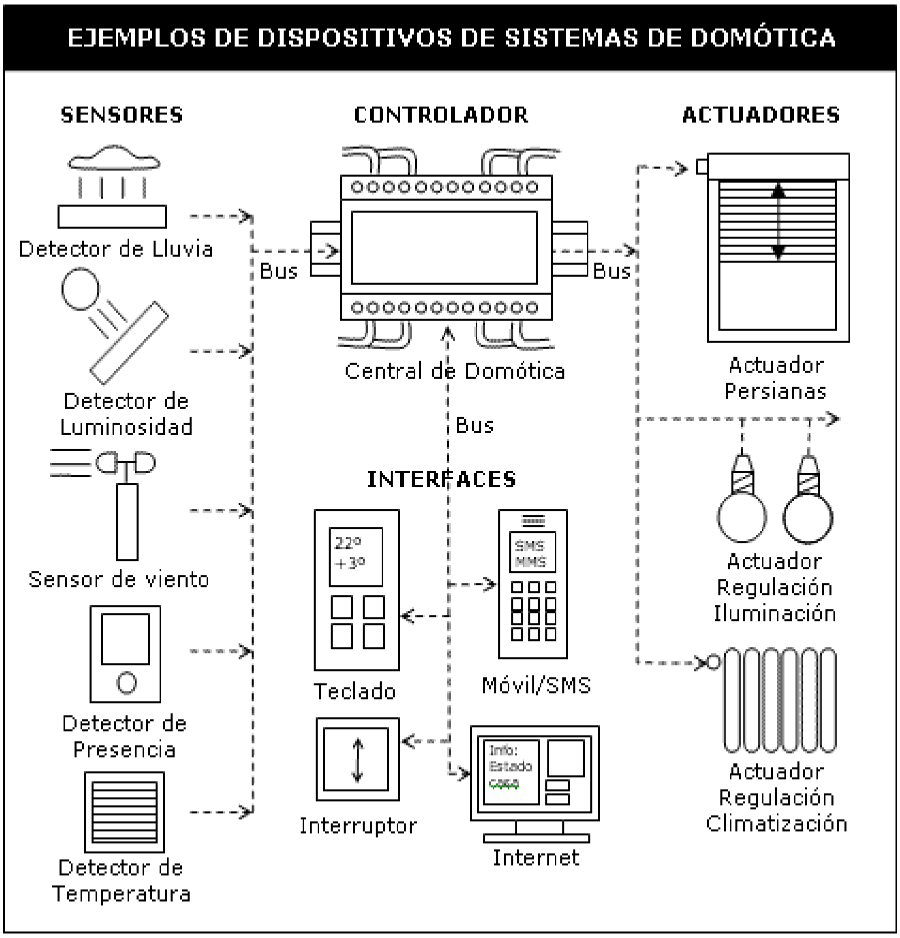


Figura 1.2 Dispositivos de Sistemas de Domótica

1.2.3 Sistemas de Control

Los tipos básicos de las redes domóticas que podemos encontrar son dos:

- Sistemas punto a punto.
- Sistema de bus.

Los sistemas punto a punto, son sensores y actuadores cableados directamente a un único modo central, como por ejemplo, las alarmas normales de hogar o el sistema Simón Vox de Simón. Sus principales ventajas son, el parecido a las instalaciones eléctricas actuales de una casa y el bajo coste de la centralita. Los inconvenientes son, el elevado número de cables y la dependencia directa de la unidad de control.

Los sistemas de bus, son diferentes nodos más o menos inteligentes, comunicados a través de un bus. Estos sistemas pueden ser centralizados o descentralizados. Las principales ventajas respecto a los sistemas punto a punto son que admiten mayores distancias. Los inconvenientes son su elevado precio, el problema con la normalización y consecuentemente la compatibilidad de los protocolos y que el cableado no es tan simple como se podría pensar inicialmente.

Las estructuras de redes domóticas que se puede encontrar son en estrella, en anillo, en árbol y una combinación de estas dependiendo de la aplicación. El control siempre podrá ser centralizado o descentralizado.

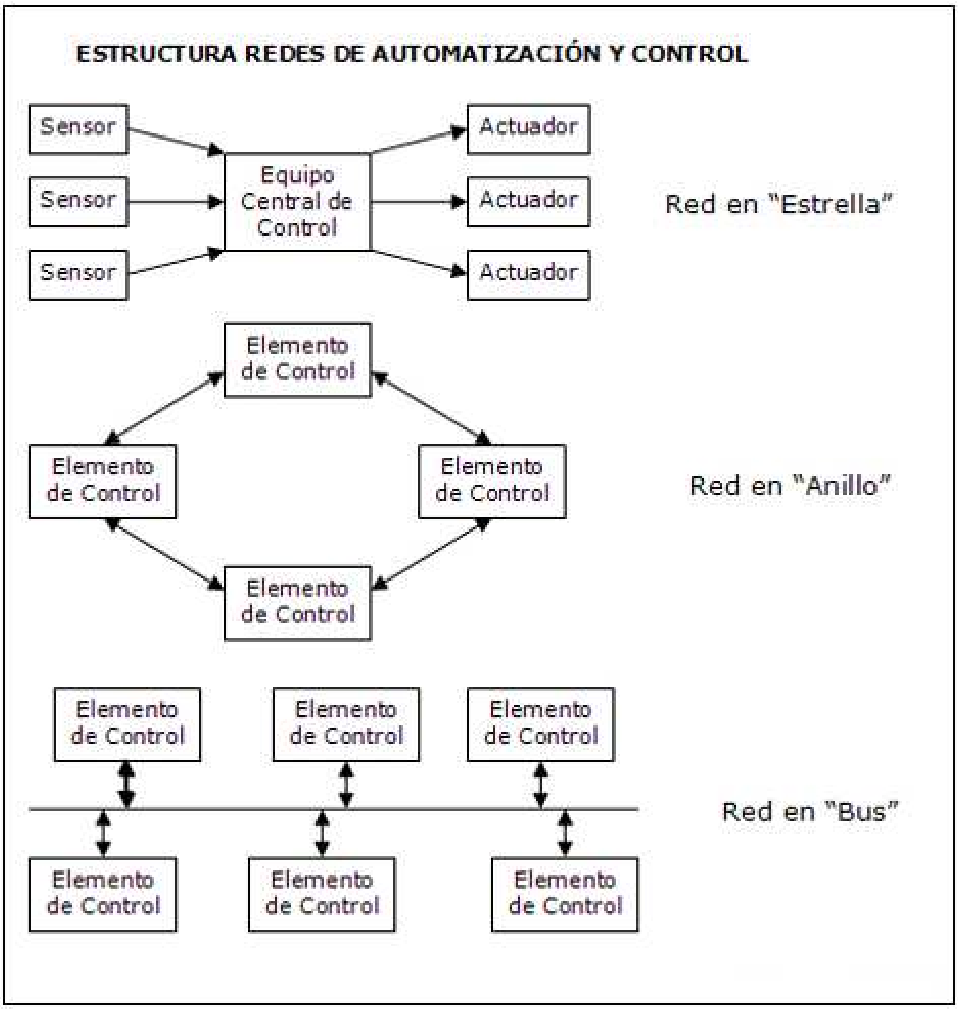


Figura 1.3 Estructuras de Redes de Automatización y Control

1.2.4 La Arquitectura

La Arquitectura de los sistemas de domótica hace referencia a la estructura de su red. La clasificación se realiza en base de donde reside la “inteligencia” del sistema domótico. Las principales arquitecturas son:

Arquitectura Centralizada: En un sistema de domótica de arquitectura centralizada, un controlador centralizado, envía la información a los actuadores e interfaces según el programa, la configuración y la información que recibe de los sensores, sistemas interconectados y usuarios.

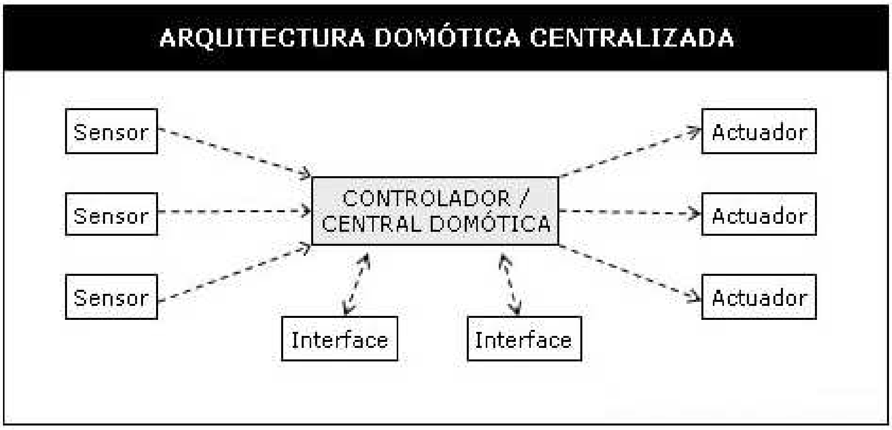


Figura 1.4 Esquema de una Arquitectura Domótica Centralizada

Arquitectura Descentralizada: En un sistema de domótica de Arquitectura Descentralizada, hay varios controladores, interconectados por un bus, que envía información entre ellos y a los actuadores e interfaces conectados a los controladores, según el programa, la configuración y la información que recibe de los sensores, sistemas interconectados y usuarios.

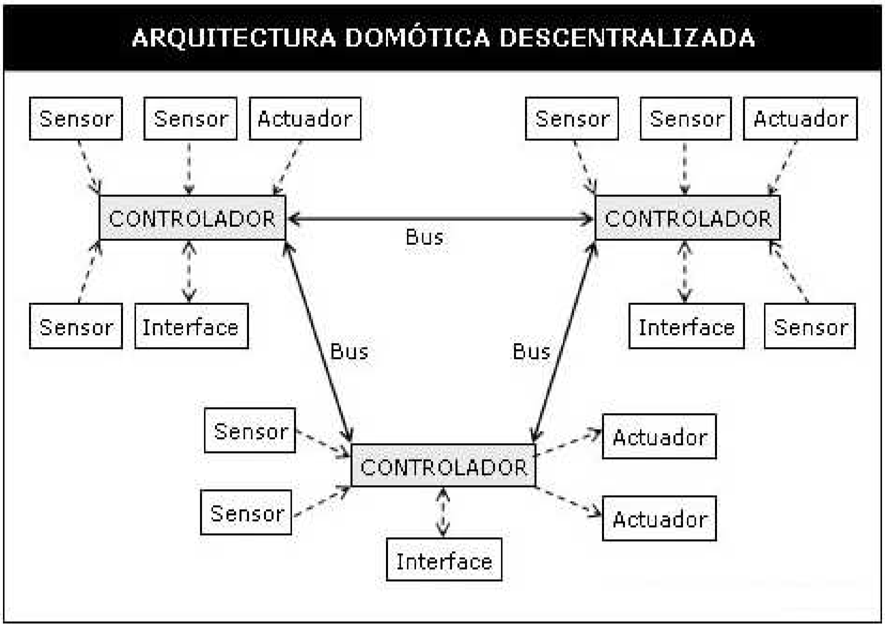


Figura 1.5 Esquema de una Arquitectura Domótica Descentralizada

Arquitectura Distribuida - En un sistema de domótica de arquitectura distribuida, cada sensor y actuador es también un controlador, capaz de actuar y enviar información al sistema según el programa, la configuración, la información que capta por sí mismo y la que recibe de los otros dispositivos del sistema.

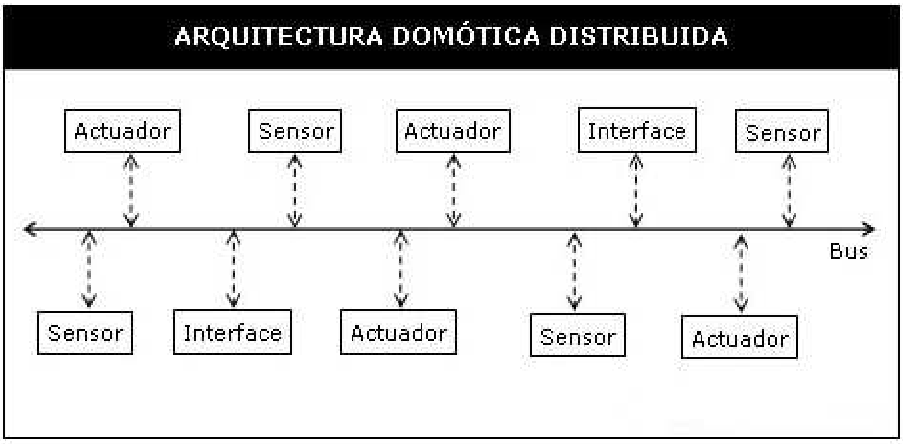


Figura 1.6 Esquema de una Arquitectura Domótica Distribuida

Arquitectura Híbrida / Mixta: En un sistema de domótica de arquitectura híbrida (también denominado arquitectura mixta), se combinan las arquitecturas de los sistemas centralizadas, descentralizadas y distribuidas. A la vez que, puede disponer de un controlador central o varios controladores descentralizados, los dispositivos de interfaces, sensores y actuadores pueden también ser controladores (como en un sistema “distribuido”) y procesar la información según el programa, la configuración, la información que capta por si mismo; además, actúa enviándola a otros dispositivos de la red, sin que necesariamente pase por otro controlador.

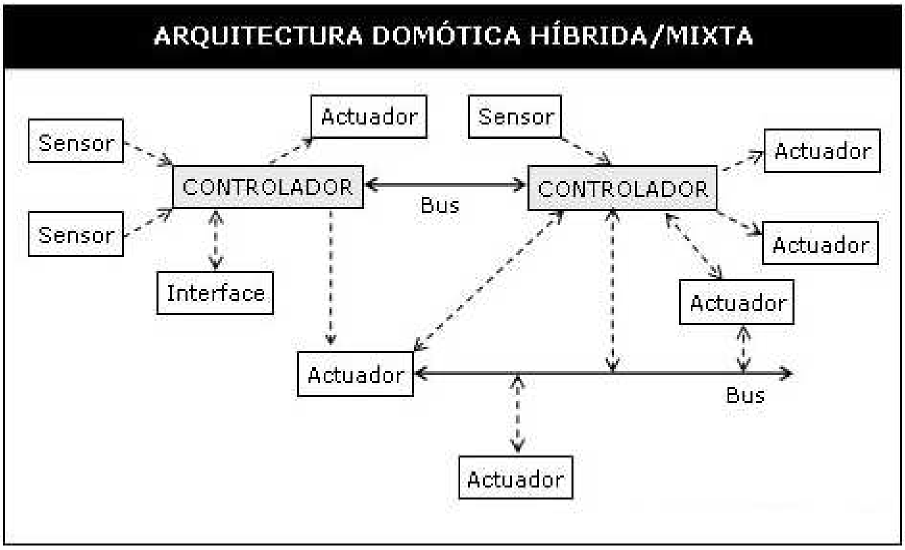


Figura 1.7 Esquema de una Arquitectura Domótica Híbrida / Mixta

1.2.5 Medios de Transmisión

El medio de transmisión de la información, interconexión y control, entre los distintos dispositivos de los sistemas de domótica puede ser de varios tipos. Los principales medios de transmisión son:

- **Cableado Propio:** La transmisión por un cableado propio, es el medio más común para los sistemas de domótica, principalmente son del tipo: par apantallado, par trenzado (1 a 4 pares), coaxial o fibra óptica.
- **Cableado Compartido:** Varias soluciones utilizan cables compartidos y/o redes existentes para la transmisión de

su información, por ejemplo la red eléctrica (corrientes portadoras), la red telefónica o la red de datos.

- **Inalámbrica:** Muchos sistemas de domótica, utilizan soluciones de transmisión inalámbrica entre los distintos dispositivos, principalmente tecnologías de radiofrecuencia o infrarrojo.

Cuando el medio de transmisión es utilizado para transferir información entre dispositivos con la función de "controlador", también se denomina "Bus". El bus también se utiliza muchas veces para alimentar a los dispositivos conectados a él (por ejemplo European Instalation Bus EIB).

1.3 Tecnologías y Protocolos Utilizados en la Domótica

CEBus

En el año 1984, varios miembros de la EIA norteamericana (Electronics Industry Association) especificaron y desarrollaron un estándar llamado CEBus (Consumer Electronic Bus). En 1992, fue presentada la primera especificación.

Se trata de un protocolo para entornos de control distribuidos, que está definido en un conjunto de documentos. Como es una especificación abierta, cualquier empresa puede conseguir estos documentos y fabricar productos que implemente este estándar.

Nivel Físico

Existen varios protocolos para que los electrodomésticos y equipos eléctricos puedan comunicarse dependiendo del medio a utilizar. Los más utilizados son: corrientes portadoras en las líneas de baja tensión, par trenzado con tele alimentación, cable coaxial, infrarrojo, radiofrecuencia y fibra óptica.

Por ejemplo: Para la transmisión de datos por corrientes portadoras, el CEBus usa una modulación en espectro expandido; es decir, los datos se transmiten, uno o varios bits dentro de una ráfaga de señal que comienza en 100 KHz y termina en 400KHz (barrido) de duración 100 microsegundo. La velocidad media de transmisión, es de 7500bps. Hay que destacar, que el nivel físico del estándar CEBus, no cumple la norma europea relativa a transmisión de señal por las líneas de baja tensión (CENELEC EN-50065), por lo que, no es aconsejable utilizarlo en Europa.

Protocolo

En CEBus las tramas definidas pueden tener longitud variable en función de la cantidad de datos que necesitan transmitir. Los tamaños oscilan entre el mínimo de 8 bytes y el máximo de 100 bytes.

Los nodos en CEBus tienen una dirección fija establecida en fábrica que los identifica de forma unívoca en una instalación. Tienen más de 4000 millones de posibilidades.

El lenguaje de programación para el CEBus se llama CAL (Common Application Language) está orientado a objetos (estándar EIA – 600).

CIC

La CIC (CEBus Industry Council) es una asociación de diferentes fabricantes de programas y dispositivos físicos que certifica que los nuevos productos CEBus que salen al mercado cumplan toda la especificación.

La empresa Intellon Corporation dispone del hardware y el protocolo en un solo circuito y además proporcionan el entorno de desarrollo en el lenguaje CAL.

LonWorks / LonTalk

En el año 1992, la empresa Echelon presentó la tecnología LonWorks. Desde el principio, muchas empresas han utilizado esta tecnología para implementar redes de control distribuidas y automatizadas. Su mayor inconveniente es, su precio elevado, que ha imposibilitado su implementación en los hogares.

Esta tecnología ha tenido mucho éxito en las instalaciones profesionales, en las que es mucho más importante la fiabilidad y robustez del sistema que su precio. Destacar su arquitectura descentralizada que permite distribuir la inteligencia entre los sensores y los actuadores.

Según Echelon, su arquitectura es un sistema abierto a cualquier fabricante, pero la patente de su circuito integrado, hace que este sistema sea poco “abierto”.

Todos los nodos o dispositivos LonWorks, están basados en un Microcontrolador llamado Neuron Chip. Este circuito integrado y su firmware que implementa el protocolo LonTalk fue desarrollado por Echelon en 1990.

Lo más destacado del Neuron Chip es:

- Tiene un identificador único, el Neuron ID, es de 48 bits y se graba en la memoria EEPROM durante la fabricación del circuito.

- Tiene un modelo de comunicaciones, que es independiente del medio físico sobre el que funciona; es decir, los datos pueden transmitirse en ondas portadoras, cable trenzado, fibra óptica, cable coaxial, radiofrecuencia, etc.

- El firmware que implementa el protocolo LonTalk, proporciona servicios de transporte y enrutamiento extremo-a-extremo, es decir, está incluido un sistema operativo que ejecuta y planifica la aplicación distribuida y que maneja las estructuras de datos que intercambian los nodos.

Los circuitos, se comunican enviándose telegramas que contienen: la dirección, destino, información para el enrutamiento, datos de control, así como, los datos de la aplicación del usuario y una suma de control como código de detección de errores. Un telegrama, puede tener hasta 229 bytes de información neta para la aplicación distribuida.

Hay dos tipos de datos:

- Mensajes Explícitos.
- Variable de red.

Los mensajes explícitos, son la forma más sencilla de intercambiar datos entre dos aplicaciones.

Las variables de red, proporcionan un modelo estructurado para el intercambio automático de datos distribuidos.

Echelon, ha concedido licencia a tres fabricantes de semiconductores, que pagan altas sumas de dinero por cada circuito fabricado. Son: Toshiba, Motorola y Cypress.

Otros fabricantes de nodos o aplicaciones son Philips, Mitsubishi y ABB.

Medio Físico

El Neuron Chip proporciona un puerto específico de cinco pines, que puede ser configurado para actuar como interfaz de diversos transceptores de línea y funcionar a diferentes velocidades binarias.

Lon Mark.

Lon Mark, es una asociación de fabricantes que desarrollan productos o servicios basados en redes de control LonWorks. Se encarga de especificar y publicar las recomendaciones e implementaciones, que mejor se adaptan a cada uno de los dispositivos típicos de las redes de control.

SCP

El SCP (Simple Control Protocolo) es un intento de Microsoft y General Electric, por generar un protocolo para redes de control, que consiga afianzarse como la solución en todas las aplicaciones de automatización de edificios y viviendas.

Es un protocolo abierto y libre de derechos. Esta iniciativa, tiene dos años de vida. Su principal característica es, la intención de unificar o poner orden en la amplia variedad de protocolos que existe hoy en los EEUU.

En esta convergencia, trabajan el CIC (CEBus Industry Council) y las empresas que desarrollan UPnP (Universal Plug&Play). Es importante destacar que, el UPnP es una iniciativa de Microsoft y es competencia directa del Jini que lidera Sun Microsystems.

Nivel Físico.

Su nivel físico, se basa en la transmisión de datos por las líneas eléctricas de baja tensión (ondas portadoras) del CEBus.

Las empresas Domosys, ITRAN Communications Ltd y Mitsubishi Electric Corporation, están desarrollando, circuitos integrados que implementan la especificación SCP en poco espacio y a bajo costo, haciendo posible su uso en una multitud de electrodomésticos, dispositivos eléctricos y equipos de consumo de las viviendas.

También, está previsto el desarrollo en medios físicos como el par trenzado y la radiofrecuencia.

Protocolo

Este protocolo, está optimizado para su uso en dispositivos eléctricos y electrónicos que tienen una memoria y capacidad de proceso ilimitada.

Como otros buses o protocolos de control distribuido, el SCP está diseñado para funcionar sobre redes de control con un ancho de banda muy pequeño (<10Kbps), y funcionar correctamente con las condiciones de ruido de la red eléctrica de baja tensión, estas redes se llaman PLC (Power Line Communication).

BACnet

Es un protocolo norteamericano, para la automatización de viviendas y redes de control que fue desarrollado bajo el patrocinio de una asociación de fabricantes e instaladores de equipos de calefacción y aire acondicionado, a finales de los ochenta.

Su principal objetivo, era crear un protocolo abierto; es decir, no propietario, que permitiera interconectar los sistemas de aire acondicionado y calefacción de las viviendas y edificios que permitiera una óptima gestión energética.

Inicialmente se definió un protocolo que implementaba arquitectura OSI y se decidió empezar usando, como soporte de nivel físico, la tecnología RS-485, similar al RS-232 ,pero sobre un par trenzado y transmisión diferenciada de la señal, para hacer más inmune esta a las interferencias electromagnéticas.

La parte más destacable de este protocolo, es el esfuerzo que ha realizado para definir un conjunto de reglas hardware y software, que permite comunicarse a dos dispositivos independientes entre si, estos usan protocolo como: el EIB, el BAtiBIS, el EHS, el LonTalk, TCP/IP, etc.

El BACnet no está cerrado a un nivel físico o a un protocolo de nivel 3 concreto. Define la forma en que se representan las funciones que puede hacer cada dispositivo, llamados “objetos”, cada una con sus respectivas propiedades. Como por ejemplo, entrada/salida analógicas, digitales, bucles de control (PID), etc. Algunas propiedades son obligatorias y otras optativas, pero siempre debe configurarse la dirección o identificador de dispositivo para localizar este dentro de una instalación BACnet.

HAVi

Es una iniciativa de los fabricantes más importantes en equipos de entretenimiento tales como: Grundig, Hitachi, Panasonic, Philips, Sharp, Sony, Thompson y Toshiba. Necesitaban un sistema estándar que permita compartir recursos y servicios entre televisión, los equipos HiFi, los videos, reproductores DVD, etc. El HAVi, es una especificación software que permite la comunicación y ejecución total entre estos.

Con el HAVi los usuarios podrán utilizar la pantalla de televisión para gobernar el equipo de música, el video, el reproductor de DVD, la cámara de video, la videoconsola y todos los aparatos conectados en una misma vivienda sin necesidad de estar cerca de esta. Se podrá escuchar música en la habitación con los CDs del equipo del salón, utilizar un ordenador para reproducir películas DVD mientras estamos en la cocina, etc. Como todos estos equipos están enlazados en una red domótica, se podrá disminuir el volumen cuando suene el teléfono o llamen a la puerta. Los sistemas de alarma y anti-incendio podrán utilizar la televisión como consola de visualización.

Tecnología

El HAVi, ha sido desarrollado para cubrir las demandas de intercambio de información, entre los equipos de video y audio digitales, que hay en los hogares actuales.

Si adquirimos un equipo con el logo HAVi, de alguno de los fabricantes antes mencionados, debemos asegurarnos que:

- La interoperabilidad será total, cualquier otro dispositivo HAVi podrá gobernar al nuevo y este a la vez podrá gobernar a los que ya tenemos.
- Compatibilidad entre dispositivos de fabricantes diferentes.
- Plug & Play inmediato. Una vez conectado el bus IEEE 1394 al nuevo dispositivo este se anunciará al resto de equipos HAVi instalados en la vivienda, ofreciendo sus funciones y servicios a los demás.
- Podremos descargar de Internet las nuevas versiones de software, controladores, que actualizan las prestaciones del equipo adecuándolo así, a las necesidades de cada usuario o de su entorno.

Nivel Físico

El HAVi ha escogido el estándar IEEE 1394 (llamado “i.link” o “FireWire”), como soporte físico de los paquetes de datos. Este estándar, que alcanza velocidades de hasta 500 Mbps, es capaz de distribuir al mismo tiempo diversos paquetes de datos de audio y video, entre diferentes equipos de una vivienda; además, de todos los paquetes de control necesarios para la correcta distribución y gestión de todos los servicios.

Jini

Esta tecnología, ha sido desarrollada por Sun Microsystems, proporcionando un mecanismo sencillo, para que diversos dispositivos conectados a una red puedan colaborar, y compartir recursos sin necesidad de que el usuario final tenga que planificar y configurar dicha red. En esta red de equipos llamada "comunidad" cada uno proporciona a los demás los servicios, controladores e interfaces necesarios para distribuirse de forma óptima la carga del trabajo o las tareas que deben realizar.

Tiene un procedimiento conocido con el nombre de "discovery" que hace que cualquier dispositivo que se conecte a la red sea capaz de ofrecer sus recursos a los demás, informando su capacidad de procesamiento, memoria y las funciones que es capaz de realizar. Al finalizar el discovery se ejecuta el procedimiento "join", asignándole una dirección fija, es decir, una posición en la red.

Esta arquitectura es totalmente distribuida, no hay ningún dispositivo que haga el papel del controlador central o maestro de la red. Todos pueden hablar con todos y ofrecer sus servicios a los demás. No es necesario el uso de un PC que controle a los dispositivos conectados a la red.

Tecnología

Este sistema desarrollado por Sun Microsystems, ha aprovechado la experiencia de muchos de los conceptos en los que está inspirado el lenguaje Java, y más concretamente, en la filosofía de la Máquina Virtual Java (JVM). El Jini puede funcionar en potentes estaciones de trabajo, en PCs, en pequeños dispositivos como: cámara de fotos, agendas electrónicas personales, móviles, etc, o en electrodomésticos de línea marrón o blanca como TV, Videos, HiFi, etc.

UPnP

UpnP (Universal Plug&Play) es una arquitectura software abierta y distribuida, permitiendo a las aplicaciones de los dispositivos conectados a una red el intercambio de información y datos, de forma sencilla y transparente para el usuario final. Esta arquitectura software, está por encima y es independiente de protocolos como: el TCP, el UDP, el IP, etc.

El UPnP se encarga de todos los procesos necesarios para que un dispositivo conectado a una red, pueda intercambiar información con el resto. Es independiente del fabricante, sistema operativo, lenguaje de programación de cada dispositivo y del medio físico utilizado para implementar la red.

El protocolo descubre cuando un nuevo equipo o dispositivo se conecta a la red, asignándole una dirección IP, un nombre lógico, informando a los demás de su capacidad de procesamiento y de su memoria, a más de las funciones que es capaz de cumplir. El usuario final, de esta forma, no tiene que preocuparse de configurar la red.

HAPi

(Home API) es un grupo de trabajo promovido por diferentes empresas cuyo objetivo es la especificación y desarrollo de un conjunto de servidores e interfaces de programación (Application Program Interface (API)) orientados hacia la automatización y control de las viviendas.

Es una iniciativa puramente orientada al software y que probablemente permitirá que diversas aplicaciones de control puedan funcionar sobre diferentes protocolos; como por ejemplo, el CEBus, el LonWorks, el HAVi e incluso las redes locales basadas en Ethernet y TCP/IP.

El HAPi facilitará la labor de los programadores de aplicaciones domóticas o de gestión de vivienda, con la creación de primitivas o APIs comunes para todos, que les permitirán aumentar la portabilidad de las aplicaciones y el reaprovechamiento del código que ha sido especialmente diseñado para el control de dispositivos de vivienda.

Su mayor colaborador es el gigante Microsoft con un desarrollo HAPi para su sistema operativo Windows.

TCP/IP

TCP/IP (Transmisión Control Protocol / Internet Protocol) más que un protocolo, es un conjunto de protocolos que definen una serie de reglas y primitivas permitiendo a una serie de máquinas muy distintas intercambiarse información mediante el uso de redes de área local "LANs" redes de área extensa "WAN", redes públicas de telefonía, etc. El mayor ejemplo es la Internet.

Los protocolos TCP/IP no están optimizados para la domótica. Los protocolos destinados a la domótica, están diseñados de forma que la parte útil de la trama sea lo mayor posible respecto a los datos de control (direcciones y CRC). Los especialistas, suelen usar dos formas para medir este factor:

- Ancho de banda neto, en bits por segundo (bps). Un bus inyecta un flujo de 5400 bps de los que solo 4800 bps son útiles para la aplicación. Aquí el protocolo añade 600bps.

- Overhead o tara. Medido en %. En una trama de 55 bytes, son útiles 50 bytes, la tara sería del 10 %.

Cuando se usan los protocolos TCP/IP para transferir pequeñas cantidades de datos, el coste en ancho de banda es muy alto, por ejemplo, para enviar 2 bytes, necesitamos como mínimo 20 bytes solo de campos de control.

TCP/IP, es un protocolo muy estandarizado y muy utilizado por infinidad de ordenadores y otras aplicaciones, cosa que hace de él, la herramienta ideal para asegurar la interconectividad. Además, el abaratamiento de los microcontroladores y sus grandes velocidades, hace que incorpore un Microcontrolador que gestione una caldera, aire acondicionado, horno, nevera, etc.; mediante una conexión a Internet, sin tener un incremento adicional en el coste final del producto.

ZigBee

ZigBee es una alianza, sin ánimo de lucro de 25 empresas, la mayoría de ellas fabricantes de semiconductores, con el objetivo de auspiciar el desarrollo de implantación de una tecnología inalámbrica vía radio y bidireccional de bajo coste. Justifican el desarrollo de este estándar para cubrir el vacío que se produce por debajo del Bluetooth.

ZigBee, también es conocido con otros nombres como “Homero Lite”, es una tecnología inalámbrica con velocidades comprendidas entre

20 KB/s y 250 KB/s y rangos de 10 m a 75 m. Puede usar las bandas libres para frecuencias de uso industrial, científica y médica (ISM), de 204 GHz, 868 MHz (Europa) y 915 MHz (EEUU). Una red puede estar formada por hasta 255 nodos. La mayor parte del tiempo el transceptor ZigBee está dormido con objeto de reducir el consumo. La idea es alimentar un sensor ZigBee con dos pilas AA durante un mínimo de 6 meses y llegar hasta 2 años.

X-10

Diseñado en Escocia entre los años 1976 y 1978, con el objetivo de transmitir datos por las líneas de baja tensión a muy baja velocidad (60 bps en EEUU y 50 bps en Europa) y costes muy bajos. Es uno de los protocolos más antiguos de los que actualmente se utilizan en la domótica. Como se utiliza la red eléctrica no es necesario instalar nuevos cables para la comunicación.

Es un protocolo no propietario, es decir, libre , que cualquier persona puede fabricar dispositivos X-10. Está obligado a utilizar los circuitos del fabricante escocés que diseñó esta alta tecnología, pero se destaca el precio de los impuestos de patente por ser simbólico.

Actualmente, podemos encontrar en Europa tres grandes familias de productos basados en X-10, teóricamente son compatibles entre sí.

- Netzbus.
- Timac
- Home Systems.

Su precio es muy competitivo, de tal forma que es líder en el mercado norteamericano residencial y pequeñas empresas. Se puede afirmar que el X-10 es ahora mismo, la tecnología más asequible para realizar una instalación domótica no muy compleja.

Nivel Físico.

El protocolo de corrientes portadoras utilizado en X-10 es muy sencillo. Consiste en enviar la información modulada en amplitud (AM) y de frecuencia fija a 120 KHZ, con una amplitud de 6Vpp en Estados Unidos y de 2.8Vpp en Europa. La señal se puede insertar tanto en un semiciclo positivo como negativo de la onda senoidal, la codificación de un bit a '1' o un bit a '0' depende de cómo se inyecte esta señal en los semiciclos. Esto significa, que el protocolo denota un '1' lógico enviando una ráfaga a 120 KHz a 6V de amplitud durante un milisegundo sobre la red de 60 Hz, y un '0' lógico mediante la ausencia de señal superpuesta a la red eléctrica. En Europa, se representa el '1' lógico enviando una ráfaga de 120 KHz y 2.8V de amplitud durante 1 milisegundo sobre la red de 50 HZ y un ‘0’ lógico mediante la ausencia de señal.

En un sistema trifásico, el pulso de 1 milisegundo se transmite tres veces para que coincida con el paso por cero en las tres fases. La velocidad binaria, viene impuesta por la frecuencia de la red, en Europa es de 50 bps y en EEUU 60 bps.

La transmisión completa de una orden en X-10 necesita once ciclos de corriente. Esta trama se divide en tres campos de información.

1. Dos ciclos representan el Código de Inicio.
2. Cuatro ciclos representan el Código de casa (letras A-P).
3. Cinco ciclos representan o bien el Código Numérico (1-16) o bien el Código de Función (Encender, Apagar, Aumento de Intensidad, etc.)

Para aumentar la fiabilidad del sistema, la trama (Código de Inicio, Código de Casa y Código de Función o Numérico) se transmite siempre dos veces, separándolas por tres ciclos completos de corriente. En funciones de regulación de intensidad, se transmiten de forma continuada (por lo menos dos veces) sin separación de tramas.

Protocolo

Hay tres tipos de dispositivos X-10, los que solo pueden transmitir órdenes, los que solo pueden recibirlas y los que pueden enviar y recibir órdenes.

Los transmisores pueden direccionar hasta 256 receptores. Los receptores vienen dotados de dos pequeños conmutadores giratorios, uno con 16 letras y el otro con 16 números, que permiten asignar una dirección de las 256 posibles. En una instalación puede haber más de un receptor con la misma dirección, todos actuarán a la orden que se les dirija.

Los dispositivos bidireccionales, tienen la capacidad de responder y confirmar la correcta realización de una orden. Esto puede ser útil en sistemas conectados a un ordenador que muestre los estados en que se encuentra una instalación domótica de la vivienda.

BatiBus (convergencia a Konnex).

Batí Bush fue una iniciativa de MERLIN GERIN, AIRELEC, EDF y LANDIS & GYR. Orientado a la gestión técnica de edificios, es un sistema formado por sensores de unión y actuadores con un cierto grado de “inteligencia” que se utilizan para gobernar puertas, ventanas, alarmas antirrobo, de humo, etc. Este protocolo necesita un cable dedicado que una todos los nodos. La alimentación de los periféricos puede ser local o del propio bus, con su máximo de 75 puntos tele alimentados. La instalación de este cable se puede hacer en diversas topologías: bus, estrella, anillo, árbol o cualquier combinación de estas.

La velocidad binaria es única de 4800 bps, la cual es más que suficiente para la mayoría de las aplicaciones de control distribuido. Tuvo mucha importancia en Francia pero no ha avanzado y se ha quedado como tecnología obsoleta.

A partir de LonWork y BatiBus se pueden encontrar entre otros: BITBus, Profibus, Modbus Modicon, FIP, S-BUS y MIL-STD-1553B.

Protocolo

Este protocolo es totalmente abierto, es decir, no propietario.

A nivel de acceso, este protocolo usa la técnica de acceso múltiple por detección de portadora con evasión de colisiones (CSMA-CA), similar a Ethernet pero con resolución positiva de las colisiones. Si dos dispositivos intentan acceder al mismo tiempo al bus, ambos detectan que se está produciendo una colisión, pero solo el que tiene más prioridad continúa transmitiendo, el otro deja de poner señal en el bus. Esta técnica es muy similar a la usada en el bus europeo EIB y también en el bus CAN, del sector Automotriz.

La filosofía consiste en que todos los dispositivos BAtiBUs escuchen lo que ha enviado cualquier otro, todos procesan la información recibida, pero solo aquellos que hayan sido programados para ello filtrarán la trama y la subirán a la aplicación.

Al igual que los dispositivos X-10, todos los dispositivos BatiBUS disponen de unos micro-interruptores circulares o miniteclados que permiten asignar una dirección física y lógica que identifica unívocamente a cada dispositivo conectado al bus.

La BCI (Club Internacional Batibus) es la asociación que ha creado un conjunto de herramientas para facilitar el desarrollo de productos que cumplan esta especificación, y que ha conseguido el certificado como estándar europeo CENELEC.

EIB (convergencia a Konnex).

El Bus de Instalación Europeo (EIB), es un sistema domótico, desarrollado bajo la supervisión favorable de la Unión Europea con el objetivo de contrarrestar las importaciones de productos similares que se estaban produciendo desde el mercado japonés y el norteamericano donde estas tecnologías se han desarrollado mucho antes que en Europa.

El objetivo principal era crear un estándar europeo, con el suficiente número de fabricantes, instaladores y usuarios, que permita comunicarse a todos los dispositivos de una instalación eléctrica como contadores, equipos de climatización, de custodia, de seguridad, de gestión energética y los electrodomésticos.

Está basado en la estructura de niveles OSI y tiene una arquitectura descentralizada. Este estándar permite distribuir la inteligencia entre los sensores y los actuadores instalados en la vivienda.

Nivel Físico.

En un principio, solo se contempló el utilizar un cable de dos hilos como soporte físico de las comunicaciones, se pretendía que el nivel EIB.MAC (Control de Acceso al Medio) pudiera funcionar sobre los siguientes medios físicos.

- EIB.TP: sobre par trenzado a 9600 bps. Por estos dos hilos se suministra 24 Vdc para la alimentación remota de los dispositivos EIB. Usa la técnica CSMA con arbitraje positivo del bus que evita las colisiones, evitando los reintentos y maximizando el ancho de banda disponible.
- EIB.PL: Corrientes portadoras sobre 230 Vac/50Hz (Power Line) a 1200/2400 bps. Usa la modulación SFSK (Spread

Frequency Shift Keying) similar a las FSK pero con las portadoras más separadas. La distancia máxima que se puede lograr sin repetidor es de 600 metros.

- EIB.net: Usando el estándar Ethernet a 10 Mbps (IEC 802-2). Sirve de columna vertebral entre segmentos EIB, además de permitir la transferencia de telegramas EIB a través del protocolo IP a viviendas o edificios remotos.
- EIB.RF: Radiofrecuencia: Usando varias portadoras, se consiguen distancias de hasta 300 metros en campo abierto. Para mayores distancias o edificios con múltiples estancias se pueden usar repetidores.
- EIB.IR: Infrarrojo: Para el uso con instrucciones a distancia en salas o salones donde se pretenda controlar los dispositivos EIB instalados.

En la práctica, solo el par trenzado ha conseguido una implantación masiva mientras que los demás apenas han conseguido una presencia testimonial.

EIBA

La EIBA, es una asociación de 113 empresas europeas, líderes en el mercado eléctrico, que se unieron en 1990 para impulsar el uso e implantación del sistema domótico EIB.

La EIBA tiene su sede en Bruselas y todos sus miembros cubren el 80% de la demanda de equipamiento eléctrico en Europa. Las tareas principales de esta asociación son:

- Fijar las directrices técnicas para el sistema y los productos EIB, así como establecer los procedimientos de ensayo y certificación de calidad.
- Distribuye el conocimiento y las experiencias de las empresas que trabajan sobre el EIB.
- Encargar a laboratorios de ensayo las pertinentes pruebas de calidad.
- Concede a los productos EIB y a los fabricantes de estos una licencia de marca EIB con la que se podrán distribuir los productos.
- Colabora activamente con otros organismos europeos o internacionales en todas las fases de la normalización y adapta el sistema EIB a las normas vigentes.

- Lidera el proceso de convergencia a (Konnex).

EHS (Convergencia a Konnex).

Este estándar EHS (Sistema de Hogar Europeo), fue favorecido por la Comisión Europea en 1984 y ha sido otro de los intentos de crear una tecnología que permitiera la implantación en el mercado residencial de forma masiva. El resultado de esto, fue la especificación EHS en 1992. Está basada en una topología de niveles OSI, y se especifican los niveles físico, de enlace de datos, de red y de aplicación.

Desde el principio, han estado involucrados los fabricantes más importantes de electrodomésticos de línea marrón y blanca, las empresas eléctricas, las operadoras de telecomunicaciones y los fabricantes de equipamiento eléctrico. La principal idea ha sido y sigue siendo, crear un protocolo abierto, que permitiera cubrir las necesidades de interconexión de los productos de todos estos fabricantes y proveedores de servicios, cubrir las necesidades de automatización de la mayoría de las viviendas europeas cuyos propietarios no se pueden permitir el lujo de usar sistemas más potentes pero también más caros como LonWorks, EIB o BatiBus, debido a la mano de obra especializada que requiere su instalación.

El EHS, es el substituto natural por prestaciones y servicios, de la parcela que tienen el CEBus norteamericano y el HBS japonés y supera las prestaciones del X-10.

EHSA.

La asociación EHS, es la encargada de desarrollar y llevar a cabo diversas iniciativas para aumentar el uso de la tecnología de las viviendas europeas.

También se ocupa de la evolución y mejora tecnológica de EHS y de asegurar la compatibilidad entre fabricantes de productos con interfaces EHS.

Nivel Físico.

Entre 1992 y 1995 la EHSA, consiguió el desarrollo de componentes electrónicos que implementaran la primera especificación. Como resultado nació un circuito integrado de ST-Microelectronics (ST7537HS1) que permitía transmitir datos por un canal serie asíncrono a través de las líneas de baja tensión de las viviendas (ondas portadoras o “Power Line Communication” PLC). Esta tecnología basada en modulación FSK, consigue velocidades de 2400 bps hasta 4800 bps y además también puede utilizar cable de par trenzado como soporte de la señal.

Actualmente se están usando y/o desarrollando los siguientes medios físicos.

- PL-2400: Ondas Portadoras a 2400bps.
- TP0: Par Trenzado a 4800 bps (igual al nivel físico del BatiBUS).
- TP1: Par Trenzado / Coaxial a 9600 bps.
- TP2: Par Trenzado a 64 Kbps.
- IR-1200: Infrarrojo a 1200 bps.
- RF-1100: Radiofrecuencia a 1100 bps.

Protocolo.

El protocolo está totalmente abierto, cualquier fabricante asociado a la EHSA puede desarrollar sus propios productos y dispositivos que implementen el EHS.

Con una filosofía Plug&Play, se pretenden dar las siguientes ventajas:

- Compatibilidad total entre dispositivos EHS.
- Configuración automática de los dispositivos, movilidad de los mismos y ampliación sencilla de las instalaciones.

- Compartir un mismo medio físico entre diferentes aplicaciones sin interferirse entre ellas.

Cada dispositivo EHS tiene asociada una subdirección única dentro del mismo segmento de red que además de identificar unívocamente a un nodo, también lleva asociada información para el enrutado de los telegramas por diferentes segmentos de red EHS.

Konnex

EL Konnex, es una iniciativa de tres asociaciones europeas con el objeto de crear un único estándar para la automatización de las viviendas y oficinas:

1. EIBA (European Installation Bus Association).
2. BatiBus Club International.
3. EHSA (European Home Systems Association).

Los objetivos de esta iniciativa son:

- Crear un único estándar para la domótica e inmótica, que cubra todas las necesidades y requisitos de las instalaciones profesionales y residenciales de ámbito europeo.

- Aumentar la presencia de estos buses domóticos en áreas como la climatización.
- Mejorar las prestaciones de los diversos medios físicos de comunicación sobretodo en la tecnología de radiofrecuencia.
- Introducir nuevos modos de funcionamiento que permitan aplicar una filosofía Plug&Play a muchos dispositivos típicos de una vivienda.

Contactar con empresas proveedoras de servicios como las de telecomunicaciones y las eléctricas con el objeto de potenciar las instalaciones de telegestión técnica de las viviendas.

En pocas palabras se trata de unir las fuerzas del EIB, BAtiBus y EHS para crear un único estándar europeo capaz de competir en calidad, prestaciones y precios con otros sistemas norteamericanos como el LonWorks o CEBus.

Actualmente la asociación Konnex está terminando las especificaciones del nuevo estándar (versión 1.0) el cual será compatible con los productos EIB instalados. El nuevo estándar tendrá lo mejor del EIB, BatiBus y EHS; esto aumentará considerablemente la oferta de productos para el mercado residencial el cual ha sido, hasta la fecha, el punto débil de estas tecnologías.

La versión 1.0

La versión 1.0 del Konnex contempla tres modos de funcionamiento.

1. S.mode (System mode): Configuración de sistema, se utiliza la misma configuración que el EIB actual. Los diversos nodos de la nueva instalación son instalados y configurados por profesionales con ayuda de la aplicación software especialmente diseñada para este propósito.

2. E.mode (Easy mode): Configuración fácil, los dispositivos son programados en fábrica para realizar una función concreta. Aún así deben ser configurados algunos detalles en la instalación, ya sea con el uso de un controlador central o mediante unos micro interruptores alojados en el mismo dispositivo.

3. A.mode (Automatic mode): Configuración automática, con una filosofía Plug&Play, ni el instalador, ni el usuario final tienen que configurar el dispositivo. Este modo está especialmente indicado para ser usado en electrodomésticos, equipos de entretenimiento como consolas, televisores, HiFi, etc.

Nivel Físico. (Convergencia)

El nuevo estándar a nivel físico podrá funcionar sobre:

- (TP1) Par Trenzado: Aprovechando la norma EIB equivalente.
- (TP0) Par trenzado: Aprovechando la norma BatiBus equivalente.
- (PL100) Ondas Portadoras: Aprovechando la norma EIB equivalente.
- (PL132) Ondas Portadoras: Aprovechando la norma EHS equivalente.
- Ethernet: Aprovechando la norma EIB.net
- Radiofrecuencia: Aprovechando la norma EIB.RF.

CAPÍTULO 2

2. ETHERNET

2.1 Generalidades

La mayor parte del tráfico en Internet se origina y termina en conexiones de Ethernet. Desde su comienzo, Ethernet ha evolucionado para satisfacer la creciente demanda de redes de área local (LAN), de alta velocidad. En el momento en que aparece un nuevo medio, como la fibra óptica, Ethernet se adapta para sacar ventaja de un ancho de banda superior y de un menor índice de errores que la fibra ofrece. Ahora, el mismo protocolo que transportaba datos a 3 Mbps en 1973 transporta datos a 10 Gbps.

Ethernet no es una tecnología sino una familia de tecnologías de redes de computadoras de área local LAN del tipo de Acceso Múltiple con Detección de Portadora y Detección de Colisiones CSMA/CD, que suelen cumplir con las especificaciones Ethernet, incluyendo IEEE 802.3; basada en tramas de datos. Las estaciones en una LAN CSMA/CD pueden acceder a la red en cualquier momento y, antes de enviar los datos, las estaciones CSMA/CD escuchan la red para ver si ya es operativa. Si lo está, la estación que desea transmitir espera. Si la red no está en uso, la estación transmite. Decimos que se produce una colisión cuando dos estaciones que escuchan el tráfico en la red no oyen nada y transmiten simultáneamente. En este caso, ambas transmisiones quedan desbaratadas y las estaciones deben transmitir de nuevo en otro momento. Los algoritmos Backoff son los que determinan cuando deben retransmitir las estaciones que han colisionado. Ethernet está bien adaptada a las aplicaciones en que el soporte de comunicaciones local a menudo tiene que procesar un elevado tráfico con puntas elevadas de intercambio de datos.

Los estándares Ethernet no necesitan especificar todos los aspectos y funciones necesarios en un Sistema Operativo de Red (NOS). Como ocurre con otros estándares de red, la especificación Ethernet se refiere solamente a las dos primeras capas del modelo de referencia OSI. Estas son la capa física y la de enlace; que proporcionan direccionamiento local, detección de errores, control del acceso a la capa física, entre otras.

2.2 Historia

A principios de 1970, mientras Norm Abramson montaba la red ALOHA, para permitir que varias estaciones de las Islas de Hawai tuvieran acceso estructurado a la banda de radiofrecuencia compartida en la atmósfera; un estudiante recién graduado en el Instituto Tecnológico de Massachusetts MIT llamado Robert Metcalfe se encontraba realizando sus estudios de doctorado en la Universidad de Harvard trabajando para la red de agencias para proyectos de investigación avanzados ARPANET, que era el tema de investigación candente en aquellos días.

En un viaje a Washington Metcalfe estuvo en casa de Steve Crocker donde éste lo dejó dormir en el sofá. Para poder conciliar el sueño Metcalfe empezó a leer una revista científica donde encontró un artículo de Norm Abramson en el que describía la red ALOHA. Metcalfe pensó cómo se podía mejorar el protocolo utilizado por Abramson, y escribió un artículo describiendo un protocolo que mejoraba sustancialmente el rendimiento de ALOHA. Ese artículo se convertiría en su tesis doctoral, que presentó en 1973. La idea básica era muy simple: las estaciones antes de transmitir deberían detectar si el canal ya estaba en uso, en cuyo caso esperarían a que la estación activa terminara. Además, cada estación mientras transmitiera estaría continuamente vigilando el medio físico por si se producía alguna colisión, en cuyo caso se pararía y retransmitiría más tarde. Este protocolo MAC recibiría más tarde la denominación Acceso Múltiple con Detección de Portadora y Detección de Colisiones CSMA/CD.

En 1972 Metcalfe se mudó a California para trabajar en el Centro de Investigación de Xerox en Palo Alto llamado Xerox PARC. Metcalfe encontró un ambiente perfecto para desarrollar sus inquietudes. Se estaban probando unas computadoras denominadas Alto, que ya disponían de capacidades gráficas y ratón; que fueron consideradas las primeras computadoras personales. También se estaban fabricando las primeras impresoras láser. Se quería conectar las computadoras entre sí para compartir ficheros e impresoras. La comunicación tenía que ser de muy alta velocidad, del orden de megabits por segundo, ya que la cantidad de información a enviar a las impresoras era enorme.

A Metcalfe, el especialista en comunicaciones del equipo con 27 años de edad, se le encomendó la tarea de diseñar y construir la red que uniera todo. Contaba para ello con la ayuda de un estudiante de doctorado de Stanford llamado David Boggs. Las primeras experiencias de la red, que se denominaron Red Alto ALOHA, las llevaron a cabo en 1972. Fueron mejorando gradualmente el prototipo hasta que el 22 de mayo de 1973 Metcalfe escribió un memorándum interno en el que informaba de la nueva red. Para evitar que se pudiera pensar que sólo servía para conectar computadoras Alto, se cambió el nombre de la red por el de Ethernet, el nombre hace referencia a la teoría de la física hoy ya abandonada, éter, (donde el medio de transmisión era el cable coaxial por el que iba la señal, que se encargaba de transportar los bits a todas las estaciones de forma muy parecida a la cual las ondas electromagnéticas viajaban por un fluido denominado éter que se suponía llenaba todo el espacio). Las dos computadoras Alto utilizadas para las primeras pruebas de Ethernet fueron rebautizadas con los nombres Michelson y Morley, en alusión a los dos físicos que demostraron en 1887 la inexistencia del éter mediante el famoso experimento que lleva su nombre.

La red de 1973 ya tenía todas las características esenciales de la Ethernet actual. Empleaba CSMA/CD para minimizar la probabilidad de colisión, y en caso de que ésta se produjera se ponía en marcha un mecanismo denominado retroceso exponencial binario para reducir gradualmente la agresividad del emisor, con lo que éste se adaptaba a situaciones de muy diverso nivel de tráfico. Tenía topología de bus y funcionaba a 2,94 Mb/s sobre un segmento de cable coaxial de 1,6Km de longitud. Las direcciones eran de 8 bits y el CRC de las tramas de 16 bits.

En vez de utilizar el cable coaxial de 75 ohm de las redes de televisión por cable se optó por emplear cable de 50 ohm que producía menos reflexiones de la señal, a las cuales Ethernet era muy sensible por transmitir la señal en banda base. Cada empalme del cable y cada ramificación de la red instalada producía la reflexión de una parte de la señal transmitida. En la práctica el número máximo de ramificaciones, y por tanto el número máximo de estaciones en un segmento de cable coaxial, venía limitado por la máxima intensidad de señal reflejada tolerable.

En 1979, Metcalfe dejo Xerox para promover el uso de ordenadores personales y de redes de área local, fundando la corporación 3Com. Convenció a las empresas DEC, Intel y Xerox para que trabajaran juntas en la promoción de Ethernet como un estándar, el llamado estándar DIX. Esta convención estandarizó el Ethernet de 10 Mbps, con direcciones de destino y origen de 48 bits y un campo de 16 bits de tipo de paquete y protocolo. Este estándar se publico el 30 de Septiembre de 1980, en el que participaban dos grandes sistemas de propietarios, anillo con paso de testigo o más conocida como token-ring y la arquitectura de red de área local también llamada ARCNET, pero fueron desbordados por una oleada de productos Ethernet. En este proceso, 3Com se convirtió en una prestigiosa compañía.

Jerome Saltzer co-escribió un importante escrito sugiriendo que las arquitecturas de token-ring son teóricamente superiores a las de Ethernet. Gracias a esto, hubo bastantes dudas para los fabricantes de computadoras y decidieron no hacer a Ethernet un estándar, lo que permitió a 3Com construir un negocio basado en la construcción de tarjetas de red. Esto condujo a decir que Ethernet funciona mejor en la práctica que en la teoría, lo que, irónicamente, en la actualidad constituye un aspecto técnico muy relevante: las características de tráfico típico en las redes actuales difieren de las que se esperaban antes de que las redes LAN se popularizaran gracias al diseño simple de Ethernet. Aparte de esto la relación velocidad y coste de los productos de Ethernet, ha situado a ésta implementación por encima de otras como token-ring, FDDI, ATM, etc, y vemos que hoy día, conectar un PC a la red significa conectarlo vía Ethernet.

2.3 Modelo de Referencia OSI

A principios de la década de 1980, se produjo un enorme crecimiento en la cantidad y el tamaño de las redes. A medida que las empresas tomaron conciencia de las ventajas de usar tecnología de Gestión de la Red de Contactos, comúnmente llamada “Networking”, las redes se agregaban o expandían a casi la misma velocidad a la que se introducían las nuevas tecnologías de red.

Para mediados de la década de 1980, estas empresas comenzaron a sufrir las consecuencias de la rápida expansión. De la misma forma en que las personas que no hablan un mismo idioma tienen dificultades para comunicarse, las redes que utilizaban diferentes especificaciones e implementaciones tenían dificultades para intercambiar información. El mismo problema surgía con las empresas que desarrollaban tecnologías de networking privadas o propietarias, es decir que una sola empresa o un pequeño grupo de empresas controlan todo uso de la tecnología. Las tecnologías de networking que respetaban reglas propietarias en forma estricta no podían comunicarse con tecnologías que usaban reglas propietarias diferentes.

Para enfrentar el problema de incompatibilidad de redes, la Organización Internacional de Normalización ISO investigó modelos de networking como los de la Red de Corporación de Equipos Digitales DECnet, la Arquitectura de Sistemas de Red SNA y TCP/IP a fin de encontrar un conjunto de reglas aplicables de forma general a todas las redes. En base a esta investigación, la ISO desarrolló un modelo de red que ayuda a los fabricantes a crear redes que sean compatibles con otras redes independiente del Fabricante, Arquitectura, Localización y Sistema Operativo; entre los principales.

El modelo de referencia de Interconexión de Sistemas Abiertos OSI lanzado en 1984 fue el modelo de red descriptivo creado por la ISO. Proporcionó a los fabricantes un conjunto de estándares que aseguraron una mayor compatibilidad e interoperabilidad entre los distintos tipos de tecnología de red producidos por las empresas a nivel mundial. En la figura 2.1 se muestra las 7 capas del modelo de referencia OSI, así como las Unidades de datos de protocolo PDU de cada capa respectivamente.

Figura 2.1 Representación gráfica de las capas del modelo OSI

El modelo de referencia OSI se ha convertido en el modelo principal para las comunicaciones por red. Aunque existen otros modelos, la mayoría de los fabricantes de redes relacionan sus productos con el modelo de referencia de OSI. Esto es en particular así cuando lo que buscan es enseñar a los usuarios a utilizar sus productos. Se considera la mejor herramienta disponible para enseñar cómo enviar y recibir datos a través de una red.

2.3.1 Capa Física

La Capa Física del modelo de referencia OSI es la que se encarga de las conexiones físicas de la computadora hacia la red, tanto en lo que se refiere al medio físico, características del medio, así como la forma en la que se transmite la información.

Esta capa es la encargada de transmitir los bits de información a través del medio utilizado para la transmisión. Se ocupa de las propiedades físicas y características eléctricas de los diversos componentes; de la velocidad de transmisión, si ésta es uni o bidireccional. Se encarga de transformar una trama de datos proveniente del nivel de enlace en una señal adecuada al medio físico utilizado en la transmisión. Estos impulsos pueden ser eléctricos, que son transmitidos por cable o electromagnéticos que se transmite sin cables. Estos últimos, dependiendo de la frecuencia y longitud de onda de la señal pueden ser ópticos, de micro-ondas o de radio. Cuando actúa en modo recepción el trabajo es inverso; se encarga de transformar la señal transmitida en tramas de datos binarios que serán entregados al nivel de enlace.

2.3.2 Capa de Enlace de Datos

Cualquier medio de transmisión debe ser capaz de proporcionar una transmisión sin errores, es decir, un tránsito de datos fiable a través de un enlace físico. Debe crear y reconocer los límites de las tramas, así como resolver los problemas derivados del deterioro, pérdida o duplicidad de las tramas. También puede incluir algún mecanismo de regulación del tráfico que evite la saturación de un receptor que sea más lento que el emisor.

La capa de enlace de datos, provee la transmisión de los Bits en tramas de información, es quien revisa que los bits lleguen libres de errores a su destino, controla las secuencias de transmisión y los acuses de recibo de los mensajes recibidos. También se encarga de retransmitir los paquetes o tramas que no han sido acusados por el otro extremo. También este nivel controla el flujo de información entre dos nodos de la red. Un ejemplo de el nivel de enlace de datos es el estándar de ETHERNET o el de ATM.

Además la capa de enlace puede considerarse dividida en dos subcapas:

- Control Lógico de Enlace LLC: define la forma en que los datos son transferidos sobre el medio físico, proporcionando servicio a las capas superiores.
- Control de Acceso al Medio MAC: se refiere a los protocolos que determinan cuál de los computadores de un entorno de medios compartidos o dominio de colisión pueden transmitir los datos.

2.3.3 Capa de Red

Esta capa se ocupa de la transmisión de los paquetes o datagramas y de encaminar cada uno en la dirección adecuada, esta tarea que puede ser complicada en redes grandes como Internet, pero no se ocupa para nada de los errores o pérdidas de paquetes. Define la estructura de direcciones y rutas de Internet. A este nivel se utilizan dos tipos de paquetes: paquetes de datos y paquetes de actualización de ruta. Como consecuencia esta capa puede considerarse subdividida en dos:

- Transporte: Encargada de encapsular los datos de usuario a transmitir. En esta categoría se encuentra el protocolo IP.
- Conmutación: Esta parte es la encargada de intercambiar información de conectividad específica de la red. Los ruteadores o en ocasiones llamados enrutadores; son dispositivos que trabajan en este nivel y se benefician de estos paquetes de actualización de ruta. En esta categoría se encuentra el protocolo ICMP.

Adicionalmente, la capa de red debe gestionar la congestión de red, que es el fenómeno que se produce cuando una saturación de un nodo tira abajo toda la red, similar a al tráfico en horas pico que se produce en una ciudad grande.

2.3.4 Capa de Transporte

Su función básica es aceptar los datos enviados por las capas superiores, dividirlos en pequeñas partes si es necesario, y pasarlos a la capa de red. También se asegura que lleguen correctamente al otro lado de la comunicación. Otra característica a destacar es que debe aislar a las capas superiores de las distintas posibles implementaciones de tecnologías de red en las capas inferiores, lo que la convierte en el corazón de la comunicación.

En esta capa se proveen servicios de conexión para la capa de sesión que serán utilizados finalmente por los usuarios de la red al enviar y recibir paquetes. Estos servicios estarán asociados al tipo de comunicación empleada, la cual puede ser diferente según el requerimiento que se le haga a la capa de transporte. Por ejemplo, la comunicación puede ser manejada para que los paquetes sean entregados en el orden exacto en que se enviaron, asegurando una comunicación punto a punto libre de errores, o sin tener en cuenta el orden de envío. Una de las dos modalidades debe establecerse antes de comenzar la comunicación para que una sesión determinada envíe paquetes, y ése será el tipo de servicio brindado por la capa de transporte hasta que la sesión finalice. Todo el servicio que presta la capa está gestionado por las cabeceras que agrega al paquete a transmitir.

2.3.5 Capa de Sesión

Esta capa establece, gestiona y finaliza las conexiones tales como procesos o aplicaciones entre usuarios finales. Ofrece varios servicios que son cruciales para la comunicación, como son:

- Control de la sesión a establecer entre el emisor y el receptor.
- Control de la concurrencia, es decir, que dos comunicaciones a la misma operación crítica no se efectúen al mismo tiempo.
- Mantener puntos de verificación, que sirven para que, ante una interrupción de transmisión por cualquier causa, la misma se pueda reanudar desde el último punto de verificación en lugar de repetirla desde el principio.

Por lo tanto, el servicio provisto por esta capa es la capacidad de asegurar que, dada una sesión establecida entre dos máquinas, la misma se pueda efectuar para las operaciones definidas de principio a fin, reanudándolas en caso de interrupción. En muchos casos, los servicios de la capa de sesión son parcialmente, o incluso, totalmente prescindibles.

2.3.6 Capa de Presentación

Esta capa se encarga de la representación de la información, de manera que aunque distintos equipos puedan tener diferentes representaciones internas de caracteres, números, sonido o imágenes, los datos lleguen de manera reconocible.

Esta capa es la primera en trabajar más el contenido de la comunicación, que cómo se establece la misma. En ella se tratan aspectos tales como la semántica y la sintaxis de los datos transmitidos, ya que distintas computadoras pueden tener diferentes formas de manejarlas.

Por lo tanto, la capa de presentación es la encargada de manejar las estructuras de datos abstractas y realizar las conversiones de representación de datos necesarias para la correcta interpretación de los mismos. Esta capa también permite cifrar los datos y comprimirlos.

2.3.7 Capa de Aplicación

Todas las capas anteriores en este modelo sirven de mera infraestructura de telecomunicaciones. Por si solas no hacen nada mas que mantener en buen estado el camino para que fluyan los datos, la capa que hace posible que una red se pueda usar es la capa de aplicación.

Esta capa ofrece a las aplicaciones la posibilidad de acceder a los servicios de las demás capas y define los protocolos que utilizan las aplicaciones para intercambiar datos, como correo electrónico, gestores de bases de datos y servidor de ficheros. Hay tantos protocolos como aplicaciones distintas y puesto que continuamente se desarrollan nuevas aplicaciones el número de protocolos crece sin parar.

Cabe aclarar que el usuario normalmente no interactúa directamente con el nivel de aplicación. Suele interactuar con programas que a su vez interactúan con el nivel de aplicación pero ocultando la complejidad subyacente.

2.4 Modelo de Referencia TCP/IP

El Protocolo de Control de Transporte/Protocolo Internet TCP/IP es un conjunto de protocolos diseñado para conjuntos de redes a gran escala que abarcan entornos LAN y WAN. Proviene de los nombres de dos protocolos importantes del conjunto de protocolos, es decir, del protocolo TCP y del protocolo IP.

El estándar histórico y técnico de la Internet es el modelo TCP/IP. Tal como indica la siguiente escala de tiempo en la figura 2.2, los orígenes de TCP/IP se remontan a 1969, cuando el Departamento de Defensa DoD de EE.UU. encargada del proyecto ARPNET, creó el modelo de referencia TCP/IP porque necesitaba diseñar una red que pudiera sobrevivir ante cualquier circunstancia, incluso una guerra nuclear. En un mundo conectado por diferentes tipos de medios de comunicación, como alambres de cobre, microondas, fibras ópticas y enlaces satelitales, el DoD quería que la transmisión de paquetes se realizará cada vez que se iniciaba y bajo cualquier circunstancia. Este difícil problema de diseño dio origen a la creación del modelo TCP/IP.

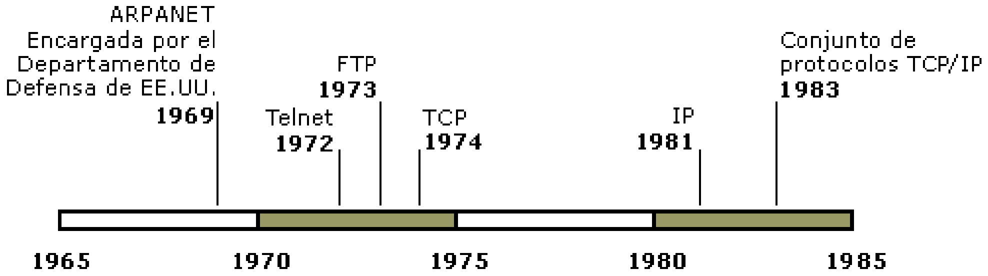


Figura 2.2 Escala de tiempo de los orígenes del TCP/IP

A diferencia de las tecnologías de conexión de redes propietarias, el TCP/IP se desarrolló como un estándar abierto. Esto significaba que cualquier persona podía usar el TCP/IP. Esto contribuyó a acelerar el desarrollo de TCP/IP como un estándar. El modelo TCP/IP tiene las siguientes cuatro capas: La capa de aplicación, transporte, Internet y la de acceso a la red.

Aunque algunas de las capas del modelo TCP/IP tienen el mismo nombre que las capas del modelo OSI, las capas de ambos modelos no se corresponden de manera exacta, la figura 2.3 describe algunas diferencias entre los dos modelos y protocolos que se utilizan en los niveles. Lo más notable es que la capa de aplicación posee funciones diferentes en cada modelo. Los diseñadores de TCP/IP crearon una capa de aplicación que incluye los detalles de las capas de sesión y presentación del modelo de referencia OSI. La relación entre IP y TCP es importante. Se puede pensar en el IP como el que indica el camino a los paquetes, en tanto que el TCP brinda un transporte seguro. IP sirve como protocolo universal que permite que cualquier computador en cualquier parte del mundo pueda comunicarse en cualquier momento.

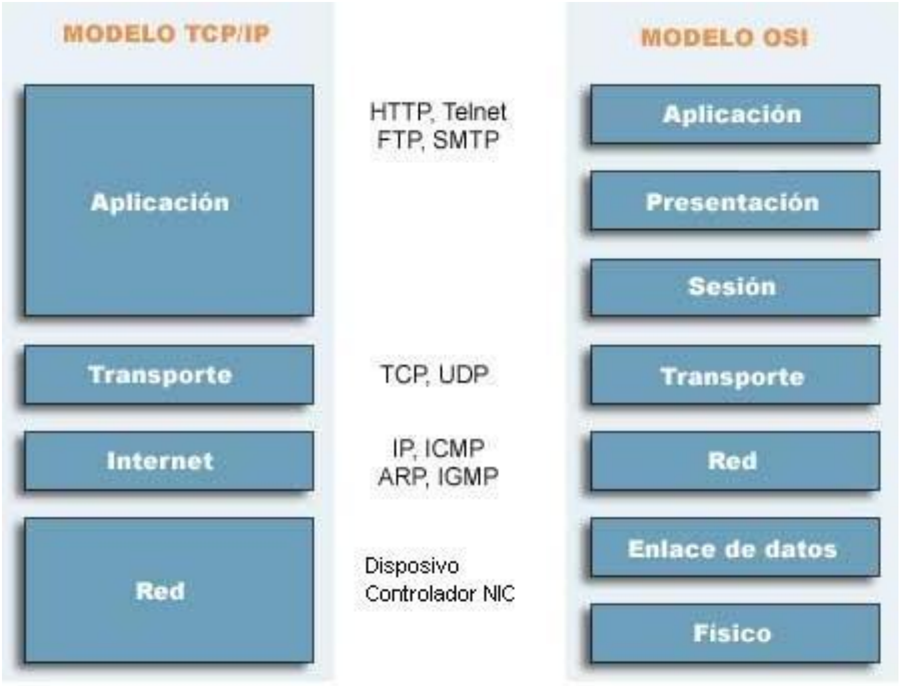


Figura 2.3 Ilustra por capas los dos modelos de referencia OSI y TCP/IP

Comparando el modelo OSI con los modelos TCP/IP, surgen algunas similitudes y diferencias.

Las similitudes incluyen:

- Ambos se dividen en capas.
- Ambos tienen capas de aplicación, aunque incluyen servicios muy distintos.
- Ambos tienen capas de transporte y de red similares.
- Ambos modelos deben ser conocidos por los profesionales de networking.
- Ambos suponen que se conmutan paquetes. Esto significa que los paquetes individuales pueden usar rutas diferentes para llegar al mismo destino. Esto se contrasta con las redes conmutadas por circuito, en las que todos los paquetes toman la misma ruta.

Las diferencias incluyen:

- TCP/IP combina las funciones de la capa de presentación y de sesión en la capa de aplicación.

- TCP/IP combina la capa de enlace de datos y la capa física del modelo OSI en la capa de acceso de red.
- TCP/IP parece ser más simple porque tiene menos capas.
- Los protocolos TCP/IP son los estándares en torno a los cuales se desarrolló la Internet, de modo que la credibilidad del modelo TCP/IP se debe en gran parte a sus protocolos. En comparación; por lo general, las redes no se desarrollan a partir del protocolo OSI, aunque el modelo de referencia OSI se usa en diferentes instituciones educativas como una guía.

2.4.1 Capa de Host a Red

La capa de host a red también se denomina capa de acceso de red. La capa de host a red, es la capa que maneja todos los aspectos que un paquete IP requiere para efectuar un enlace físico real con los medios de la red. Esta capa incluye los detalles de la tecnología LAN, WAN y todos los detalles de la capa física y de enlace de datos del modelo OSI.

Los controladores para las aplicaciones de software, las tarjetas de módem y otros dispositivos operan en la capa de acceso de red. La capa de acceso de red define los procedimientos para realizar la interfaz con el hardware de la red y para tener acceso al medio de transmisión. Debido a un intrincado juego entre las especificaciones del hardware, el software y los medios de transmisión, existen muchos protocolos que operan en esta capa. Esto puede generar confusión en los usuarios. La mayoría de los protocolos reconocibles operan en las capas de transporte y de Internet del modelo TCP/IP.

Las funciones de la capa de acceso de red incluyen la asignación de direcciones IP a las direcciones físicas y el encapsulamiento de los paquetes IP en tramas. Basándose en el tipo de hardware y la interfaz de la red, la capa de acceso de red definirá la conexión con los medios físicos de la misma.

2.4.2 Capa de Internet

La capa Internet, maneja la comunicación de una máquina a otra. Ésta acepta una solicitud para enviar un paquete desde la capa de transporte, junto con una identificación de la máquina, hacia la que se debe enviar el paquete. La capa Internet también maneja la entrada de datagramas, verifica su validez y utiliza un algoritmo de ruteo para decidir si el datagrama debe procesarse de manera local o debe ser transmitido. El protocolo principal que funciona en esta capa es el Protocolo de Internet

IP. La determinación de la mejor ruta y la conmutación de los paquetes ocurren en esta capa.

Los siguientes protocolos operan en la capa de Internet TCP/IP:

- IP proporciona un enrutamiento de paquetes no orientado a conexión de máximo esfuerzo. El IP no se ve afectado por el contenido de los paquetes, sino que busca una ruta de hacia el destino.
- El Protocolo de mensajes de control en Internet (ICMP) suministra capacidades de control y envío de mensajes.
- El Protocolo de resolución de direcciones (ARP) determina la dirección de la capa de enlace de datos, la dirección MAC, para las direcciones IP conocidas.
- El Protocolo de resolución inversa de direcciones (RARP) determina las direcciones IP cuando se conoce la dirección MAC.

2.4.3 Capa de Transporte

La capa de transporte proporciona servicios de transporte desde el host origen hacia el host destino. Esta capa forma una conexión lógica entre los puntos finales de la red, el host transmisor y el host receptor. Los protocolos de transporte segmentan y reensamblan los datos mandados por las capas superiores en el mismo flujo de datos, o conexión lógica entre los extremos. La corriente de datos de la capa de transporte brinda transporte de extremo a extremo.

Generalmente, se compara la Internet con una nube. La capa de transporte envía los paquetes de datos desde la fuente transmisora hacia el destino receptor a través de la nube. El control de punta a punta, que se proporciona con las ventanas deslizantes y la confiabilidad de los números de secuencia y acuses de recibo, es el deber básico de la capa de transporte cuando utiliza TCP. La capa de transporte también define la conectividad de extremo a extremo entre las aplicaciones de los hosts. Los servicios de transporte incluyen los siguientes servicios:

TCP y UDP

- Segmentación de los datos de capa superior
- Envío de los segmentos desde un dispositivo en un extremo a otro dispositivo en otro extremo.

TCP solamente

- Establecimiento de operaciones de punta a punta.
- Control de flujo proporcionado por ventanas deslizantes.
- Confiabilidad proporcionada por los números de secuencia y los acuses de recibo.

2.4.4 Capa de Aplicación

La capa de aplicación del modelo TCP/IP, es el nivel que los programas más comunes utilizan para comunicarse a través de una red con otros programas. Los procesos que acontecen en este nivel son aplicaciones específicas que pasan los datos al nivel de aplicación en el formato que internamente use el programa y es codificado de acuerdo con un protocolo estándar. Algunos programas específicos se considera que se ejecutan en este nivel y proporcionan servicios que directamente trabajan con las aplicaciones de usuario. Estos programas y sus correspondientes protocolos incluyen a HTTP, FTP, SMTP, SSH, DNS y a muchos otros.

En esta capa los usuarios llaman a una aplicación que acceda servicios disponibles a través de la red de redes TCP/IP. Una aplicación interactúa con uno de los protocolos de nivel de transporte para enviar o recibir datos. Cada programa de aplicación selecciona el tipo de transporte necesario, el cual puede ser una secuencia de mensajes individuales o un flujo continuo de octetos. El programa de aplicación pasa los datos en la forma requerida hacia el nivel de transporte para su entrega. Una vez que los datos de la aplicación han sido codificados en un protocolo estándar del nivel de aplicación son pasados hacia el siguiente nivel que es la capa de transporte del modelo TCP/IP.

2.5 Protocolos Utilizados en Modelo TCP/IP

2.5.1 Protocolo de Resolución de Direcciones ARP

El Protocolo de resolución de direcciones ARP es un protocolo de nivel de red responsable de encontrar la dirección hardware o MAC que corresponde a una determinada dirección IP. Para ello, cada equipo conectado a la red tiene un número de identificación de 48 bits. Éste es un número único establecido en la fábrica en el momento de fabricación de la tarjeta. Sin embargo, la comunicación en Internet no utiliza directamente este número, ya que las direcciones de los equipos deberían cambiarse cada vez que se cambia la tarjeta de interfaz de red, sino que utiliza una dirección lógica asignada por un organismo denominada “dirección IP”.

Para que las direcciones físicas se puedan conectar con las direcciones lógicas, el protocolo ARP interroga a los equipos de la red enviando un paquete de petición ARP, para averiguar sus direcciones físicas y luego crea una tabla de búsqueda entre las direcciones lógicas y físicas en una memoria caché.

Cuando un equipo debe comunicarse con otro, consulta la tabla de búsqueda. Si la dirección requerida no se encuentra en la tabla, el protocolo ARP envía una solicitud a la red. Todos los equipos en la red comparan esta dirección lógica con la suya. Si alguno de ellos se identifica con esta dirección, el equipo responderá con un paquete de respuesta ARP, que almacenará el par de direcciones en la tabla de búsqueda, y, a continuación podrá establecerse la comunicación.

2.5.2 Protocolo de Internet IP

Protocolo de Internet IP, es el principal protocolo de la capa de red. Este protocolo define la unidad básica de transferencia de datos entre el origen y el destino, atravesando toda la red de redes. Además, es el encargado de elegir la ruta más adecuada por la que los datos serán enviados. Se trata de un sistema de entrega de paquetes llamados datagramas IP. Algunas características de este es que es un protocolo de datagramas sin conexión y no confiable, responsable principalmente del direccionamiento y enrutamiento de paquetes entre hosts. Sin conexión significa que no se establece una sesión antes de intercambiar datos. No confiable significa que la entrega no está garantizada. IP siempre intenta por todos los medios entregar los paquetes. Un paquete IP se puede perder, entregar fuera de secuencia, duplicar o retrasar. IP no intenta recuperarse de estos tipos de errores. La confirmación de paquetes entregados y la recuperación de paquetes perdidos es responsabilidad de un protocolo de nivel superior, como TCP. Un paquete IP, también llamado datagrama IP, consta de un encabezado IP y una carga IP.

2.5.3 Protocolo de Mensajes de Control de Internet ICMP

El Protocolo de Mensajes de Control de Internet ICMP es el subprotocolo de control y notificación de errores del protocolo de internet. Como tal, se usa para controlar si un paquete no puede alcanzar su destino, si su vida ha expirado, si el encabezamiento lleva un valor no permitido, si es un paquete eco o respuesta, etc.

Los mensajes ICMP se suelen enviar automáticamente en una de las siguientes situaciones:

- Un datagrama IP no puede llegar a su destino.
- Un enrutador IP (puerta de enlace) no puede reenviar datagramas a la velocidad actual de transmisión.
- Un enrutador IP redirige el host que realiza el envío para que utilice una ruta mejor para llegar al destino.

Los mensajes ICMP están encapsulados y se envían dentro de datagramas IP, como se muestra en la figura 2.4.

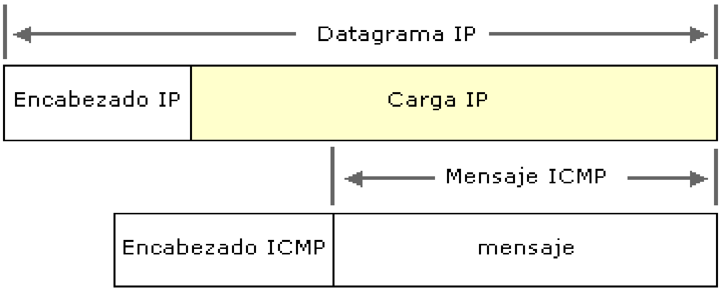


Figura 2.4 Encapsulamiento del mensaje ICMP

En el encabezado ICMP se identifican diferentes tipos de mensajes ICMP. Como los mensajes ICMP se transmiten en datagramas IP, no son confiables.

2.5.4 Protocolo de Administración de Grupos de Internet IGMP

El Protocolo de administración de grupos de Internet IGMP es un protocolo de red utilizado para intercambiar y actualizar información acerca del estado de pertenencia entre enrutadores IP que admiten multidifusión y miembros de grupos de multidifusión. Las estaciones de trabajo miembros individuales informan acerca de la pertenencia de estaciones al grupo de multidifusión y los enrutadores de multidifusión sondean periódicamente el estado de pertenencia.

Multidifusión IP

El tráfico de multidifusión IP se envía a una única dirección, pero se procesa en múltiples hosts. La multidifusión IP es similar a la suscripción a un boletín. Al igual que sólo los suscriptores reciben el boletín cuando se publica, sólo los equipos host que pertenecen al grupo de multidifusión reciben y procesan el tráfico IP enviado a la dirección IP reservada del grupo. El conjunto de hosts que atienden en una dirección de multidifusión IP específica se denomina grupo de multidifusión.

Otros aspectos importantes de la multidifusión IP son los siguientes:

- La pertenencia a grupos es dinámica, lo que permite a los hosts unirse al grupo o abandonarlo en cualquier momento.

- La capacidad de los hosts de unirse a grupos de multidifusión se realiza mediante el envío de mensajes IGMP.
- Los grupos no tienen límite de tamaño y los miembros pueden estar repartidos en diversas redes IP (si los enrutadores de conexión admiten la propagación del tráfico de multidifusión IP y la información de pertenencia a grupos).
- Un host puede enviar tráfico IP a la dirección IP del grupo aunque no pertenezca al grupo correspondiente.

2.5.5 Protocolo de Datagrama de Usuario UDP

El Protocolo de datagramas de usuario UDP es un protocolo de nivel de transporte basado en el intercambio de datagramas. Permite el envío de datagramas a través de la red sin que se haya establecido previamente una conexión, por lo que no establece un diálogo previo entre las dos partes, ya que el propio datagrama incorpora suficiente información de direccionamiento en su cabecera. Tampoco tiene confirmación, ni control de flujo, por lo que los paquetes pueden adelantarse unos a otros; y tampoco se sabe si ha llegado correctamente, ya que no hay confirmación de entrega o de recepción, ofreciendo en su lugar una manera directa de enviar y recibir datagramas a través de una red IP. Se utiliza sobre todo cuando la velocidad es un factor importante en la transmisión de la información, por ejemplo, RealAudio utiliza el UDP. Entonces una estación de trabajo origen que necesita comunicación confiable debe utilizar TCP o un programa que proporcione sus propios servicios de secuencia y confirmación.

Los mensajes UDP están encapsulados y se envían en datagramas IP, como se muestra en la siguiente ilustración de la figura 2.5.

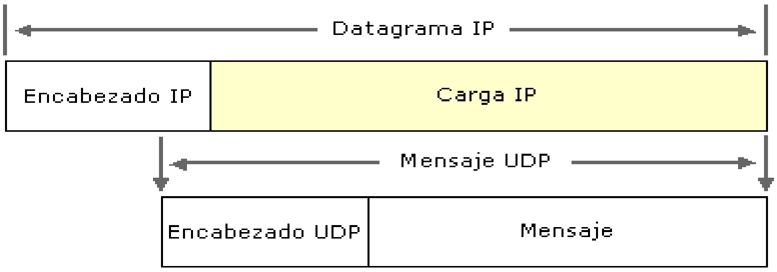


Figura 2.5 Encapsulamiento del mensaje UDP

2.5.6 Protocolo de Control de Transporte TCP

El Protocolo de Control de Transmisión TCP es uno de los protocolos fundamentales en Internet. En el nivel de aplicación, posibilita la administración de datos que vienen del nivel más bajo del modelo, o van hacia él. Este protocolo está orientado a conexión y es fiable a nivel de transporte. Con el uso de TCP, las aplicaciones pueden comunicarse en forma segura, gracias al sistema de acuse de recibo del protocolo TCP, independiente de las capas inferiores. Esto significa que los ruteadores que funcionan en la capa de internet, sólo tienen que enviar los datos en forma de datagramas, sin preocuparse con el monitoreo de datos porque esta función la cumple la capa de transporte o siendo más específico la del protocolo TCP.

Durante una comunicación usando el protocolo TCP, las dos máquinas deben establecer una conexión. La máquina emisora, la que solicita la conexión, se llama cliente, y la máquina receptora se llama servidor. Por eso es que decimos que estamos en un entorno Cliente-Servidor.

Las máquinas de dicho entorno se comunican en modo full-duplex, es decir, que la comunicación se realiza en ambas direcciones. Para posibilitar la comunicación y que funcionen bien todos los controles que la acompañan, los datos se agrupan; es decir, que se agrega un encabezado a los paquetes de datos que permitirán sincronizar las transmisiones y garantizar su recepción. Otra función del TCP es la capacidad de controlar la velocidad de los datos usando su capacidad para emitir mensajes de tamaño variable. Estos mensajes se llaman segmentos. También dicho protocolo proporciona un mecanismo para distinguir distintas aplicaciones dentro de una misma máquina, a través del concepto de puerto. Además esto da soporte a muchas de las aplicaciones más populares de Internet, incluidas HTTP, SMTP, SSH y FTP.

2.5.7 Protocolo de Configuración Dinámica de Estaciones de Trabajo DHCP

El Protocolo de Configuración Dinámica de Estaciones de trabajo DHCP, es un protocolo de red que permite a los nodos de una red IP obtener sus parámetros de configuración automáticamente. Se trata de un protocolo de tipo cliente/servidor en el que generalmente un servidor posee una lista de direcciones IP dinámicas y las va asignando a los clientes conforme éstas van estando libres, sabiendo en todo momento quién ha estado en posesión de esa IP, cuánto tiempo la ha tenido y a quien se la ha asignado después.

DHCP se deriva de del protocolo Bootstrap *BootP*. BootP fue de los primeros métodos para asignar de forma dinámica, direcciones IP a otros equipos como ordenadores, impresoras, etc. Al ser las redes cada vez más grandes, BootP ya no era tan adecuado y DHCP fue creado para cubrir las nuevas demandas. Una configuración básica que puede ser enviada junto con la dirección IP es:

- Dirección IP y la máscara.
- La puerta de enlace para la máquina que quiere acceder a la red.
- Servidor DNS para que la estación de trabajo pueda resolver nombres a direcciones IP.

2.5.8 Protocolo Simple de Administración de Red SNMP

El Protocolo Simple de Administración de Red SNMP es un protocolo de la capa de aplicación que facilita el intercambio de información de administración entre dispositivos de red. Este protocolo permite a los administradores supervisar el desempeño de la red, buscar y resolver sus problemas, y planear su crecimiento.

Un agente *proxy* es un conversor de protocolo que traduce las órdenes SNMP a las comprensibles por el protocolo de gestión propio del dispositivo. Actualmente SNMP está soportado en muchos sistemas distintos tales como puentes, PC's, estaciones de trabajo, ruteadores, terminales, servidores, hubs, concentradores, y tarjetas avanzadas ethernet, token ring y FDDI. SNMP se basa en un sistema de petición-respuesta. La autoridad gestora no es la red como sistema sino una o varias estaciones distinguidas.

2.5.9 Protocolo de Transferencia de Hipertexto HTTP

El Protocolo de Transferencia de HiperTexto HTTP define la sintaxis y la semántica que utilizan los elementos software de la arquitectura web para comunicarse. Es un protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor. Al cliente que efectúa la petición se lo conoce como “agente del usuario”. A la información transmitida se la llama recurso y se la identifica mediante un URL. Los recursos pueden ser archivos, el resultado de la ejecución de un programa, una consulta a una base de datos, la traducción automática de un documento, entre otras.

HTTP es un protocolo sin estado, es decir, que no guarda ninguna información sobre conexiones anteriores. El desarrollo de aplicaciones web necesita frecuentemente mantener estado. Para esto se usan las cookies, que es información que un servidor puede almacenar en el sistema cliente. Esto le permite a las aplicaciones web instituir la noción de "sesión", y también permite rastrear usuarios ya que las cookies pueden guardarse en el cliente por tiempo indeterminado.

2.5.10 Protocolo de Transferencia de Archivo FTP

Los objetivos del Protocolo de Transferencia de Archivos FTP son: promover el compartimiento de archivos ya sea de programas de computadoras o datos, fomentar indirectamente o implícitamente vía programas el uso de computadoras en forma remota, proteger a un usuario de variaciones en el sistema de almacenamiento de archivos entre ordenadores y finalmente para transferir confiablemente y eficientemente datos.

Como es necesario hacer un registro de usuario en el host remoto, el usuario debe tener un nombre de usuario y una clave para acceder a ficheros y a directorios. El usuario que inicia la conexión asume la función de cliente, mientras que el host remoto adopta la función de servidor

En ambos extremos del enlace, la aplicación FTP se construye con intérprete de protocolo, un proceso de transferencia de datos, y una interfaz de usuario como muestra la figura 2.6.

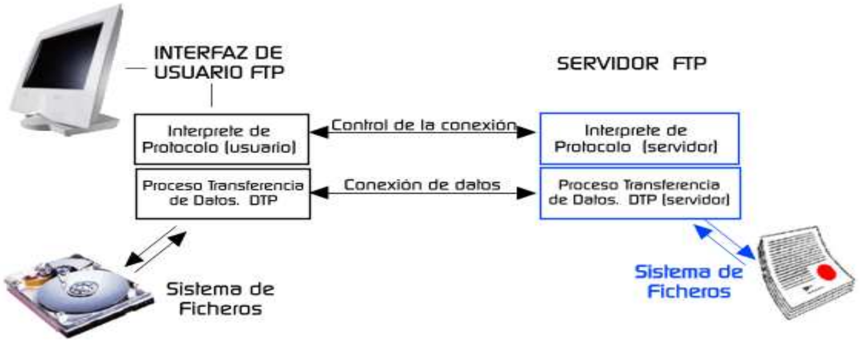


Figura 2.6 Interprete de Protocolo.

La interfaz de usuario se comunica con el intérprete de protocolo que está a cargo del control de la conexión. Este intérprete de protocolo ha de comunicar la información necesaria a su propio sistema de archivos.

2.5.11 Protocolo de Transferencia de Archivos Trivial TFTP

TFTP, es un protocolo simple para transferir archivos, y por lo tanto, fue nombrado como El Protocolo de Transferencia de Archivo Trivial o TFTP. Esto ha sido implementado encima del Protocolo de Datagramas de Usuario, así esto puede ser usado para mover archivos entre máquinas o computadoras en diferentes redes que implementen UDP. Esto no debe excluir la posibilidad de implementar TFTP encima de otros protocolos de datagrama. Esto está diseñado para ser pequeño y fácil de implementar. Por consiguiente; esto carece de alguna de las características de un FTP regular. La única cosa que esto puede hacer es leer y escribir archivos o correos de un servidor remoto. Este protocolo no puede listar directorios, y actualmente no tiene disposición para usar autenticación.

Cualquier transferencia comienza con una solicitud para leer o escribir un fichero. Si el servidor concede la solicitud, se abre la conexión y el fichero se envía en bloques consecutivos de 512 bytes(longitud fija). Los bloques del fichero se numeran correlativamente, comenzando en 1. Cada paquete de datos debe ser reconocido mediante un paquete de reconocimiento antes de que se envíe el siguiente paquete. Se asume la terminación de la transferencia al recibir un paquete de menos de 512 bytes.

La mayoría de los errores provocarán la terminación de la conexión, si un paquete se pierde en la red, se producirá un tiempo fuera o de espera, tras el que se efectuará la retransmisión del último paquete de datos o de reconocimiento.

CAPÍTULO 3

3. EL MICROCONTROLADOR PIC18F97J60

Hay muchos fabricantes de microcontroladores, los principales son: ST Microelectronics, Texas Instruments, Infineon Technologies, Intel Corp, Royal Philips Electronics, Analog Devices, National Semiconductors, Motorola Semiconductors, Toshiba, Microchip, NEC Corp., Mitsubishi, Hitachi Corp y Atmel Corporation. Todos tienen sus ventajas e inconvenientes y se deben valorar factores muy dispares desde prestaciones, relación calidad/precio, disponibilidad comercial y tipo de herramientas disponibles.

La familia del Microcontrolador **PIC18F97J60** está diseñada especialmente para el uso de Ethernet, en él se **ha integrado el controlador y el periférico de Ethernet 10BaseT (IEEE 802.3)**. La familia PIC18F97J60 consiste en nueve microcontroladores que se optimizan para los usos, además tiene un controlador de acceso al medio (MAC) y dispositivo para la capa física (PHY).

El módulo de Ethernet se puede utilizar para conectar los dispositivos de una LAN y a su vez con Internet. Agregando conectividad de

Ethernet, los microcontroladores pueden distribuir datos sobre Internet y se pueden supervisar y ser remotamente controlados.

Los nueve miembros de la familia de los microcontroladores PIC18F97J60 son: **PIC18F66J60, PIC18F66J65, PIC18F66J60, PIC18F86J60, PIC18F86J65, PIC18F87J60, PIC18F96J60, PIC18F96J65, y PIC18F97J60**

La siguiente tabla muestra las características más importantes y dominantes de los microcontroladores PIC18F97J60:

Dispositivo	Memoria Flash del programa (octetos)	Memoria de los datos SRAM (octetos)	Almacenador intermediario de Ethernet TX/RX (octetos)	I/O	Convertidor Analógico a Digital de 10-bit	CCP/ ECCP
PIC18F66J60	64K	3908	8192	39	11	2/3
PIC18F66J65	96K	3908	8192	39	11	2/3
PIC18F66J60	128K	3908	8192	39	11	2/3
PIC18F86J60	64K	3908	8192	55	15	2/3
PIC18F86J65	96K	3908	8192	55	15	2/3
PIC18F87J60	128K	3908	8192	55	15	2/3
PIC18F96J60	64K	3908	8192	70	16	2/3
PIC18F96J65	96K	3908	8192	70	16	2/3
PIC18F97J60	128K	3908	8192	70	16	2/3

Dispositivo	MSSP			EUSART	Comparadores	Contadores de tiempo Real 8/16-Bit	PSP	Bus Externo De Memoria
		SPI	Maestro I2CTM					
PIC18F66J60	1	Y	Y	1	2	2/3	N	N
PIC18F66J65	1	Y	Y	1	2	2/3	N	N
PIC18F66J60	1	Y	Y	1	2	2/3	N	N
PIC18F86J60	1	Y	Y	2	2	2/3	N	N
PIC18F86J65	1	Y	Y	2	2	2/3	N	N
PIC18F87J60	1	Y	Y	2	2	2/3	N	N
PIC18F96J60	2	Y	Y	2	2	2/3	Y	Y
PIC18F96J65	2	Y	Y	2	2	2/3	Y	Y
PIC18F97J60	2	Y	Y	2	2	2/3	Y	Y

Tabla 3.1 Características principales de los PICs 18F9XJ6X

Ejemplos del uso de los microcontroladores PIC18F97J60:

- Automatización Industrial
    - Control Industrial
    - Supervisión de Fuentes de Energía
    - Supervisión de Redes y servidores
    - Control del medio ambiente
- Automatización De Viviendas

  - Fuego y Seguridad
  - Control de Acceso
  - Paneles de Seguridad
  - Control de la Iluminación
  - Intercomunicador De VoIP
- Control Comercial
  - Aplicaciones de Cocina
  - Dispensadores de Bebida
  - Minibares de Hoteles
  - Terminales de Posición
- Control Casero
  - Seguridad
  - Redes
  - Aplicaciones

3.1 Introducción a Microcontroladores de la Familia PIC18

La familia de microcontroladores PIC 18 gozan de características que le dan un alto rendimiento.

Entre sus principales características encontramos: que poseen convertidores análogo-digitales incorporados y una avanzada arquitectura RISC, que permite trabaja con dispositivos Flash y OTP.

En comparación a otros microcontroladores, los de la familia PIC 18 han aumentado características en su núcleo, poseen una pila de 32 niveles de memoria y un amplio número de interrupciones internas y externas. Su avanzada arquitectura permite instrucciones de palabras de 16 bits, opciones de empaquetado físico de 18 a 100 pines. Se cuenta con 79 instrucciones que son ejecutadas en dos pasos en un solo ciclo.

Debido a todos sus beneficios y elementos incorporados reduce el uso de elementos externos, reduciendo así también el consumo de energía.

Características de Alto rendimiento

- Memoria de Programa Flash Flexible
- Bajo consumo de energía

- Memoria lineal de programa con capacidad de hasta 2 Mbytes
- Rendimiento de operación de 10 Millones de Instrucciones por segundo (MIPS)
- Multiplicador 8 x 8.
- Avanzada comunicación con periféricos y protocolos (CAN, USB y TCP/IP).

3.1.1 Descripción del Microcontrolador PIC18F97J60

El **PIC18F97J60** está optimizado para aplicaciones de Ethernet embebidas y tiene en el propio chip el control de acceso al medio (MAC) y la capa física (PHY).

Integrando un controlador de Ethernet 10BASE-T en un Microcontrolador **PIC18** de 10 MIPS con hasta 128 Kbytes de memoria de programa flash, nos proporciona a los usuarios una solución de comunicación remota, simple y con un coste efectivo para un amplio rango de aplicaciones.

Ethernet es la tecnología líder en redes de área local (LANs) y esto puede ser usado para conectar dispositivos embebidos a Internet a través de una red de área local. La infraestructura, características, interoperabilidad, posibilidad de ampliación y fácil de desarrollar hacen de Ethernet una elección estándar para comunicaciones embebidas.

Cualquier aplicación que requiera conectividad Ethernet puede obtener ventajas del Microcontrolador **PIC18F97J60**, como aplicaciones industriales, automatización de edificios, control comercial, control vivienda inteligente, etc.

Podemos citar sus principales características de diseño:

- Tecnología nanoWatt

  Incorpora un gran rango de características, que pueden significativamente reducir el consumo de energía durante su operación, entre los puntos clave para ello tenemos:

  - Alterna modos de funcionamiento: Podemos utilizar un oscilador externo o el oscilador interno del microcontrolador que reduce a un 90% el consumo de energía.

  - Múltiples Modos de inactividad: El controlador puede funcionar con la unidad central de proceso de su interior desactivada pero los periféricos conectados a él aún funcionando. Esto nos ahorrará un 4% de la energía que normalmente se consume en su normal funcionamiento.

  - Modo de conmutación por software: Los diferentes modos de consumo de energía son invocados mediante el código del usuario durante la operación, permitiendo así al usuario incorporar ideas de ahorro de consumo de energía en el diseño de su programa.

- Diferentes tipos de fuentes de oscilación

  El Microcontrolador nos ofrece cinco diferentes opciones de osciladores, permitiéndonos elegir la más conveniente para nuestra aplicación.

  Las opciones que nos ofrece son las siguientes:

  - Dos opciones de oscilador de cristal: Usando cristales o resonadores de cerámica.
  - Dos modos de relojes externos: Ofreciendo la opción de dividir para cuatro una salida de reloj.
  - Un multiplicador de frecuencia por Lazo de Enganche de Fase (PLL), disponible para modos de oscilación externa que permite velocidades de hasta 41.67 MHz.

    - Un oscilador interno RC, con una salida fija de 31 KHz que provee un consumo bajo de energía, ideal para aplicaciones no sensibles a sincronismo.
    - El bloque de oscilador interno, que provee una fuente de referencia estable para robustecer su operación.

- Memoria Expandida

  Nos provee un amplio espacio para el código de nuestras aplicaciones, desde 64 Kbytes a 128 Kbytes. Las celdas Flash para la memoria de programa nos permiten hasta 100 ciclos de escritura y lectura. La retención de datos en su memoria está estimada para veinte años.

  Además tenemos un espacio para aplicaciones dinámicas con capacidad de 3808 bytes de datos en su memoria RAM.

- Bus de Memoria Externa

  En caso de requerir mayor capacidad de memoria, el PIC18F97J60 implementa un bus de memoria externa. Esto permite al controlador interno de contador de programa direccionar un espacio de memoria de hasta 2 Mbytes. Podemos adicionar memoria de las siguientes maneras:

  - Usando combinaciones de memorias externas de hasta 2 Mbytes.
  - Usando una memoria Flash externa para aplicaciones reprogramables de código o largas tablas de datos.
  - Usando dispositivos con memoria RAM para almacenamiento de datos.
- Conjunto Extendido de Instrucciones

  El PIC18F97J60 implementa la extensión opcional del grupo de instrucciones normalmente utilizados en la familia PIC18, añadiendo ocho nuevas instrucciones y un indicador de modo de direccionamiento.

  Habilitada la opción de configuración de dispositivo, la extensión ha sido específicamente diseñada para optimizar el código de aplicaciones re-entrantes originalmente desarrolladas en lenguajes de alto nivel, como C.

- Comunicación

  El PIC18F97J60 incorpora un rango de periféricos de comunicación serial, incluyendo dos transceptores universales sincrónicos asincrónicos (USART) independientes y mejorados y dos módulos SSP Maestros. Además, uno de los propósitos generales de los puertos de entrada/salida es que pueden ser reconfigurados como un puerto esclavo paralelo de 8 bits para comunicaciones directas de procesador a procesador.

- Módulos CCP

  Incorpora dos módulos de captura, comparación y modulación por ancho de pulsos (CCP) y tres módulos CCP mejorados (ECCP) para maximizar la flexibilidad en aplicaciones de control. Pueden ser usadas cuatro diferentes bases de tiempo para llevar a cabo varias operaciones al mismo tiempo.

  Cada uno de los tres módulos ECCP ofrecen cuatro salidas PWM, totalizando un número de 12 salidas PWM

- Convertidores A/D de 10-Bits

  Este módulo incorpora tiempo de adquisición programable, permitiendo a un canal seleccionar una conversión a ser iniciada sin esperar por un periodo de muestreo, reduciendo código.

- Temporizador de vigía extendido (WDT)

Esta ampliada versión incorpora un pre-escalador de 16 bits, permitiendo un mayor tiempo de espera que es estable, operando voltaje y temperatura.

El dispositivo seleccionado por nosotros viene en un empaquetado de 100 pines que se muestra a continuación con su respectivo diagrama de bloques, correspondiente al funcionamiento antes mencionado:

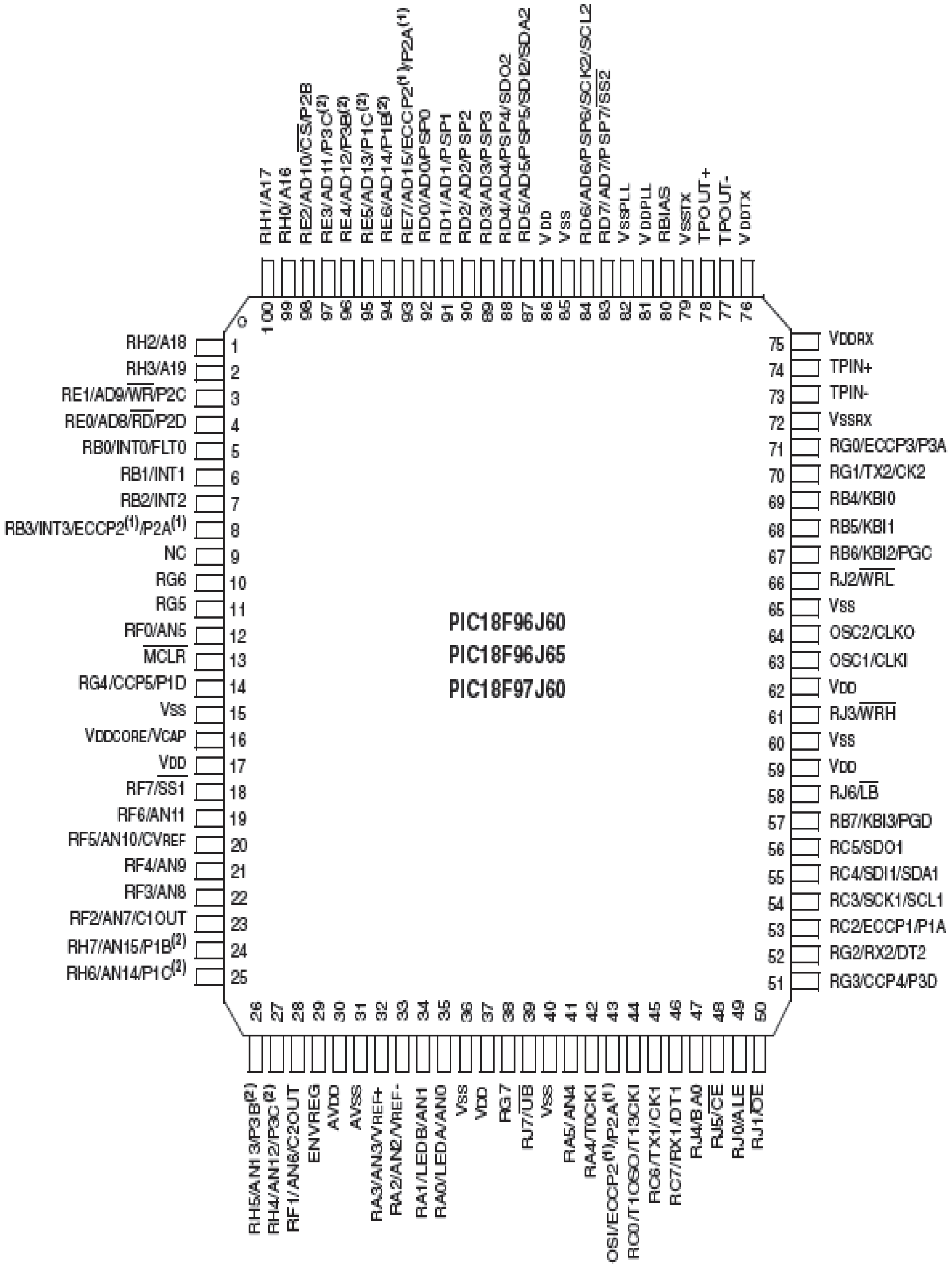


Figura 3.1 Distribución de pines del PIC18F97J60

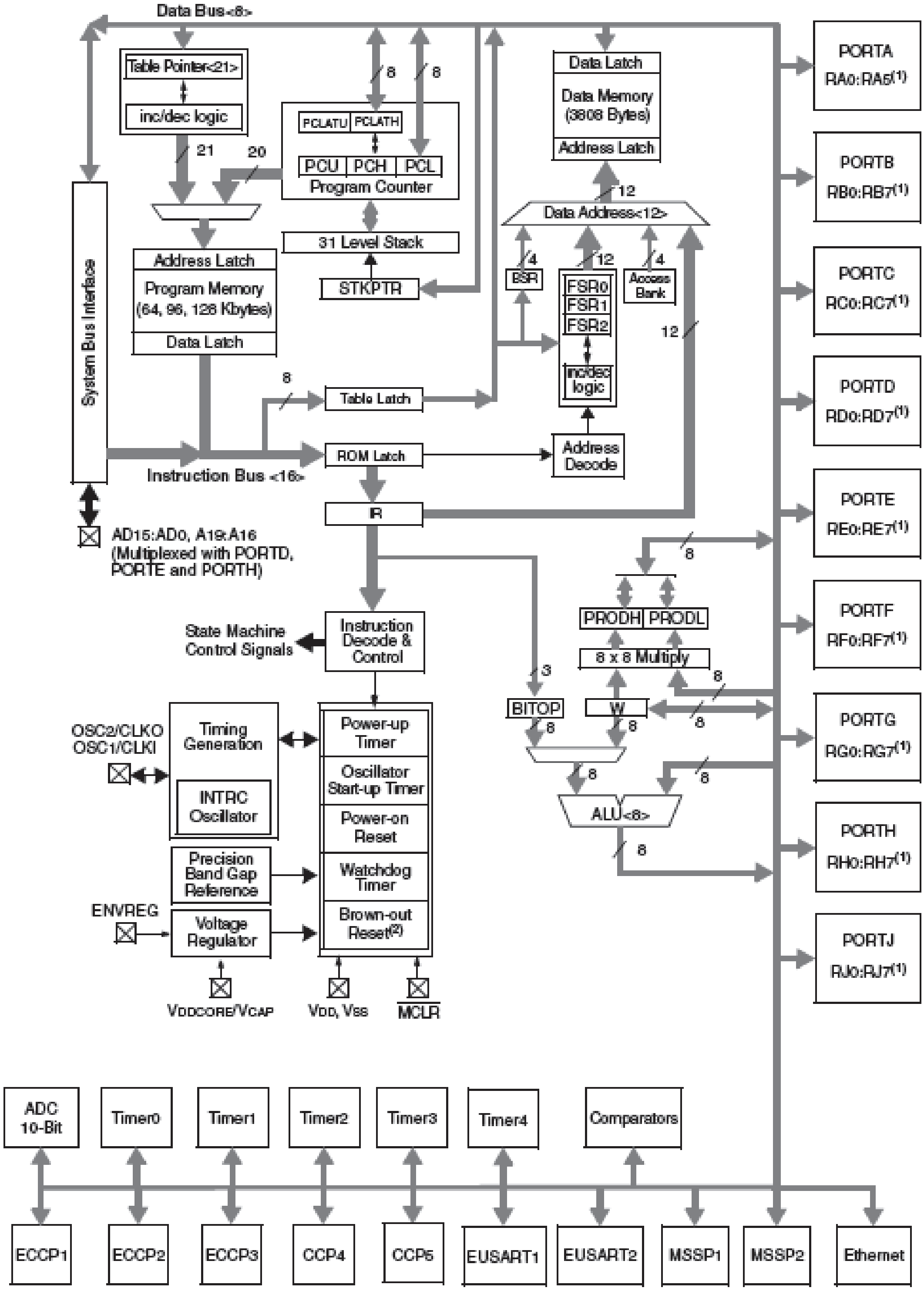


Figura 3.2 Diagrama de bloques del PIC 18F97J60

3.1.2 Utilización del Microcontrolador PIC18F97J60

El Microcontrolador PIC18F97J60 nos ofrece una serie de módulos embebidos que nos permiten desarrollar diferentes aplicaciones; hablando de nuestra aplicación podemos citar los módulos utilizados y su respectiva función en el diseño final.

El módulo Ethernet, es uno de los componentes principales del Microcontrolador y de nuestro proyecto, incluye implementaciones completas de módulos de Control de Acceso al Medio (MAC) y de la capa física (PHY). Lo único que requeriremos para la conexión a la red serán dos transformadores y un conjunto de elementos pasivos que se detallarán en el capítulo siguiente. Este módulo reconoce todas las especificaciones para la conectividad: 10 Base T en una red de par trenzado, incorpora un número de esquemas de filtros para limitar el tráfico de paquetes y también provee un módulo interno DMA, para proceso de datos y cálculos de verificación de direcciones IP

Para verificar su actividad el módulo, Ethernet incorpora dos señales de LEDS que actúan como salidas para demostrar su actividad y enlace respectivamente.

Podemos subdividir el módulo Ethernet en cinco bloques de funcionalidad específica que son:

1. El bloque de transmisión y recepción de la capa física, que codifica y decodifica los datos análogos presentes en la interfaz de par trenzado, y envía o recibe estos datos a través de la red.

2. El bloque MAC que implementa la lógica del estándar IEEE 802.3 y provee de control independiente de interface para controlar la capa física PHY.

3. Un almacenador RAM de 8 Kbytes, para los paquetes que han sido recibidos y que serán transmitidos.

4. Un controlador de acceso al almacenador RAM, cuando los requerimientos son hechos desde el núcleo del microcontrolador, DMA.

5. La interface de registro que funciona como un interpretador de comandos y señales internas de estado entre el módulo Ethernet y el Registro de Funciones Especiales, del microcontrolador.

Otro de los módulos importantes de nuestro PIC es el EUSART, que es uno de los dos módulos de entradas y salidas seriales;

este puede ser configurado como un sistema full-duplex asincrónico que puede comunicarse con diferentes dispositivos periféricos. También puede ser configurado como un sistema half-duplex sincrónico que así mismo se comunicará con diferentes dispositivos periféricos como EEPROMs, circuitos integrados análogos - digitales, etc.

Este modulo implementa características adicionales, incluyendo una detección y calibración automática de la tasa de baudios, recepción automática de alarmas en quiebres de sincronismo, y transmisión de caracteres de 12 bits.

El PIC18F97J60 posee dos módulos EUSART; EUSART1 y EUSART2, y su operación es controlada a través de tres registros:

- Registro de Estado de transmisión y control (TXSTAx).
- Registro de Estado de recepción y control (RCSTAx).
- Control de tasa de baudios (BAUDCONx).

El modulo de Puerto Sincrónico Serial Maestro (MSSP) es otro de los módulos importantes utilizados en nuestro proyecto, es una interface serial, utilizada para comunicarnos con otros periféricos o microcontroladores. El módulo MSSP puede operar de dos maneras:

- Interface Serial Periférica (SPI).
- Circuito Inter-Integrado (I2C).

A su vez la interface I2C se puede manejar de las siguientes maneras:

- Modo Maestro.
- Modo Multi Maestro.
- Modo esclavo.

El modo SPI permite 8 bits de datos para ser sincrónicamente transmitidos y recibidos a la vez. Para llevar a cabo la comunicación, son utilizados tres pines:

- Salida de Dato Serial (SDOx): RD4/SDO2.
- Entrada de Dato Serial (SDIx): RD5/SDI2/SDA2.
- Reloj Serial (SCKx): RD6/SCK2/SCL2.

Adicionalmente puede ser utilizado un cuarto pin en el modo de operación esclavo:

- Seleccionar esclavo (SSX): RD7/SS2.

Nuestro Microcontrolador posee dos módulos MSSP: MSSP1 y MSSP2 que operan independientemente uno del otro.

Cada módulo MSSP tiene asociado dos registros de control SSPxCON1 y SSPxCON2, y uno de estado (SSPxSTAT), cuyo uso depende del modo en que estemos trabajando el módulo.

Otro de los módulos importantes utilizados en nuestro proyecto es el Temporizador TIMER0, que incorpora las siguientes características:

- Puede ser configurado por software como temporizador o como contador de 8 o 16 bits.
- Posee registros fácil de leer y programar.
- Posee un pre escalador de 8 bits programable.
- Da la opción de trabajar con una fuente de reloj interna o externa.

- Se puede seleccionar un margen de oscilación para el reloj externo.
- Tiene una interrupción de sobreflujo.

El registro que controla la operación del TIMER0 es el T0CON.

Cabe recalcar que el microcontrolador nos da la facilidad de trabajar con otros cuatro temporizadores que para este proyecto no los hemos utilizado.

Para sincronizar el funcionamiento de todas nuestras señales utilizadas, el PIC18F97J60 permite seleccionar diferentes fuentes de oscilación que esencialmente las podemos dividir en tres:

- Osciladores Primarios.
- Osciladores Secundarios.
- Bloque interno de oscilación.

Nos hemos ayudado también con las interrupciones que nos provee nuestro Microcontrolador, específicamente con la interrupción INT0; contamos con trece registros que controlan las 29 interrupciones que son:

- RCON.
- INTCON
- INTCON2
- INTCON3
- PIR1, PIR2, PIR3
- PIE1, PIE2, PIE3
- IPR1, IPR2, IPR3

Para la utilización de las diferentes señales de nuestro proyecto, el PIC cuenta con 70 entradas y salidas repartidas en 9 puertos.

La programación serial se la llevará a cabo mediante de los pines de entrada y salida RB6 y RB7, donde RB6 llevará la señal de oscilación y RB7 los datos que se programarán en el PIC.

3.2 El Compilador MPLAB

MPLAB IDE, es un programa que funciona en un computador personal para desarrollar aplicaciones de microcontroladores Microchip.

Es llamado Ambiente Integrado de Desarrollo (IDE), porque provee un fácil e integrado “ambiente” que contiene todos los componentes necesarios para desarrollar y desplegar aplicaciones para sistemas embebidos.

Un sistema impregnado es típicamente un diseño interno, que utiliza los recursos y energía, en este caso de un Microcontrolador. Estos microcontroladores combinan una unidad microprocesador con circuitos adicionales llamados periféricos, más algunos circuitos adicionales en el mismo chip para hacer un pequeño módulo de control requiriendo otros dispositivos externos. Este simple dispositivo puede ser a su vez introducido en otros dispositivos electrónicos y mecánicos.

Un desarrollador de un sistema para controladores embebidos es un conjunto de programas corriendo en una computadora de escritorio que ayuda a escribir, editar y depurar el código del programa en un Microcontrolador.

MPLAB IDE, funciona en un computador y contiene todos los componentes necesarios para desarrollar y desplegar aplicaciones para sistemas embebidos.

Las típicas tareas para desarrollar una aplicación de sistemas embebidos son:

1. Crear el diseño de alto nivel: Desde las características y funcionamiento deseado, eligiendo el Microcontrolador indicado, luego diseñar los circuitos necesarios. Después de determinar que periféricos y pines de control tendrá el sistema, escribir el programa que controlará los aspectos del hardware para la aplicación requerida.

   Es necesaria una herramienta de lenguaje como un ensamblador que traduce directamente a código de máquina o un compilador que permite una forma más natural de lenguaje para crear programas. Los ensambladores y compiladores ayudan a hacer el código más entendible con facilidades como etiquetas para identificar rutinas de programa, variables que tengan nombres asociadas con su uso y con editores que ayudan a organizar el programa en estructuras manejables.

2. Compilar, ensamblar y enlazar el programa usando el ensamblador o compilador, para convertir nuestro código en binario.

3. Probar el código: Usualmente un complejo programa no trabaja exactamente de la forma que se imagina, y ciertos errores necesitan ser eliminados, para que el diseño de los resultados esperados.

   El depurador nos permite ver la ejecución de programa a nivel de máquina, relacionando nuestro código con los símbolos y funciones del programa. Además el depurador nos permite experimentar cambiando valores de variables, rutinas, etc.

4. Grabar el código en un microcontrolador y verificar que funcione correctamente la aplicación terminada.

El editor de programa de MPLAB IDE nos ayuda a escribir correctamente el código con las herramientas de lenguaje seleccionadas. El editor automáticamente colorea las palabras claves en nuestro programa para asegurar que su escritura sea la adecuada.

El manejador de proyectos permite organizar todos los archivos utilizados en la aplicación.

Cuando el código está construido se puede controlar rigurosamente la manera en que será optimizado para la capacidad, y velocidad del compilador, el lugar donde serán programadas las variables y datos en el dispositivo.

MPLAB tiene componentes llamados depuradores y simuladores, gratuitos para todos los microcontroladores Microchip que ayudan a probar el código.

Si el sistema físico no está terminado aún, podemos empezar probando el código con el simulador, un programa que simula la ejecución de el Microcontrolador.

El simulador puede aceptar una entrada simulada, para verificar como el programa de control responde a señales externas. También puede medir el tiempo de ejecución del código, revisar paso a paso la ejecución del programa y generar una lista con detalles de la ejecución.

El proceso para escribir una aplicación es a menudo descrito como un ciclo de desarrollo, como tal es imposible que se puedan realizar todos los pasos de implementación de una sola vez.

En la mayoría de los casos el código es escrito, probado y luego modificado con el objeto de producir una aplicación que trabaje de acuerdo a las necesidades de los usuarios.

El siguiente es un esquema del ciclo de desarrollo, utilizado para la implementación de sistemas embebidos con MPLAB IDE.

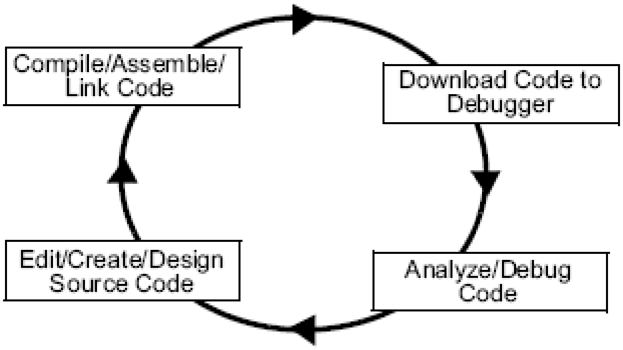


Figura 3.3 Ciclo de desarrollo para implementación de sistemas embebidos en MPLAB IDE

El ambiente de desarrollo integrado coordina todas las herramientas desde una simple interfaz gráfica. Una vez que el código es escrito puede ser convertido a instrucciones ejecutables y grabados en un Microcontrolador para observar como trabaja.

En este proceso son necesarias múltiples herramientas: un editor, para escribir el código; un manejador de proyecto para organizar archivos y configuraciones; un compilador o ensamblador para convertir el código fuente a código de máquina; y algunos tipos de hardware o programas que se conectan a un microcontrolador o simulan la operación del mismo.

El manejador de proyectos organiza los archivos que van a ser editados y otros asociados, ellos pueden ser enviados hacia las herramientas de lenguaje para ensamblarse, compilarse y por último enlazarse.

El enlace tiene la tarea de ubicar los fragmentos de código objeto desde el ensamblador, compilador y librerías; en sus propias áreas de memoria del controlador embebido y asegurarse de la función de los módulos entre sí.

Esta operación, desde el ensamblaje y compilación a través del proceso de enlace es llamada una construcción de proyecto.

Desde el manejador de proyectos, las propiedades de las herramientas de lenguaje pueden ser invocadas diferentemente por cada archivo y construir un proceso que integra todas las operaciones de las herramientas de lenguaje.

Los archivos fuente son archivos de texto que son escritos de acuerdo a las reglas del ensamblador o compilador. El ensamblador y el compilador convierten a estos en módulos intermedios de lenguaje de máquina; ubicándolos para referencia de funciones y almacenamiento de datos.

El enlazador resuelve esta ubicación y combina todos los módulos en un archivo ejecutable de código de máquina. También produce un archivo de depuración que permite a MPLAB IDE relacionar el código de máquina en ejecución con los archivos fuente.

Un editor de texto es usado para escribir el código, no es un editor de texto normal sino un editor diseñado específicamente para la escritura de código en microcontroladores Microchip.

El editor soporta operaciones comúnmente usadas en escritura de código como: encontrar corchetes y llaves en C, hacer comentarios de bloques de código, encontrar texto en múltiples archivos, etc.

Después que el código es escrito, el editor trabaja con otras herramientas para mostrar el código de ejecución en el depurador.

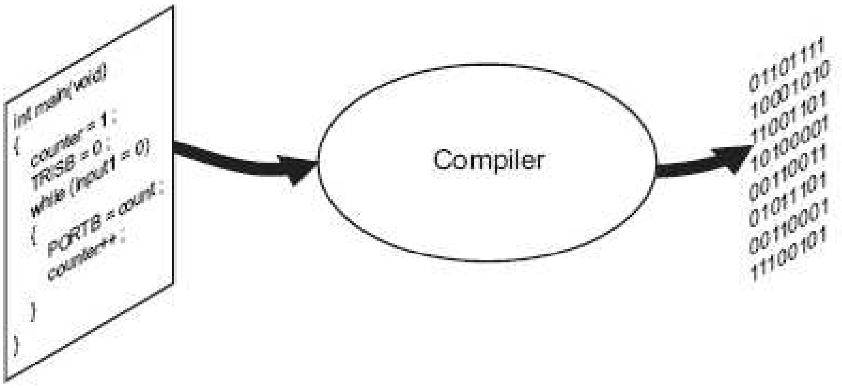
Los puntos de quiebre pueden ser situados en el editor, y los valores de las variables pueden ser inspeccionadas con el puntero del ratón sobre la misma.

Las herramientas de lenguaje difieren de los compiladores típicos, debido a que producen código para funcionar en otro microprocesador.

Las herramientas de lenguaje también producen un archivo de depuración que MPLAB IDE usa para correlacionar las instrucciones de máquina y localidades de memoria con el código fuente.

Este bit de integración permite al editor de MPLAB situar puntos de quiebre y observar ventanas para ver contenidos variables.

El siguiente gráfico nos muestra la acción que realiza el compilador, traduciendo el código fuente en código de máquina.



```
int main(void)
{
   counter = 1 ;
   TRISB = 0 ;
   while (input1 = 0)
   {
      PORTB = count ;
      counter++ ;

   }
}
```

```
01101111
10001010
11001101
10100001
10110011
00110011
01011101
00110001
11100101
```

Figura 3.4 Efecto del Compilador

En MPLAB IDE, la ejecución del código es probada en un depurador. El depurador puede ser un programa que simula la operación del

Microcontrolador para pruebas o puede ser un equipo especial, para analizar como el programa ejecuta una determinada aplicación.

Los simuladores están construidos para que un programa pueda ser probado sin ningún hardware adicional. Un simulador, es un programa depurador, y las funciones del depurador para el simulador son idénticas a los depuradores de hardware.

Existen 2 tipos de hardware que pueden ser usados con MPLAB IDE: programadores y depuradores de hardware. Un programador simplemente transfiere el código de máquina desde el computador hacia una memoria interna del Microcontrolador destino.

Un dispositivo puede ser programado con el “depurador en circuito”, o con un programador de dispositivos. MPLAB IDE puede ser ejecutado como programador y así grabar la información requerida en los PICs.

MPLAB IDE tiene tanto componentes internos como componentes externos para configurar una variedad de programas y herramientas para hardware.

Entre los componentes internos tenemos:

- Project Manager

- Editor
- Assembler/Linker and Language Tools
- Debugger
- Execution Engines

Componentes adicionales de MPLAB IDE

- Herramientas Compiladoras de Lenguaje

  MPLAB C17, MPLAB C18 y MPLAB C30: de Microchip proveen un código completamente integrado y optimizado.

  Ellos son invocados por el manejador de proyectos para compilar el código, que es automáticamente cargado en el depurador destino para su posterior prueba y verificación.

- Programadores

  PICSTART Plus, PRO MATE II, MPLAB PM3 y MPLAB ICD 2 son capaces de programar el respectivo código en los microcontroladores.

- Emuladores “En circuito”

  MPLAB ICE 2000 y MPLAB ICE 4000

Estos se conectan al computador por medio de puertos de entrada y salida y permiten un completo control sobre la operación de los microcontroladores en sus diferentes aplicaciones.

- Depurador “En circuito”

MPLAB ICD 2 provee una alternativa económica a un microcontrolador, puede bajar código en él, con su respectiva información.

3.2.1 Herramientas de MPLAB

3.2.1.1 Herramienta C18

MPLAB C18 es un compilador que se ejecuta en un computador y produce código que puede ser ejecutado por microcontroladores de la misma familia. Al igual que un ensamblador, el compilador C18, traduce de código de programación a lenguaje de máquina.

MPLAB C18 toma el estándar de lenguaje de programación C con instrucciones como: “if(x==y)” y “temp=0x27”.

El compilador puede optimizar código utilizando rutinas que fueron empleadas en una función de lenguaje C para ser usadas por otras funciones. Además es capaz de reorganizar el código, eliminar código que nunca será ejecutado, compartir fragmentos de código entre varias funciones, e identificar datos y registros que son usados ineficientemente.

El código es escrito utilizando la notación del estándar ANSI C. El código fuente es compilado en bloques de código y datos que posteriormente son enlazados con otros bloques similares para ser luego ubicados en diferentes regiones de memoria del Microcontrolador.

Este proceso es llamado una “construcción”, y es a menudo ejecutado a medida que el código de programa es escrito, probado y depurado.

El compilador MPLAB C18 y sus herramientas asociadas, como el enlazador y ensamblador, pueden ser llamados desde una línea de comando para construir un archivo con extensión .HEX, que puede ser programado en el dispositivo.

MPLAB C18 y sus demás herramientas pueden ser llamadas dentro de MPLAB IDE.

MPASM es un componente de MPLAB IDE y trabaja en conjunto con MPLINK para enlazar las secciones de lenguaje de código con código en lenguaje C del compilador MPLAB C18.

Las rutinas de lenguaje de ensamblaje son prácticamente para pequeñas secciones de código que necesiten ejecutarse rápido o en un determinado tiempo.

REQUERIMIENTOS DEL SISTEMA

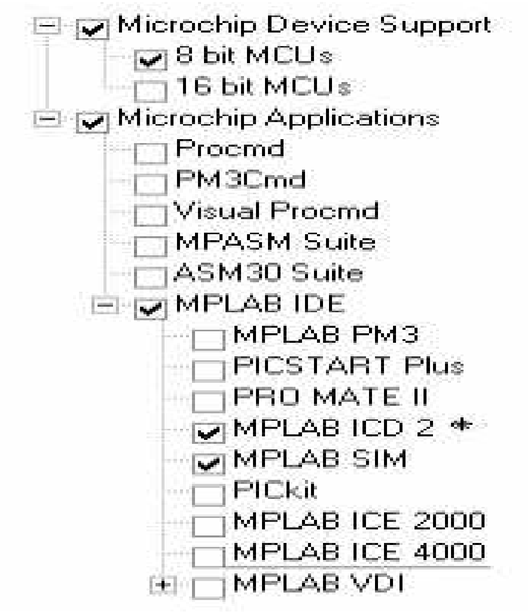
Los requerimientos de sistema sugeridos para el uso de MPLAB C18 y MPLAB IDE son:

- Procesador de la clase Intel® Pentium® con sistema operativo Microsoft® 32-bit Windows (Windows 2000, Windows XP Home or Windows XP Professional).
- Aproximadamente 250 MB de espacio de memoria en el disco duro.
- Herramientas opcionales:
  - PICDEM 2
  - MPLAB ICD 2 (Requiere conexión serial o vía USB)

INSTALACION

Primeramente, se debe haber instalado MPLAB IDE para reinstalar MPLAB C18. Cuando se instala MPLAB IDE para usar con MPLAB C18 se deben seleccionar las siguientes opciones:

- MPLAB IDE Device Support
    - 8-bit MCUs
- Microchip Applications
    - MPLAB IDE
    - MPLAB SIM
    - MPASM Suite (este se instala con MPLAB c18, por ello no necesita ser instalado con MPLAB IDE).



- ☑ Microchip Device Support
    - ☑ 8 bit MCUs
    - ☐ 16 bit MCUs
- ☑ Microchip Applications
    - ☐ Procmd
    - ☐ PM3Cmd
    - ☐ Visual Procmd
    - ☐ MPASM Suite
    - ☐ ASM30 Suite
    - ☑ MPLAB IDE
        - ☐ MPLAB PM3
        - ☐ PICSTART Plus
        - ☐ PRO MATE II
        - ☑ MPLAB ICD 2 *
        - ☑ MPLAB SIM
        - ☐ PICkit
        - ☐ MPLAB ICE 2000
        - ☐ MPLAB ICE 4000
        - ☐ MPLAB VDI

Figura 3.5 Selección de elementos a instalar en MPLAB IDE

DIRECTORIOS

MPLAB C18 puede ser instalado en cualquier directorio del computador, normalmente siempre es instalado en el directorio:

C:\mcc18 folder.

La siguiente figura muestra la estructura del directorio para la instalación de MPLAB C18:

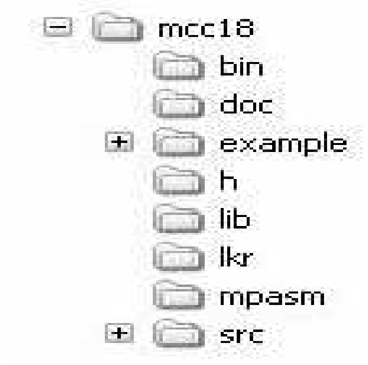


Figura 3.6 Estructura de directorio de instalación de MPLAB IDE

El directorio de instalación de MPLAB C18 contiene el archivo de lectura para el compilador, el ensamblador y el enlazador.

A continuación, mostramos el diagrama de ejecución de las herramientas de lenguaje:

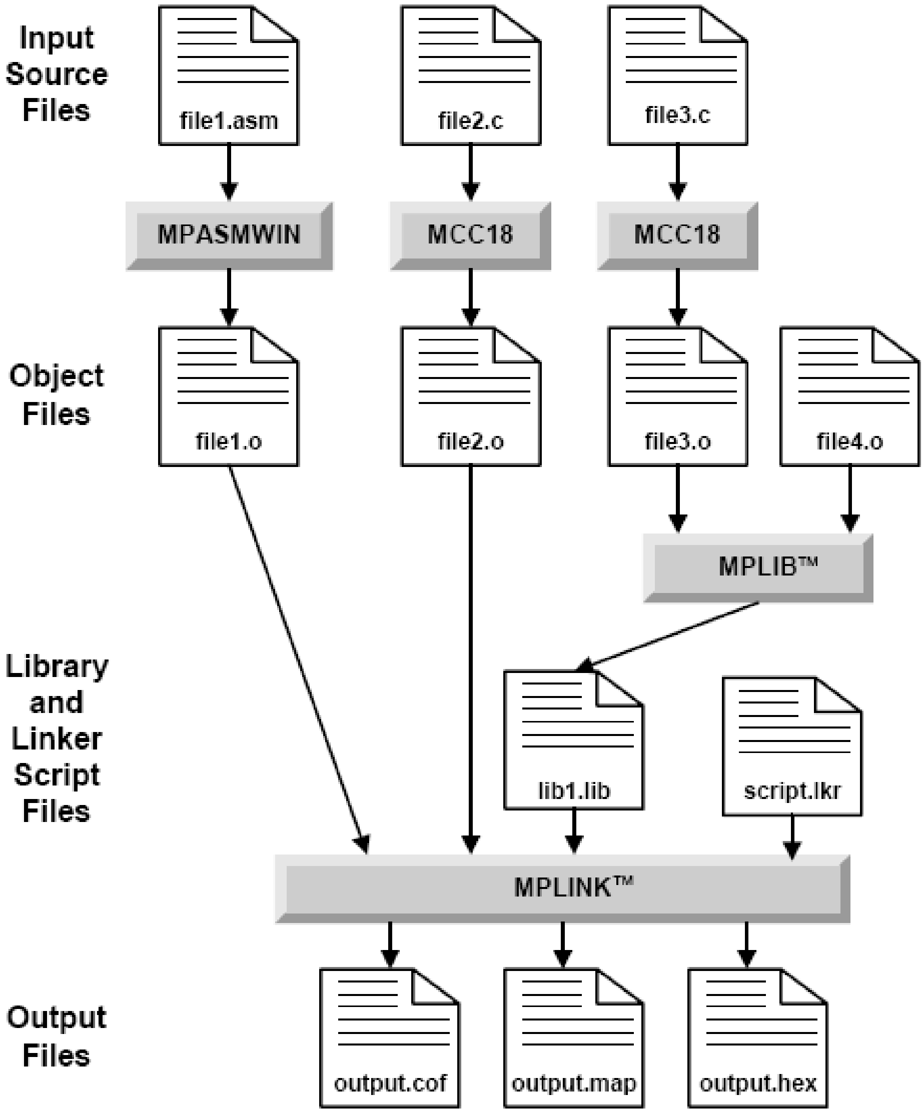


Figura 3.7 Diagrama de ejecución de herramientas de lenguaje

Vemos que para este caso en particular, dos archivos C son compilados por MPLAB C18: file2.c y file3.c, y un archivo de ensamblaje. File1.asm, es ensamblado por MPASM. Esto genera archivos objeto, llamados file1.o, file2.o y file3.o.

Un archivo objeto precompilado, file4.o, es usado con file3.o para formar una librería llamada lib1.lib. Finalmente, lor archivos objeto remanentes son combinados con los archivos de librería por el enlazador.

MPLINK también tiene un enlazador de guión de entrada, script.lkr. MPLINK produce los archivos de salida, output.cof y output.map, y el archivo HEX, output.hex.

3.2.1.2Herramienta C30

MPLAB C30 es un compilador de lenguaje C con extensiones de lenguaje, para dsPICs en aplicaciones de control embebidas. El compilador es una aplicación de consola de Windows que provee una plataforma para el desarrollo de código en lenguaje C.

Además compila archivos fuente en C produciendo archivos de ensamblaje y lenguaje C. Estos archivos son generados por el compilador y enlazados con otros archivos objeto y librerías para producir la aplicación final en archivos de formato COFF o ELF. El archivo COFF o ELF puede ser cargado en el MPLAB IDE donde es probado y depurado, o será usada la utilidad de conversión para convertir el archivo COFF o ELF al formato Intel hex; adecuado para cargar en la línea de comando del simulador o en un programador.

La siguiente figura muestra un resumido desarrollo del programa.

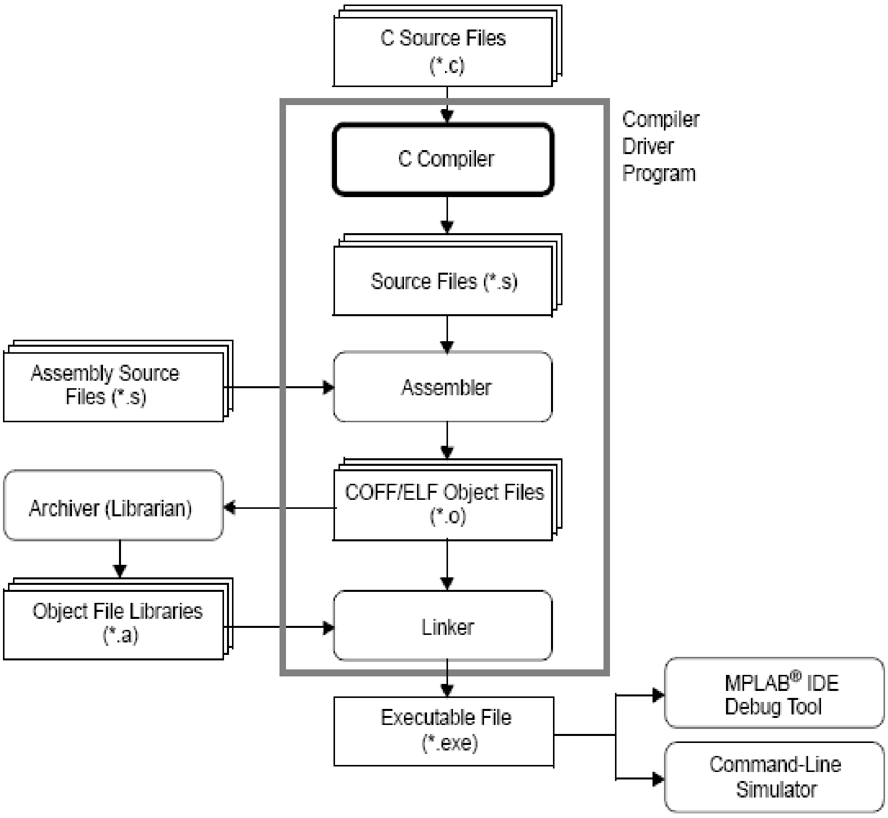


Figura 3.8 Desarrollo del programa

El compilador MPLAB C30 C traduce programas estándares ANSI C en lenguaje fuente de ensamblaje dsPIC DSC.

El estándar ANSI incluye extensiones para la definición original de C que ahora son características estándares de lenguaje. Estas extensiones aumentan la portabilidad y ofrecen un incremento de capacidad.

El compilador usa un conjunto de pasos de sofisticada optimización que emplean muchas técnicas avanzadas para generar eficiente y compacto código fuente en C. Los pasos de optimización incluyen ejecuciones de alto nivel que son aplicables a cualquier código C.

MPLAB C30 posee una librería estándar ANSI C, en la que sus funciones han sido validadas conforme a los requerimientos de la misma. La librería incluye funciones para manipulación de cadenas de caracteres, direcciones dinámicas de memoria, conversión de dato, cronometraje y funciones.

El código fuente es provisto en la distribución del compilador y puede ser usado como punto de inicio para las aplicaciones que requieren esta capacidad.

El compilador soporta dos modelos de acceso de constantes. El modelo “constantes en datos”, el cual usa memoria de datos y el modelo “constantes en código” que usa memoria de programa, que es accesada a través de la ventana de Visibilidad de Espacio de Programa.

3.2.2 Programador ICD2

El programador serial ICD 2 permite ambas funciones: la de depuración y programación de un dispositivo, ofreciéndonos las siguientes características:

- Ejecución paso a paso de código en tiempo real.
- Puntos de quiebre, Modificación de Registros y Variables.
- Monitoreo y diagnóstico de voltajes y Diodos de Emisión de Luz.
- Interfaz de usuario MPLAB IDE.
- Comunicación entre dispositivo y computador vía RS232 o USB.

MPLAB ICD2 es conectado al microcontrolador con el cable de interfaz modular, que es un conector de 6 pines. En la figura vemos la numeración y asignación de pines para el conector.

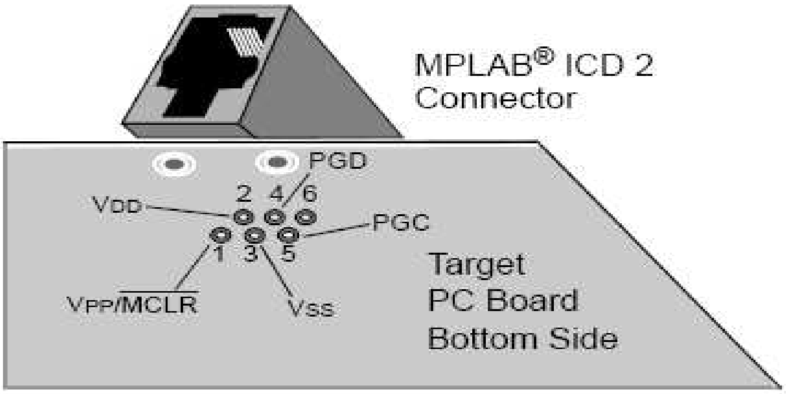


Figura 3.9 Asignación de pines de conector ICD2

Como vemos, tenemos seis pines en el conector ICD, pero son usados únicamente cinco, de los cuales se detallan las conexiones hacia el microcontrolador.

Es recomendado conectar una resistencia entre la línea Vpp/MCLR y la línea VDD.

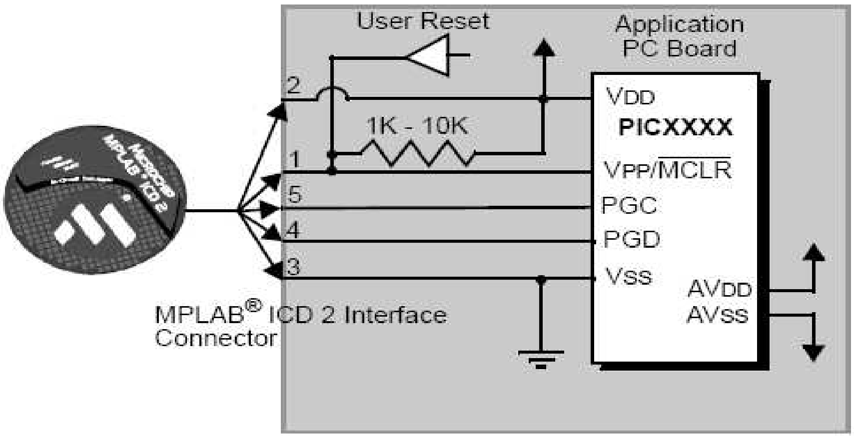


Figura 3.10 Conexión de MPLAB ICD2 al PIC

Además el pin 2 (VDD) será aquel que energice el dispositivo y el pin 3 será su respectiva tierra (VSS).

Modo de Depuración

Existen dos pasos para usar el MPLAB ICD 2 como depurador. El primero requiere que una aplicación sea programada en el Microcontrolador. El segundo usa el depurador interno de la memoria Flash para ejecutar y probar la aplicación del programa.

Estos dos pasos están directamente relacionados con las operaciones de MPLAB IDE: Programar el código en el microcontrolador y usar el depurador para seleccionar puntos de quiebre y ejecutar. Si el objetivo no puede ser programado correctamente, MPLAB ICD 2 no lo depurará.

Para depurar con MPLAB ICD 2 existen elementos críticos que deben estar trabajando correctamente.

MPLAB ICD 2 debe ser conectado a un computador, estar alimentado por una fuente externa o por el computador vía USB., y comunicarse con el computador por medio del puerto RS232 o por cable USB.

El MPLAB ICD 2 se conectará a los pines VPP, PGC y PGD del microcontrolador con el cable modular de interface. El microcontrolador posee energía y un oscilador para su correcto funcionamiento. Si el microcontrolador por alguna razón no tiene un buen funcionamiento, MPLAB ICD 2 no podrá depurar la aplicación.

Modo Programador

Si seleccionamos Programador>Programa, estamos dando paso a la programación del dispositivo; los registros de depuración deben ser deshabilitados en el MPLAB IDE para que el MPLAB ICD 2 programe únicamente el código de la aplicación y los bits de configuración en el microcontrolador.

En este modo el MPLAB ICD 2 solamente puede pasar la línea de reset e iniciar el objetivo. Un punto de quiebre no podrá ser seteado y los registros no podrán ser vistos o alterados. El MPLAB ICD 2 programa el objetivo usando ICSP. No se requiere una señal de reloj durante la programación y todos los modos del procesador pueden ser programados incluyendo el de protección de código.

3.2.2.1 Bus Serial Universal (USB)

El **Bus Serial Universal** (USB) es un puerto que sirve para conectar periféricos a una computadora. Fue creado en 1996 por siete empresas: IBM, Intel, Northern Telecom, Compaq, Microsoft, Digital Equipment Corporation y NEC.

El estándar incluye la transmisión de energía eléctrica al dispositivo conectado. Algunos dispositivos requieren una potencia mínima, así que se pueden conectar varios sin necesitar fuentes de alimentación extra. La gran mayoría de los concentradores incluyen fuentes de alimentación que brindan energía a los dispositivos conectados a ellos, pero algunos dispositivos consumen tanta energía que necesitan su propia fuente de alimentación. Los concentradores con fuente de alimentación pueden proporcionarle corriente eléctrica a otros dispositivos sin quitarle corriente al resto de la conexión (dentro de ciertos límites).

El diseño del USB tenía en mente eliminar la necesidad de adquirir tarjetas separadas para poner en los puertos bus ISA o PCI, y mejorar las capacidades plug-and-play permitiendo a esos dispositivos ser conectados o desconectados al sistema sin necesidad de reiniciar. Cuando se conecta un nuevo dispositivo, el servidor lo enumera y agrega el software necesario para que pueda funcionar.

El USB puede conectar los periféricos como ratón, teclados, escáneres, cámaras digitales, teléfonos celulares, reproductores multimedia, impresoras, discos duros externos, tarjetas de sonido, sistemas de adquisición de datos y componentes de red. Para dispositivos multimedia como escáneres y cámaras digitales, el USB se ha convertido en el método estándar de conexión. Para impresoras, el USB ha crecido tanto en popularidad que ha empezado a desplazar a los puertos paralelos porque el USB hace sencillo el poder agregar más de una impresora a una computadora personal.

En el caso de los discos duros, el USB es poco probable que reemplace completamente a los buses como el ATA (IDE) y el SCSI porque el USB tiene un rendimiento un poco más lento que esos otros estándares. El nuevo estándar Serial ATA permite tasas de transferencia de hasta aproximadamente 150/300 MB por segundo. Sin embargo, el USB tiene una importante ventaja en su habilidad de poder instalar y desinstalar dispositivos sin tener que abrir el sistema, lo cual es útil para dispositivos de almacenamiento externo. Hoy en día, una gran parte de los fabricantes ofrece dispositivos USB portátiles que ofrecen un rendimiento casi indistinguible en comparación con los ATA (IDE).

El USB no ha remplazado completamente a los teclados AT y mouse PS/2, pero virtualmente todas las placas base de PC traen uno o más puertos USB.

Características de Transmisión

Los dispositivos USB se clasifican en cuatro tipos según su velocidad de transferencia de datos:

- Baja Velocidad (1.0): Tasa de Bit de 1.5Mbit/s (192KB/s). Utilizado en su mayor parte por Dispositivos de Interfaz Humana (HID) como los teclados, los ratones y los joysticks.
- Velocidad Completa (1.1): Tasa de Bit de 12Mbit/s (1.5MB/s). Esta fue la más rápida antes de que se especificara la USB 2.0 y muchos dispositivos fabricados en la actualidad trabajan a esta velocidad. Estos dispositivos, dividen el ancho de banda de la conexión USB entre ellos basados en un algoritmo FIFO.
- Alta Velocidad (2.0): Tasa de Bit de 480Mbit/s (60MB/s).
- Súper Velocidad (3.0) Actualmente en fase experimental. Tasa de Bit de 4.8Gbit/s (600MB/s). Esta especificación será lanzada a mediados de 2008 por la compañía Intel, de acuerdo a información recabada de Internet. Las velocidades de los buses serán 10 veces más rápidas que la de USB 2.0 debido a la

> inclusión de un enlace de fibra óptica que trabaja con los conectores tradicionales de cobre. Se espera que los productos fabricados con esta tecnología lleguen al consumidor en 2009 o 2010.

Las señales del USB son transmitidas en un cable de datos, par trenzado con impedancia de 90Ω ±15% llamados D+ y D-. Éstos, colectivamente utilizan señalización diferencial en half dúplex para combatir los efectos del ruido electromagnético en enlaces largos. D+ y D- usualmente operan en conjunto y no son conexiones simplex. Los niveles de transmisión de la señal varían de 0 a 0.3V para bajos (ceros) y de 2.8 a 3.6V para altos (unos) en las versiones 1.0 y 1.1, y en ±400mV en Alta Velocidad (2.0). En las primeras versiones, los alambres de los cables no están conectados a masa, pero en el modo de alta velocidad se tiene una terminación de 45 Ω a tierra o un diferencial de 90 Ω para acoplar la impedancia del cable. Este puerto sólo admite la conexión de dispositivos de bajo consumo, es decir, que tengan un consumo máximo de 100mA por cada puerto, pero en caso que estuviese conectado un dispositivo que permite 4 puertos por cada salida USB (extensiones de máximo 4 puertos) entonces la energía del USB se asigna en unidades de 100mA hasta un máximo de 500mA por puerto.

Pin	Nombre	Color del Cable	Descripción
1	VCC	Rojo	+5V
2	D−	Blanco	Data −
3	D+	Verde	Data +
4	GND	Negro	Tierra

Tabla 3.2 Distribución y asignación de pines del USB

Compatibilidad y Conectores

Existen diferente tipos de Conectores USB como: Micro USB, Mini USB, Tipo B, Hembra tipo A y Tipo A.

El estándar USB especifica tolerancias para impedancia y de especificaciones mecánicas relativamente bajas para sus conectores, intentando minimizar la incompatibilidad entre los conectores fabricados por distintas compañías. Una meta a la que se ha logrado llegar. El estándar USB, a diferencia de otros estándares también define tamaños para el área alrededor del conector de un dispositivo, para evitar el bloqueo de un puerto adyacente por el dispositivo en cuestión.

Las especificaciones USB 1.0, 1.1 y 2.0 definen dos tipos de conectores para conectar dispositivos al servidor: A y B. Sin embargo, la capa mecánica ha cambiado en algunos conectores. Por ejemplo, el IBM UltraPort es un conector USB privado localizado en la parte superior del LCD de los computadores portátiles deIBM. Utiliza un conector mecánico diferente mientras mantiene las señales y protocolos característicos del USB. Otros fabricantes de artículos pequeños han desarrollado también sus medios de conexión pequeños, y ha aparecido una gran variedad de ellos, algunos de baja calidad.

Una extensión del USB llamada "USB sobre la marcha" permite a un puerto actuar como servidor o como dispositivo. Incluso después de que el cable está conectado y las unidades se están comunicando, las 2 unidades pueden "cambiar de papel" bajo el control de un programa. Esta facilidad está específicamente diseñada para dispositivos como PDA, donde el enlace USB podría conectarse a un PC como un dispositivo, y conectarse como servidor a un teclado o ratón. El "USB-On-The-Go" también ha diseñado 2 conectores pequeños, el mini-A y el mini-B, así que esto debería detener la proliferación de conectores miniaturizados de entrada.

Programadores

- PICStart Plus (puerto serie y USB)
- Promate II (puerto serie)
- MPLAB PM3 (puerto serie y USB)
- ICD2 (puerto serie y USB)
- PICKit 1 (USB)
- PICAT 1.25 (puerto USB2.0 para PICs y Atmel)
- WinPic 800 (puerto paralelo, serie y USB)
- Terusb1.0 (USB)

Depuradores integrados

- ICD (Serie)
- ICD2 (USB)

Emuladores

- ICE2000 (puerto paralelo, convertidor a USB disponible)
- ICE4000 (USB)
- PIC EMU

- PIC CDlite

3.2.2.2 Configuración del Programador

Como citamos anteriormente, el programador ICD2 puede funcionar como depurador o como un programador serial; para nuestra aplicación lo vamos a usar únicamente para programar nuestra tarjeta de aplicación.

Para configurar el ICD2, primeramente debemos definir varias opciones en el MPLAB IDE. Una vez instalados los controladores de la comunicación hacia el computador por medio del puerto USB, iniciamos el MPLAB IDE donde desarrollamos y guardamos nuestro código fuente.

Terminado nuestro código seleccionamos el dispositivo en el cuadro de diálogo de Selección de Dispositivo para elegir el modelo a depurar o programar con MPLAB ICD 2. Posteriormente seleccionamos MPLAB ICD 2 como la herramienta de depuración (Depurador>Seleccionar Herramienta>MPLAB ICD2).

La comunicación se llevará a cabo por medio del puerto USB, por ello debemos seleccionar en la tabla de comunicación el menú “Depurador” (Debugger>Settings).

Para llevar a cabo la programación, ingresamos al menú Programador y a la opción Seleccionar Programador, donde elegiremos MPLAB ICD2.

El siguiente paso será configurar las características del programador (Programmer>Settings). Configuramos luego los Bits de Configuración tal como se encuentran en nuestro programa, incluyendo el oscilador en el menú configurar. Realizamos un chequeo de que el dispositivo a programar esté en blanco o haya sido borrado para que quede listo para la programación.

Finalmente seleccionamos “Programar” en el menú Programador (Programmer>Program) para grabar el código en nuestra tarjeta.

CAPÍTULO 4

4. DISEÑO Y DESARROLLO DEL SISTEMA

4.1 Descripción General

Como ya hemos dicho al inicio de nuestra documentación, la domótica es el resultado de la integración de los sistemas de gestión de seguridad, comunicaciones, gestión del confort y control energético. Estudiando las diferentes aplicaciones prácticas y características del PIC18F97J60 en especial la del módulo Ethernet, y tomando en cuenta la importancia que tiene hoy en día el Internet en la vida diaria del ser humano, hemos pensado en diseñar e implementar un sistema que integre estas tecnologías en una aplicación útil, que permita el monitoreo y control de las luminarias de una vivienda en forma remota.

Para conseguir desarrollar nuestro proyecto, es necesario recurrir a la modularidad y dividir el sistema total en diferentes sistemas parciales, tales como: el programa o software del microcontrolador, aplicación web, tarjeta de control y tarjeta de potencia. Con estos módulos conseguimos simplificar el trabajo, los cuales serán explicados detalladamente a lo largo de todo este capítulo.

La tarjeta de control como su mismo nombre lo indica, nos permite controlar la tarjeta de potencia, por medio de instrucciones que da el usuario a través de una interfaz web definida y guardada en el microcontrolador como lo indique el programa.

La tarjeta de potencia permitirá la adaptación de voltajes entre la tarjeta de control y el circuito de luces que mostrará su funcionamiento, de tal manera que las señales digitales de salida del microcontrolador se amplifiquen para obtener el potencial necesario con el objetivo de encender y apagar las luminarias.

La aplicación web de fácil manejo, nos permitirá crear una interfaz gráfica entre el usuario y la tarjeta de control. En el cual la persona podrá visualizar el estado de las luminarias y manipularlas según su conveniencia.

El programa o software del microcontrolador nos permitirá utilizar y manipular las diferentes características físicas, y lógicas del mismo mediante instrucciones ya definidas por el fabricante. Además de darle el funcionamiento requerido para el cual ha sido diseñado nuestro proyecto.

Para demostrar el funcionamiento de la comunicación por la red de Internet, conectaremos la tarjeta de control al punto de red más cercano y esta a su vez estará conectada a la tarjeta de potencia que dará la energía necesaria para el encendido y apagado de luminarias como lo indica nuestra aplicación. También necesitaremos conectar una computadora personal a otro punto de red distinto, mediante la cual accederemos a la aplicación web que nos ayudara a monitorear y controlar remotamente la aplicación final.

4.1.1 Diagrama de Bloques del Sistema

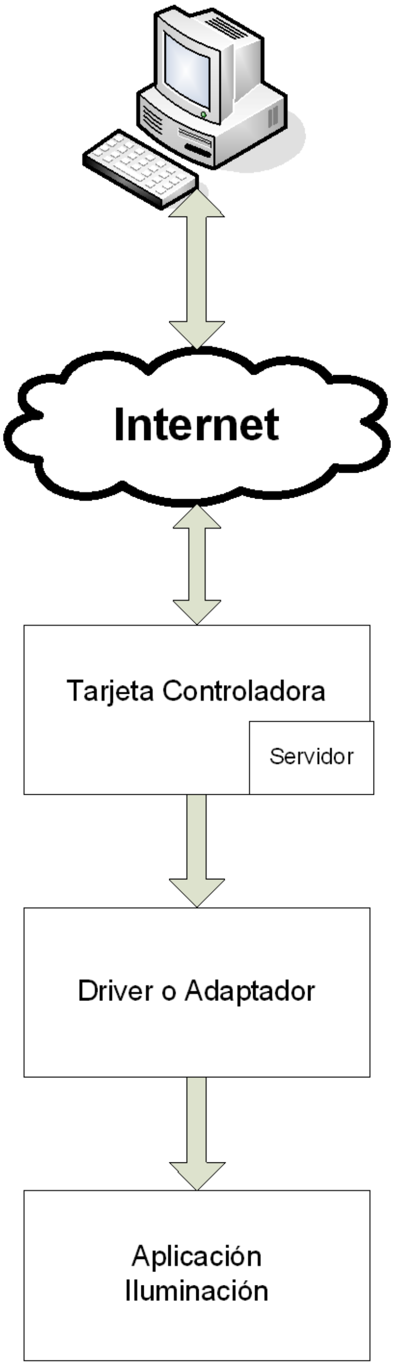


Figura 4.1 Diagrama de Bloques del Sistema.

Como podemos observar en el diagrama de bloques que mostramos en este capítulo, nuestro sistema consta de cinco partes fundamentales para su correcto funcionamiento, que son: una computadora personal, la red de Internet, la tarjeta controladora, el adaptador y nuestra aplicación; las cuales detallaremos a continuación.

Para nuestra computadora personal los requerimientos son mínimos, es suficiente con que esta pueda conectarse y trabajar a través de la red de internet con facilidad.

El usuario tendrá la comodidad de transmitir las instrucciones mediante el computador y estas viajarán a través de la red cumpliendo los protocolos del modelo TCP/IP y siguiendo los estándares que involucran toda una conectividad. Además, por la misma vía se podrá recibir los paquetes de datos con el estado del microcontrolador, de esta manera podremos controlar y monitorear nuestra aplicación.

El siguiente paso del diagrama involucra la tarjeta controladora que posee como parte principal el microcontrolador PIC18F97J60 programado previamente para cumplir las funciones requeridas en nuestro proyecto, tomando en cuenta que la tarjeta deberá estar conectada a internet por medio de un cable de red directo de 8 hilos con conectores RJ45 en ambos extremos. Este bloque es el responsable de procesar todas las instrucciones recibidas por el usuario, para luego ejecutarlas y transmitirlas hacia el bloque de potencia; siendo así la parte más importante de todo nuestro sistema. Para poder acceder a nuestra tarjeta de control desde cualquier parte de la red necesitaremos de la ayuda de un servidor, el cual esta incluido en el microcontrolador.

Siguiendo con nuestro diagrama de bloques encontramos la etapa adaptadora o de potencia que será la encargada de amplificar los voltajes de salida de la tarjeta controladora desde 3.3 voltios dc hasta los 12 voltios dc que son los que requerimos para nuestra aplicación. Este bloque constará de amplificadores y relés de conmutación como detallaremos más adelante, que trabajarán en conjunto con el fin de obtener la energía suficiente para encender o apagar las luminarias en nuestra aplicación.

La etapa final o de aplicación es la que nos demostrará el correcto funcionamiento de nuestro sistema y consistirá en bombillos de luz que serán activados o desactivados por la tarjeta de potencia.

4.2 Programación del Microcontrolador

Para la programación del Microcontrolador, nos hemos apoyado en la versión de programación TCP/IP 4.11 que facilita el fabricante de forma libre. Para que nuestro programa cumpla una de las normas del buen programador, es decir todo sistema debe ser estructurado, hemos dividido nuestro código en funciones que serán utilizadas para el adecuado funcionamiento de nuestra aplicación.

Como podemos observar en el siguiente diagrama de bloques de la figura 4.2, se muestra el procedimiento de la función principal ‘main’. Empezamos por declarar e inicializar las variables locales ‘t’, del tipo TICK e ‘i’, del tipo BYTE. Luego procedemos a inicializar la tarjeta controladora y el LCD, haciendo un llamada a las funciones ‘Inicializar_Tarjeta’ y ‘LCDInit’ respectivamente. Después de poner un retardo aproximadamente de 100 milisegundos, realizamos la llamada a una función para poner el texto ‘Sandoya-Montegro’ en el LCD, finalmente para que se pueda apreciar esto, llamamos a la función ‘LCDUpdate’.

Continuando con la configuración, inicializamos el tiempo (timer 0) y la variable global ‘AppConfig’, llamando a las funciones ‘TickInit’ e ‘Inicializar_AppConfig’. En este punto realizamos una rutina para permitir que el usuario pueda borrar el contenido de la EEPROM, con un simple procedimiento que detallamos a continuación:

1. Mantener presionado el botón RB3.
2. Presionar y soltar el botón MCLR.
3. Continuar presionando el botón RB3, por aproximadamente 4 segundos.
4. Dejar de presionar el botón RB3.
5. Presionar y soltar nuevamente el botón MCLR, para restaurar la aplicación.

En este mismo diagrama de bloques se puede observar que si se repite los pasos 1 y 2, pero se mantiene presionado el botón RB3 menos de 4 segundos, se procede a llamar a la función ‘Seteo_Config’, cuya explicación detallada se lo hará en el diagrama de bloques de dicha función que está representado en la figura 4.7.

Una vez explicada esta rutina, el programa continúa con la inicialización del stack, del módulo EUSART y del protocolo HTTP, llamando a las funciones ‘StackInit’, ‘UART2TCPBridgeInit’ y ‘HTTPInit’ consecutivamente. Después continua con una pequeña rutina, la cual me sirve para contar cuantas veces se ha deshabilitado el DHCP.

En esta parte del programa tenemos un lazo infinito, dentro de este lazo se llaman a las diferentes funciones que ejecutan tareas específicas que hacen que nuestra aplicación funcione correctamente. Estas funciones las explicamos a continuación:

- **StackTask:** Maneja el paquete Rx interno para preprocesarlo previamente a ser despachado a las capas de aplicación superiores. Esta función chequea nuevos paquetes entrantes y guía a ellos hacia los componentes de la pila apropiado.

- **UART2TCPBridgeTask**: Transmite todos los bytes TCP entrantes en una ranura o conexión exterior del modulo USART.

- **HTTPServer**: Servidor de páginas dinámicas para los navegadores web tales como el Explorador de Internet de Microsoft, Mozilla Firefox, etc.

- **DiscoveryTask**: Examina los paquetes de broadcast para saber si se trata de una petición de descubrimiento, luego comenzamos el envío de nuestra dirección MAC en formato legible (convertimos la dirección MAC de bytes a hexadecimal), al último nodo de donde recibimos la petición.

- **NBNSTask**: Responde a la petición de nombre NBNS para permitir la asignación de un nombre (dado por el usuario), a la tarjeta. Es decir, permite a los nodos de la misma subred IP usar un nombre de cliente en lugar de una dirección IP para acceder a nuestra tarjeta.

- **DHCPServerTask**: Proporciona una automatica dirección IP, máscara de subred, dirección de puerta de enlace, la dirección del servidor DNS, y otros parámetros de configuración DHCP utilizados en redes.
- **GenericTCPServer**: Esta función implementa un servidor TCP simple. La función se invoca periódicamente por nuestra aplicación para escuchar las conexiones entrantes. Cuando una conexión es realizada, el servidor lee todos los datos que ingresan, lo transforma a mayúsculas, y este proceso se repite de regreso. Esto debe ser usado como una base para crear nuevas aplicaciones de servidor TCP.
- **TelnetTask**: Proporciona el servicio de Telnet sobre el puerto TCP 23.
- **RebootTask**: Permite reiniciar remotamente el PIC. Esta función revisa el tráfico entrante en el puerto 30304 y reinicia el PIC si una ‘R’ es recibido.

- **SNTPClient**: Esta función localiza un servidor NTP(Protocolo de Internet utilizado para sincronizar los relojes de los ordenadores a un tiempo de referencia), de un lugar público usando DNS. Pide el tiempo UTC(estandar oficial para el tiempo actual), usando SNTP y actualiza periodicamente la estructura SNTPtime.

Adicionalmente, en dicho lazo tenemos una rutina la cual sirve para actualizar la dirección IP en el momento que el usuario habilite el modo DHCP.

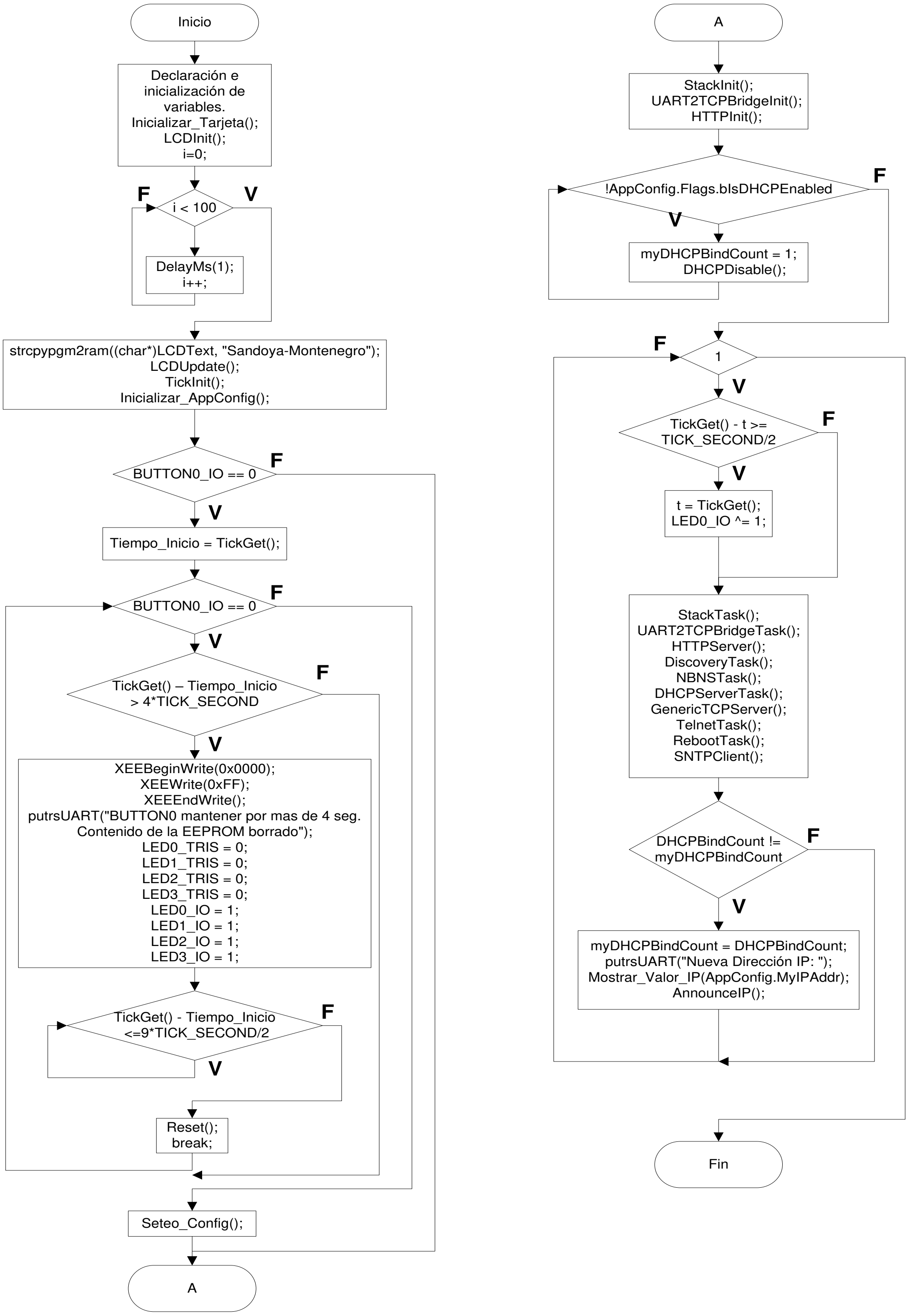


Figura 4.2 Diagrama de flujo de la función main

El diagrama de bloques de la figura 4.3, se muestra la secuencia de la función ‘Inicializar_Tarjeta’. Se empieza por la configuración del PORTJ como salidas. En el registro OSCTUNE, habilitamos el bloque PLL con un reloj de 41.6667 MHz. Con RBPU en ‘0’, del registro INTCON2, habilitamos todas las resistencias internas del PORTB. En TXSTA, habilitamos la transmisión (Tx) y el modo asíncrono en el modulo EUSART. En el registro RCSTAT, habilitamos el puerto serial y el receptor en el módulo EUSART. SPBRG, es el registro generador de la tasa de baudios en el módulo EUSART, la cual se la calcula mediante la fórmula que se muestra en el diagrama de bloques de la figura 4.3. Como la configuración esta en modo asíncrono, seleccionamos la velocidad alta poniendo en ‘1’ el bit BRGH del registro TXSTA. El bit IPEN del registro RCON lo colocamos en ‘1’, para habilitar los niveles de prioridad de las interrupciones. Una vez ejecutado el paso anterior (que es requisito para nuestro próximo paso), configuramos los bits GIEH y GIEL en ‘1’, para habilitar los niveles alto y bajo de prioridad.

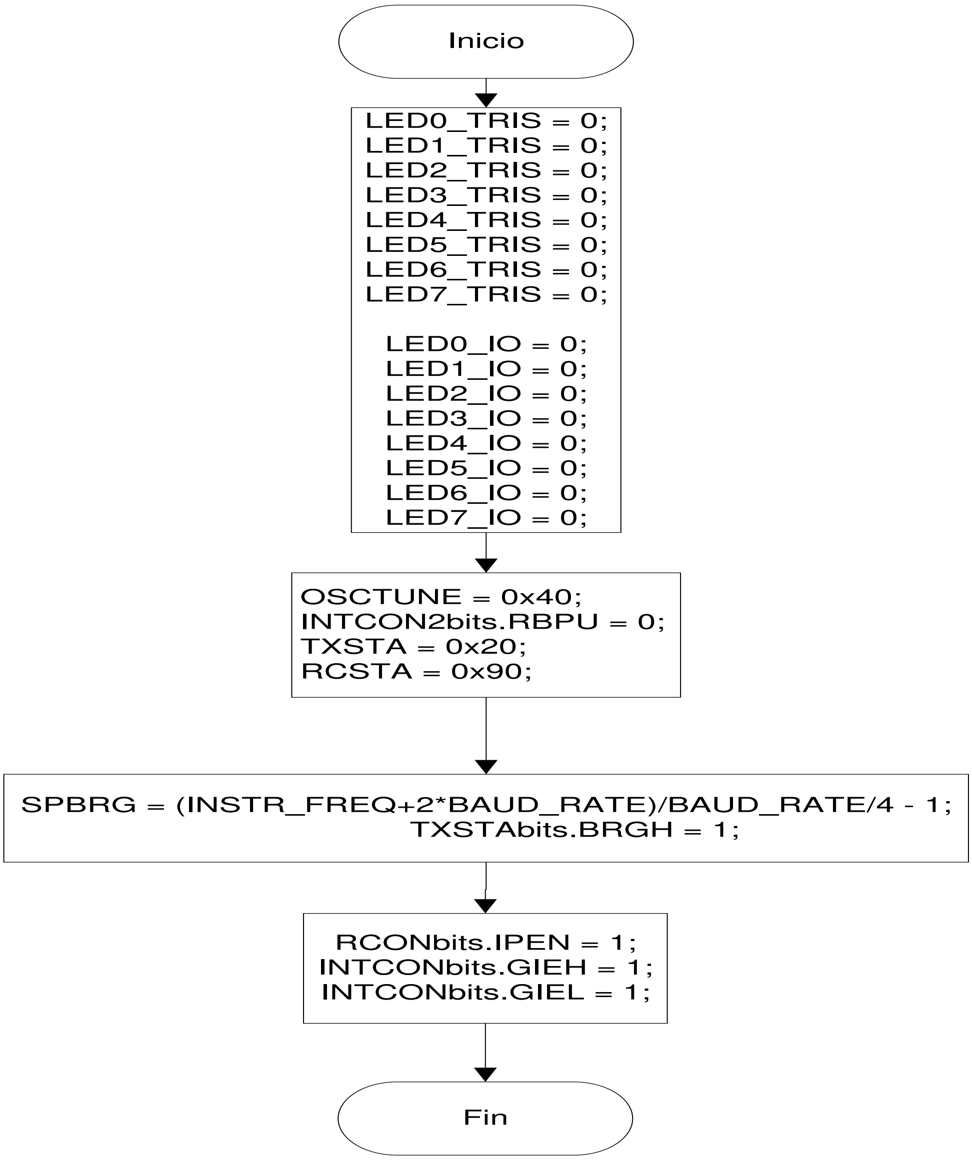
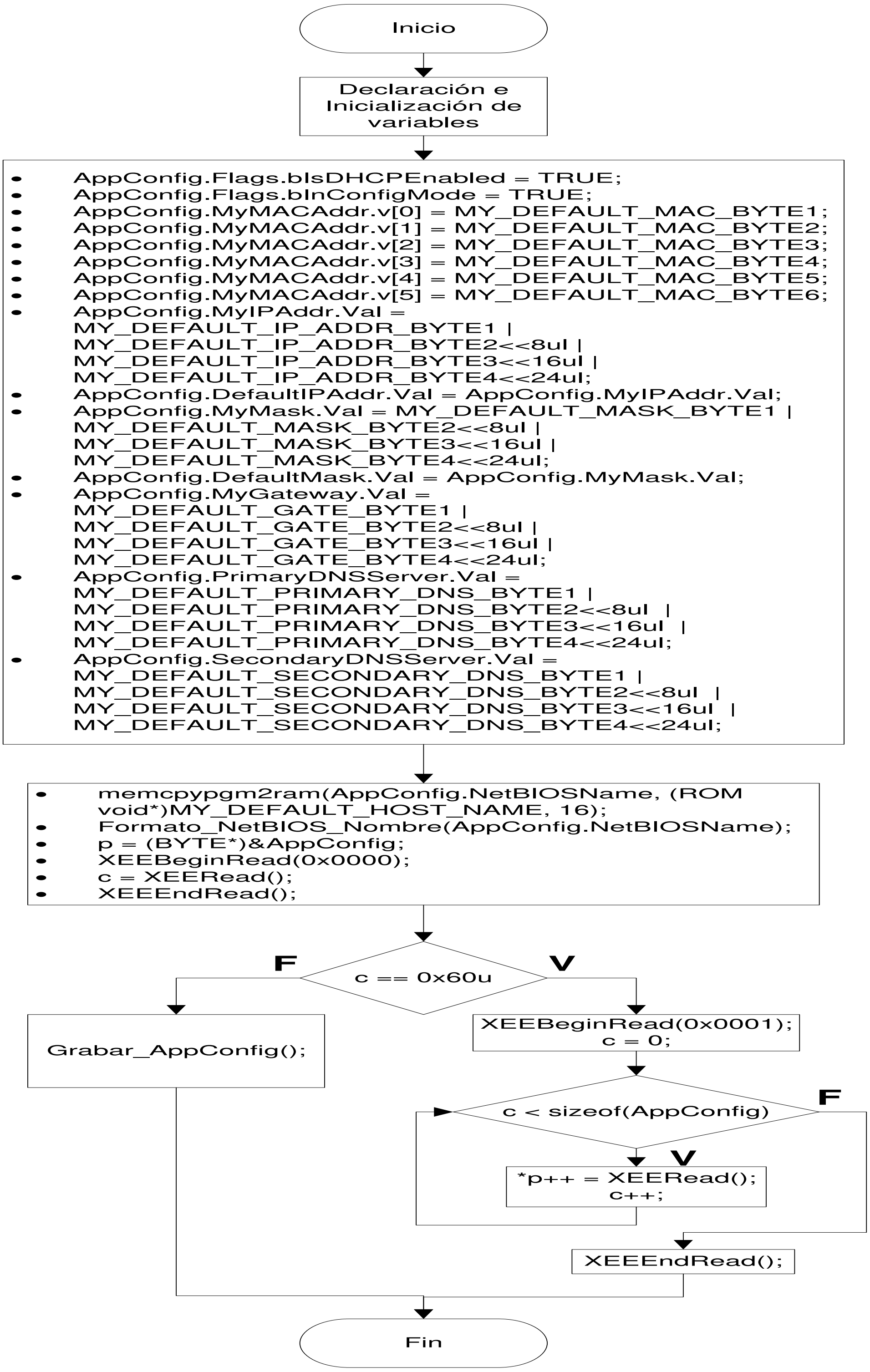


Figura 4.3 Diagrama de flujo de la función Inicializar_Tarjeta

La figura 4.4, muestra el diagrama de bloques de la función 'Inicializar_AppConfig'. Comenzamos con la declaración e inicialización de las variables locales que utilizaremos en esta función. Luego procedemos a inicializar todos los campos de la variable global

‘AppConfig’. El nombre de host por defecto lo copiamos en el campo NETBIOS de la variable ‘AppConfig’, después de esto procedemos a darle un formato de nombre de NETBIOS a dicho campo, llamando a la función Formato_NetBIOS_Nombre. Despues se procede a leer la EEPROM para comprobar si hay algun registro corrupto, este resultado se guarda en nuestra variable local ‘c’. Cuando un registro es guardado, el primer byte es escrito como 0X60, para indicar que un registro valido fue grabado. Si nuestra variable ‘c’, tiene un valor diferente, la aplicación empieza a reescribir la información en la EEPROM.



```
• AppConfig.Flags.bIsDHCPEnabled = TRUE;
• AppConfig.Flags.bInConfigMode = TRUE;
• AppConfig.MyMACAddr.v[0] = MY_DEFAULT_MAC_BYTE1;
• AppConfig.MyMACAddr.v[1] = MY_DEFAULT_MAC_BYTE2;
• AppConfig.MyMACAddr.v[2] = MY_DEFAULT_MAC_BYTE3;
• AppConfig.MyMACAddr.v[3] = MY_DEFAULT_MAC_BYTE4;
• AppConfig.MyMACAddr.v[4] = MY_DEFAULT_MAC_BYTE5;
• AppConfig.MyMACAddr.v[5] = MY_DEFAULT_MAC_BYTE6;
• AppConfig.MyIPAddr.Val =
  MY_DEFAULT_IP_ADDR_BYTE1 |
  MY_DEFAULT_IP_ADDR_BYTE2<<8ul |
  MY_DEFAULT_IP_ADDR_BYTE3<<16ul |
  MY_DEFAULT_IP_ADDR_BYTE4<<24ul;
• AppConfig.DefaultIPAddr.Val = AppConfig.MyIPAddr.Val;
• AppConfig.MyMask.Val = MY_DEFAULT_MASK_BYTE1 |
  MY_DEFAULT_MASK_BYTE2<<8ul |
  MY_DEFAULT_MASK_BYTE3<<16ul |
  MY_DEFAULT_MASK_BYTE4<<24ul;
• AppConfig.DefaultMask.Val = AppConfig.MyMask.Val;
• AppConfig.MyGateway.Val =
  MY_DEFAULT_GATE_BYTE1 |
  MY_DEFAULT_GATE_BYTE2<<8ul |
  MY_DEFAULT_GATE_BYTE3<<16ul |
  MY_DEFAULT_GATE_BYTE4<<24ul;
• AppConfig.PrimaryDNSServer.Val =
  MY_DEFAULT_PRIMARY_DNS_BYTE1 |
  MY_DEFAULT_PRIMARY_DNS_BYTE2<<8ul  |
  MY_DEFAULT_PRIMARY_DNS_BYTE3<<16ul  |
  MY_DEFAULT_PRIMARY_DNS_BYTE4<<24ul;
• AppConfig.SecondaryDNSServer.Val =
  MY_DEFAULT_SECONDARY_DNS_BYTE1 |
  MY_DEFAULT_SECONDARY_DNS_BYTE2<<8ul  |
  MY_DEFAULT_SECONDARY_DNS_BYTE3<<16ul  |
  MY_DEFAULT_SECONDARY_DNS_BYTE4<<24ul;
```

```
• memcpypgm2ram(AppConfig.NetBIOSName, (ROM
  void*)MY_DEFAULT_HOST_NAME, 16);
• Formato_NetBIOS_Nombre(AppConfig.NetBIOSName);
• p = (BYTE*)&AppConfig;
• XEEBeginRead(0x0000);
• c = XEERead();
• XEEEndRead();
```

```
c == 0x60u
Grabar_AppConfig();
XEEBeginRead(0x0001);
c = 0;
c < sizeof(AppConfig)
*p++ = XEERead();
c++;
XEEEndRead();
```

Figura 4.4 Diagrama de flujo de la función Inicializar_AppConfig

La figura 4.5, proporciona el diagrama de bloques de la función ‘Grabar_AppConfig’, dicha función nos permite guardar nuestra variable global ‘AppConfig’ en la EEPROM. Empezamos por declarar e inicializar las variables. Luego procedemos a inicializar la grabación, donde nuestro primer byte será escrito como 0X60 como lo mencionamos anteriormente. Después empezamos a grabar byte por byte hasta que dicha variable haya sido graba completamente (esta condición sería hasta que ‘c < sizeof(AppConfig)’), y finalmente cerramos el ciclo de grabación.

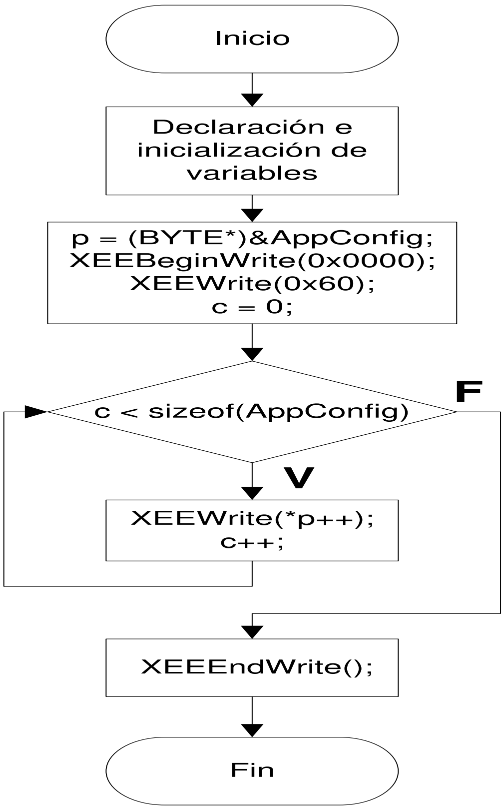


Figura 4.5 Diagrama de flujo de la función Grabar_AppConfig

La figura 4.7, presenta el diagrama de bloques de la función 'Seteo_Config', que nos permite configurar los campos de la variable global 'AppConfig' por medio de la interfaz RS-232. Para poder entrar en este modo seguimos los siguientes pasos:

1. Del menú Inicio, seleccionar. Programas/Accesorios/Comunicaciones/HyperTerminal.

2. En el cuadro de dialogo inicial 'Descripción de la conexión', ingresar un nombre para la conexión. Se puede poner cualquier nombre fácil de recordar. Clic en **OK**.

3. En el cuadro de dialogo 'Conectar a', escoger el apropiado puerto COM del menú inferior. Clic **OK**.

4. En el cuadro de dialogo 'Propiedades COM' que sigue, seleccionar la siguiente configuración:

   Bits por segundo: 19200

   Bits de Datos: 8

   Paridad: ninguna

   Bits de Parada: 1

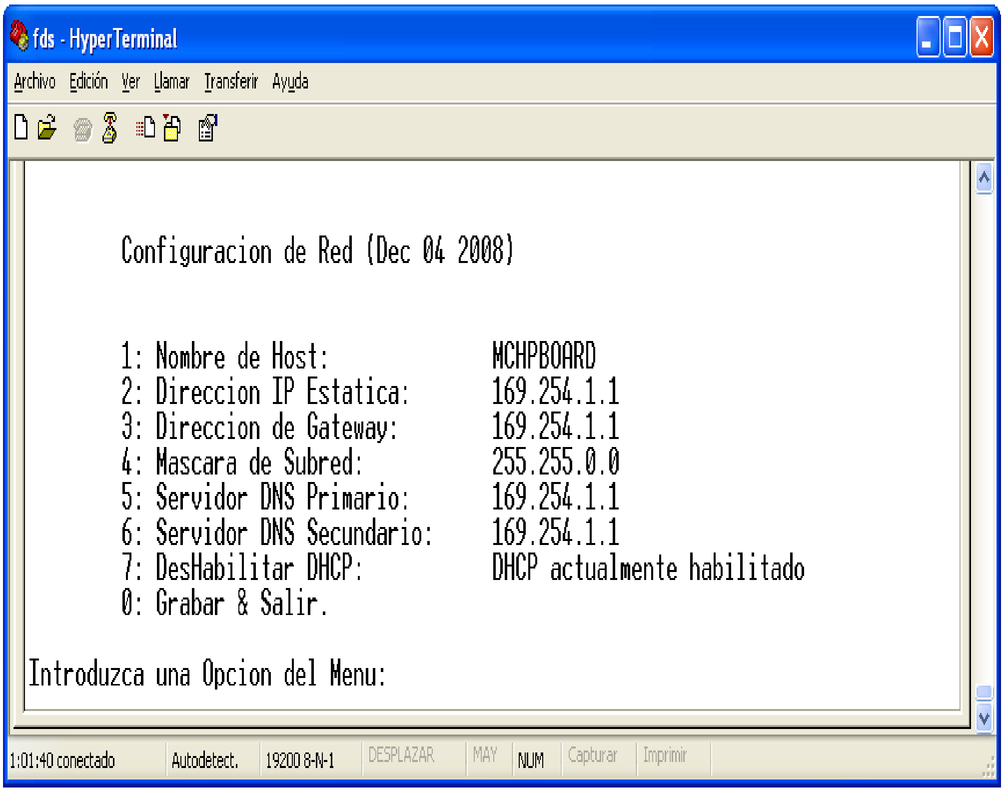
   Control de Flujo: ninguno

Clic **OK**. La ventana del Terminal se abre con el cursor parpadeando. El mensaje ‘Conectado’, aparece en la barra de estado al pie de la ventana del Terminal, también en la misma barra se muestra el tiempo de enlace de la conexión.

5. Seleccionar Archivo/Propiedades, entonces la viñeta **Configuraciones** en el dialogo ‘Propiedades’. Clic en el botón **Configuración ASCII** y dar un visto en el cuadro de **Eco de los caracteres escritos localmente** en el siguiente cuadro de dialogo. Clic **OK** para salir y finalmente **OK** para salir del dialogo ‘Propiedades’.

6. Mientras se mantenga presionado el botón RB3, presionar y soltar el botón MCLR. El Terminal responde con el menú de configuración serial (Figura). En el mismo tiempo, el LCD muestra: “Sandoya-Montenegro”**.** En este punto, soltar el botón RB3.

7. Una vez realizado los pasos anteriores se procede a realizar las configuraciones deseadas. Como por ejemplo, dirección IP, nombre de la estación de trabajo, mascara de subred, etc. En la misma figura 4.6, se indica de forma detallada como realizar la configuración idónea.



```
        Configuracion de Red (Dec 04 2008)


        1: Nombre de Host:          MCHPBOARD
        2: Direccion IP Estatica:   169.254.1.1
        3: Direccion de Gateway:    169.254.1.1
        4: Mascara de Subred:       255.255.0.0
        5: Servidor DNS Primario:   169.254.1.1
        6: Servidor DNS Secundario: 169.254.1.1
        7: DesHabilitar DHCP:       DHCP actualmente habilitado
        0: Grabar & Salir.

Introduzca una Opcion del Menu:
```

Figura 4.6 Menú de Configuración de Red

En este algoritmo se empieza por declarar e inicializar las variables locales, luego utilizamos un lazo con una bandera como condición. Con una serie de instrucciones manipulamos el menú que se muestra en la figura 4.6. Después se espera hasta que el usuario presione una tecla seleccionando la opción deseada. La opción escogida por el usuario se almacena en una variable, donde con la ayuda de una función de selección ‘switch’, la aplicación procede a ejecutar la configuración deseada por el usuario. Después de cada configuración seleccionada nuestro programa regresa al menú principal. Para finalizar, se debe seleccionar la opción ‘0’, donde la bandera es activada, permitiendo de esta manera salir de este modo de configuración.

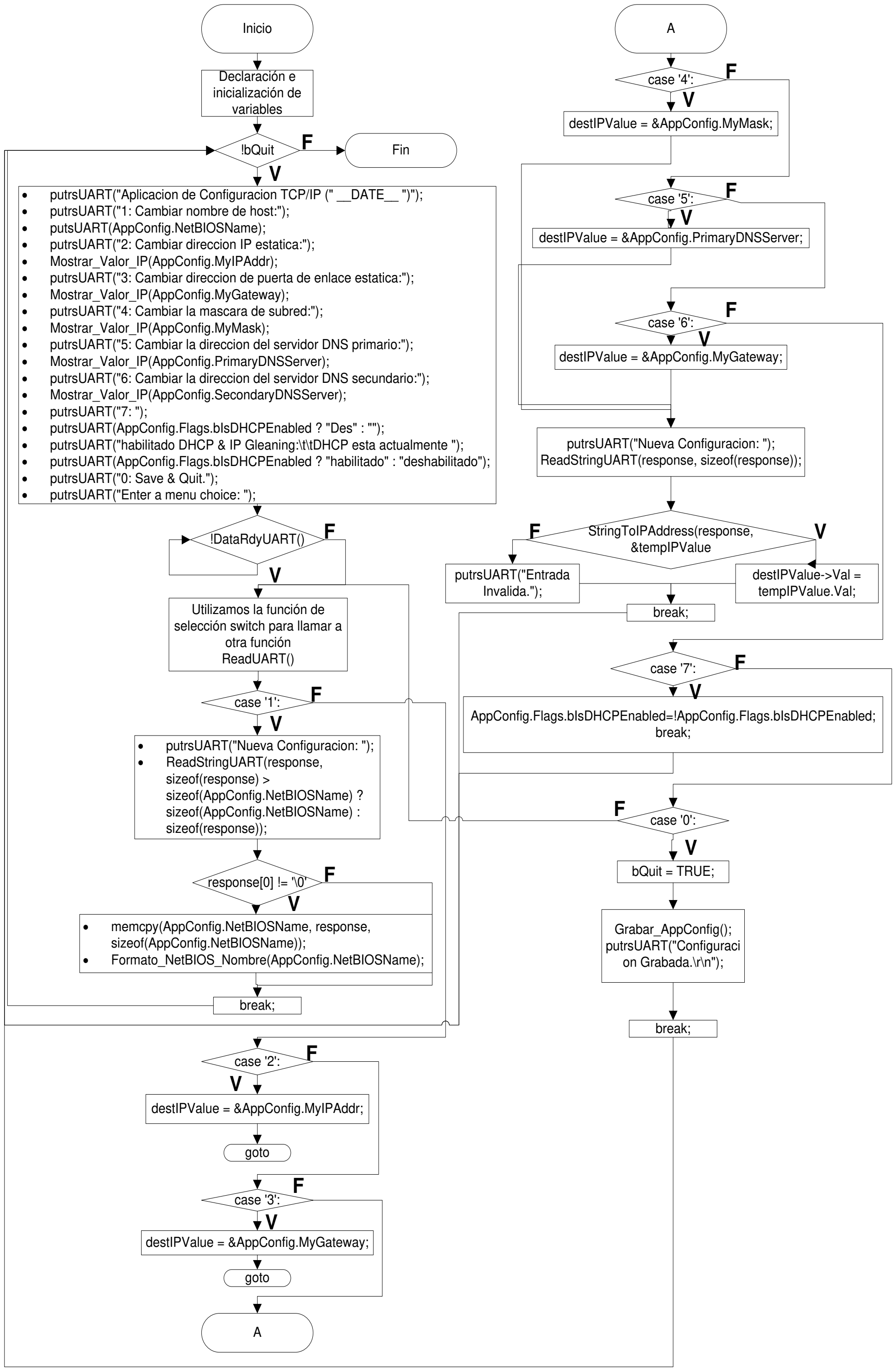


Figura 4.7 Diagrama de flujo de la función Seteo_Config

El diagrama de Bloques de la figura 4.8, muestra el algoritmo de la función ‘Formato_NetBIOS_Nombre’, la cual nos sirve para poner al nombre de cliente en formato con el nombre de NETBIOS. Este algoritmo comienza con la declaración e inicialización de las variables. El nombre NETBIOS tiene una longitud máxima de 16 caracteres, pero solo se permiten 15 porque el ultimo esta reservado para el sistema. La función ‘strupr’, se la utiliza para convertir todo carácter en mayúscula. Luego procedemos a ejecutar una serie de instrucciones, para limpiar la cadena de caracteres de cualquier posible basura.

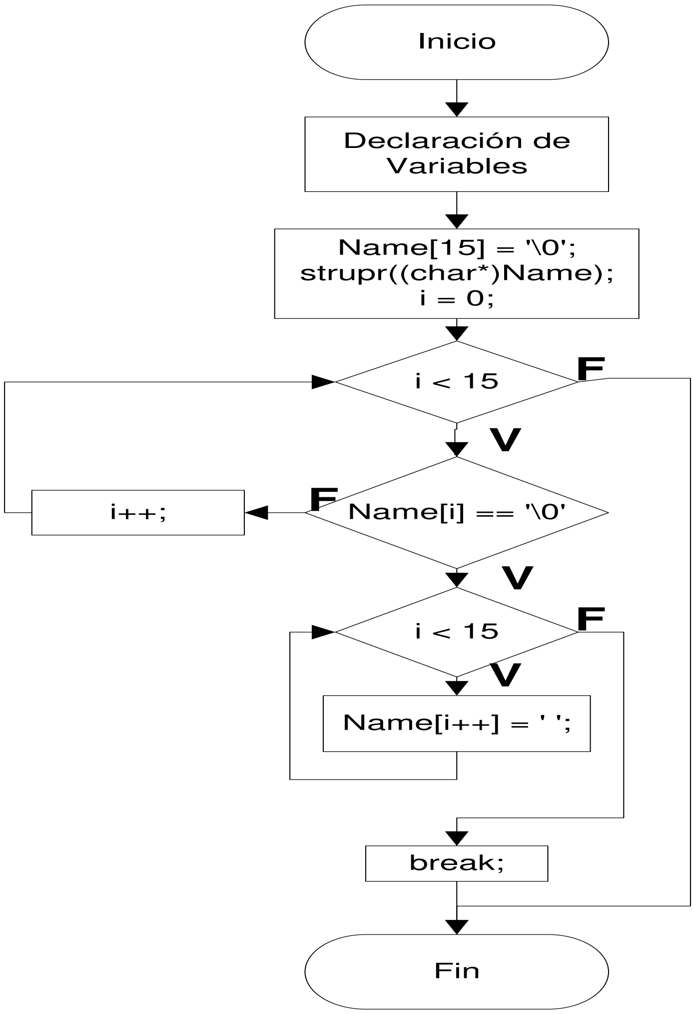


Figura 4.8 Diagrama de flujo de la función Formato_NetBIOS_Nombre

La figura 4.9, muestra el diagrama de bloques de la función 'Mostrar_Valor_IP', que nos permite imprimir en el LCD la dirección IP actual del la aplicación. Como en todas las funciones, empezamos por la declaración e inicialización de las variables. Como la dirección IP es un tipo de dato entero y lo que necesitamos es tener a la dirección IP como texto, procedemos a la conversión con la ayuda de la función ‘uitoa’. Luego con una serie de instrucciones procedemos a imprimir por medio del modulo EUSART la dirección IP en el LCD.

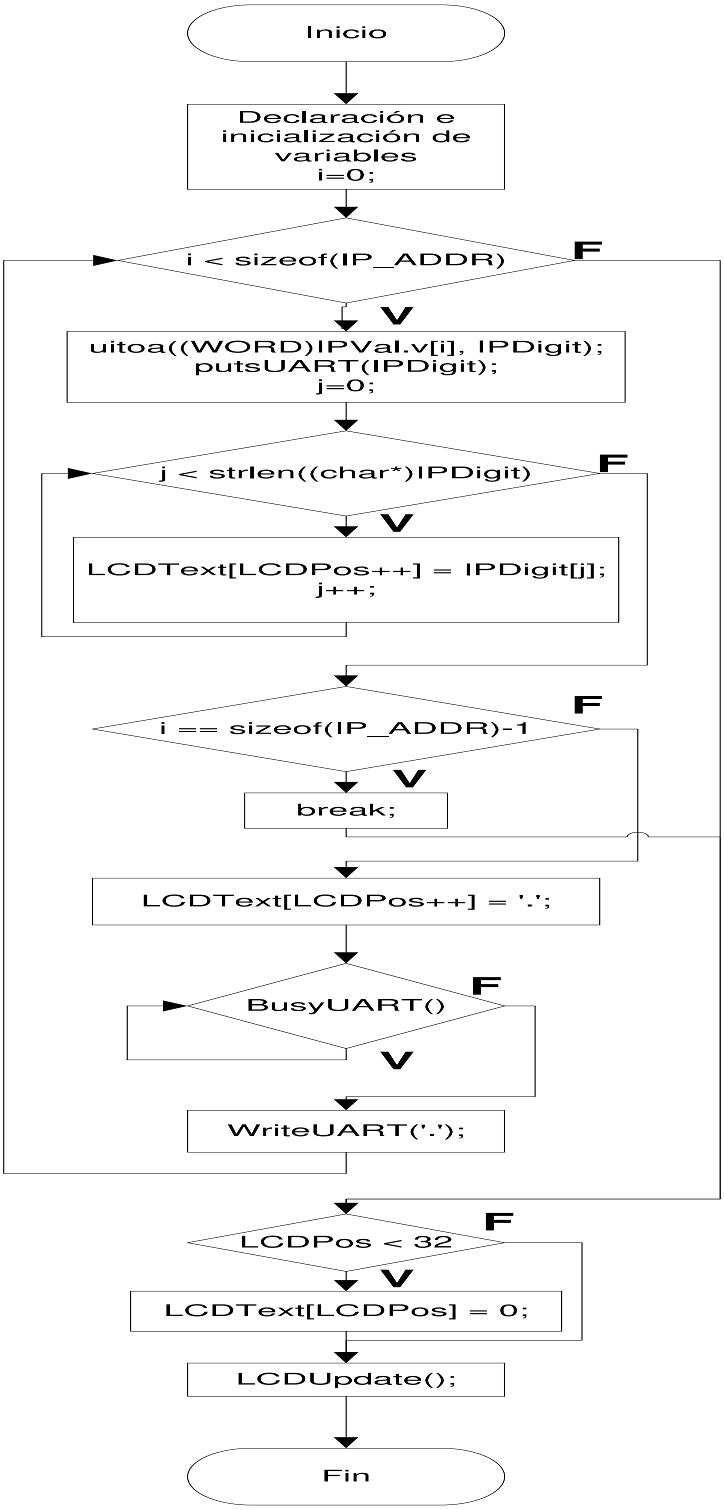


Figura 4. 9 Diagrama de flujo de la función Mostrar_Valor_IP

En nuestro programa utilizamos directivas tales como ‘#pragma interruplow ISR_baja’ y ‘#pragma interrup ISR_alta’, las cuales definen rutinas del servicio de interrupción (ISR). La primera de prioridad baja y la segunda de prioridad alta. Estas interrupciones suspenden la ejecución de nuestra aplicación, guarda la información y transfiere el control a un ISR (Las ISR son funciones como cualquier otra, pero con las restricciones de que no pueden devolver ni aceptar parámetros o no se pueden invocar desde otros puntos del programa), entonces el evento será procesado. Para nuestra prioridad alta (la interrupción de mayor prioridad), se activa cuando la bandera o bit RCIF del registro PIR1 se pone en ‘1’, despues el programa ejecuta la función ‘UART2TCPBridgeISR’. El algoritmo de dicha funcion se muestra en la siguiente figura 4.10, la cual empezamos por declarar una variable local ‘i’, luego realizamos 2 rutinas de instrucciones, que nos ayuda a la recepción y transmisión de los bytes. Estas rutinas se ejecutan siempre y cuando los bits RCIF o TXIF del registro PIR1 se encuentran activados. Una vez realizada la recepción o transmisión se procede a poner en ‘0’ los bits RCIF o TXIF. Cabe recalcar que cualquier bit que representa una bandera de interrupción se debe poner a ‘0’ desde el software.

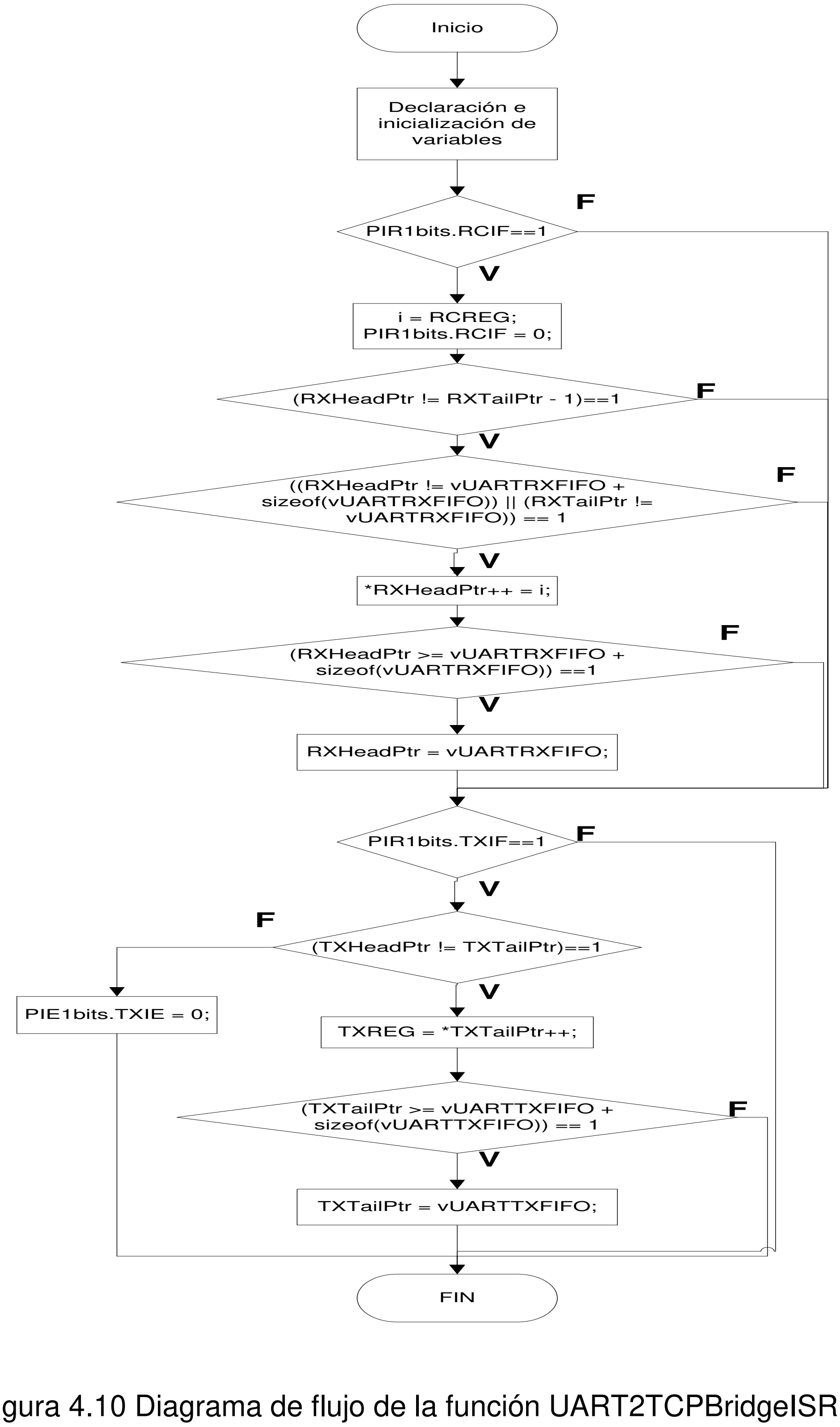


Figura 4.10 Diagrama de flujo de la función UART2TCPBridgeISR

Para nuestra prioridad baja (la interrupción de menor prioridad), se activa cuando la bandera o bit TMR0IF del registro INTCON se pone en ‘1’, despues el programa ejecuta la función ‘TickUpdate’. El diagrama de bloques del funcionamiento de esta función, se muestra el la figura 4.11, la cual empieza realizando una consulta al bit TMR0IF para saber si se ha activado, si esto ocurre se procede a incrementar el contador interno alto, para finalizar se pone en ‘0’ el bit TMRIF.

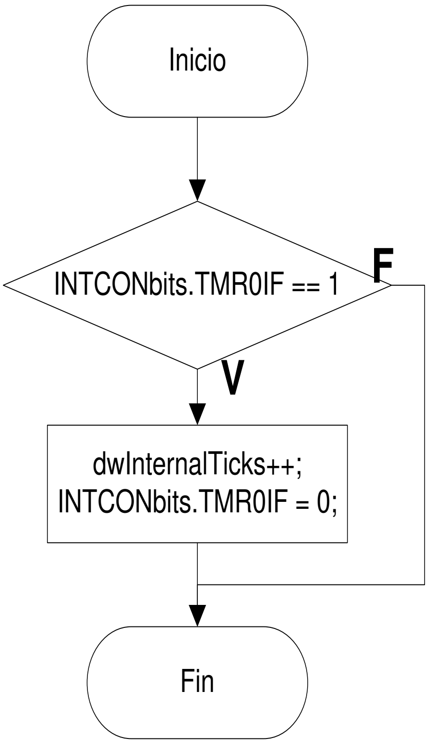


Figura 4.11 Diagrama de flujo de la función TickUpdate

4.2.1 Estructuras

Con el objetivo de facilitar la manipulación de nuestra programación, hemos visto conveniente el uso de estructuras como una forma de organizar un conjunto de datos que son elementales. En esta sección hemos puesto las más importantes estructuras de datos, que serán explicadas a continuación.

En el primer tipo de datos se almacenará la información necesaria, para el correcto funcionamiento del protocolo DHCP, esta se encuentra localizado en el archivo DHCP.h del stack TCP/IP.

```
typedef struct __attribute__((aligned(2), packed))

{

BYTE MessageType;

BYTE HardwareType;

BYTE HardwareLen;

BYTE Hops;

DWORD  TransactionID;
```

```
WORD SecondsElapsed;

WORD BootpFlags;

IP_ADDR ClientIP;

IP_ADDR YourIP;

IP_ADDR NextServerIP;

IP_ADDR RelayAgentIP;

MAC_ADDR ClientMAC;

} BOOTP_HEADER;

typedef enum _SM_DHCP

{

SM_DHCP_DISABLED = 0,

SM_DHCP_GET_SOCKET,

SM_DHCP_SEND_DISCOVERY,

SM_DHCP_GET_OFFER,

SM_DHCP_SEND_REQUEST,

SM_DHCP_GET_REQUEST_ACK,
```

```
SM_DHCP_BOUND,

SM_DHCP_SEND_RENEW,

SM_DHCP_GET_RENEW_ACK,

SM_DHCP_SEND_RENEW2,

SM_DHCP_GET_RENEW_ACK2,

SM_DHCP_SEND_RENEW3,

SM_DHCP_GET_RENEW_ACK3

} SM_DHCP;

typedef union _DHCP_CLIENT_FLAGS

{

    struct

    {

        unsigned char bIsBound : 1;

        unsigned char bOfferReceived : 1;

 unsigned char bDHCPServerDetected : 1;

    } bits;
```

```
   BYTE Val;
} DHCP_CLIENT_FLAGS;
```

Esta estructura se encarga de almacenar la información para el módulo del transceptor de la capa física PHY, así como de las tramas tanto transmisión y recepción con la red. Esta estructura se encuentra localizado en el archivo ETH97J60.h del stack TCP/IP.

```
typedef union {
BYTE v[7];
struct {
 WORD  ByteCount;
 unsigned CollisionCount:4;
 unsigned CRCError:1;
 unsigned LengthCheckError:1;
 unsigned LengthOutOfRange:1;
 unsigned Done:1;
 unsigned Multicast:1;
```

```
 unsigned Broadcast:1;

 unsigned PacketDefer:1;

 unsigned ExcessiveDefer:1;

 unsigned MaximumCollisions:1;

 unsigned LateCollision:1;

 unsigned Giant:1;

 unsigned Underrun:1;

 WORD  BytesTransmittedOnWire;

 unsigned ControlFrame:1;

 unsigned PAUSEControlFrame:1;

 unsigned BackpressureApplied:1;

 unsigned VLANTaggedFrame:1;

 unsigned Zeros:4;

} bits;

} TXSTATUS;

typedef union {
```

```
BYTE v[4];
struct {
 WORD  ByteCount;
 unsigned PreviouslyIgnored:1;
 unsigned RXDCPreviouslySeen:1;
 unsigned CarrierPreviouslySeen:1;
 unsigned CodeViolation:1;
 unsigned CRCError:1;
 unsigned LengthCheckError:1;
 unsigned LengthOutOfRange:1;
 unsigned ReceiveOk:1;
 unsigned Multicast:1;
 unsigned Broadcast:1;
 unsigned DribbleNibble:1;
 unsigned ControlFrame:1;
 unsigned PauseControlFrame:1;
```

```
 unsigned UnsupportedOpcode:1;

 unsigned VLANType:1;

 unsigned Zero:1;

} bits;

} RXSTATUS;

typedef union {

WORD Val;

WORD_VAL VAL;

// PHCON1 bits ----------

struct {

 unsigned :8;

 unsigned PDPXMD:1;

 unsigned :7;

} PHCON1bits;

// PHSTAT1 bits --------

struct {
```

```
 unsigned :2;

 unsigned LLSTAT:1;

 unsigned :5;

 unsigned :8;

} PHSTAT1bits;

// PHCON2 bits ----------

struct {

 unsigned :4;

 unsigned RXAPDIS:1;

 unsigned :3;

 unsigned HDLDIS:1;

 unsigned :5;

 unsigned FRCLNK:1;

 unsigned :1;

} PHCON2bits;

// PHSTAT2 bits --------
```

```
struct {
 unsigned :8;
 unsigned :2;
 unsigned LSTAT:1;
 unsigned COLSTAT:1;
 unsigned RXSTAT:1;
 unsigned TXSTAT:1;
 unsigned :2;
} PHSTAT2bits;
// PHIE bits -----------
struct {
 unsigned :1;
 unsigned PGEIE:1;
 unsigned :2;
 unsigned PLNKIE:1;
 unsigned :3;
```

```
 unsigned :8;

} PHIEbits;

// PHIR bits -----------

struct {

 unsigned :2;

 unsigned PGIF:1;

 unsigned :1;

 unsigned PLNKIF:1;

 unsigned :3;

 unsigned :8;

} PHIRbits;

// PHLCON bits -------

struct {

 unsigned :1;

 unsigned STRCH:1;

 unsigned LFRQ0:1;
```

```
 unsigned LFRQ1:1;

 unsigned LBCFG0:1;

 unsigned LBCFG1:1;

 unsigned LBCFG2:1;

 unsigned LBCFG3:1;

 unsigned LACFG0:1;

 unsigned LACFG1:1;

 unsigned LACFG2:1;

 unsigned LACFG3:1;

 unsigned :4;

} PHLCONbits;

struct {

 unsigned :1;

 unsigned STRCH:1;

 unsigned LFRQ:2;

 unsigned LBCFG:4;
```

```
 unsigned LACFG:4;
 unsigned :4;
} PHLCONbits2;
} PHYREG;
```

Como podemos observar, esta estructura de datos se la utiliza para renombrar las variables que nos ofrece el compilador como variables tipo: char, int, long, etc, a variables conocidas en la transmisión de datos como del tipo: BOOL, BYTE, WORD, DWORD,QWORD, etc. Esto se encuentra localizado en el archivo GENERICTYPEDEFS.h del stack TCP/IP.

```
typedef enum _BOOL { FALSE = 0, TRUE } BOOL;
typedef unsigned char BYTE;
typedef unsigned short int WORD;
typedef unsigned long DWORD;
typedef unsigned long long QWORD;
typedef signed char CHAR;
typedef signed short int SHORT;
```

```
typedef signed long  LONG;

typedef signed long long LONGLONG;

typedef union _BYTE_VAL

{

    BYTE Val;

    struct

    {

        unsigned char b0:1;

        unsigned char b1:1;

        unsigned char b2:1;

        unsigned char b3:1;

        unsigned char b4:1;

        unsigned char b5:1;

        unsigned char b6:1;

        unsigned char b7:1;

    } bits;
```

```
} BYTE_VAL;

typedef union _CHAR_VAL

{

    CHAR Val;

    struct

    {

        unsigned char b0:1;

        unsigned char b1:1;

        unsigned char b2:1;

        unsigned char b3:1;

        unsigned char b4:1;

        unsigned char b5:1;

        unsigned char b6:1;

        unsigned char b7:1;

    } bits;

} CHAR_VAL;
```

```
typedef union _WORD_VAL

{

    WORD Val;

    BYTE v[2];

    struct

    {

        BYTE LB;

        BYTE HB;

    } byte;

    struct

    {

        unsigned char b0:1;

        unsigned char b1:1;

        unsigned char b2:1;

        unsigned char b3:1;

        unsigned char b4:1;
```

```
        unsigned char b5:1;
        unsigned char b6:1;
        unsigned char b7:1;
        unsigned char b8:1;
        unsigned char b9:1;
        unsigned char b10:1;
        unsigned char b11:1;
        unsigned char b12:1;
        unsigned char b13:1;
        unsigned char b14:1;
        unsigned char b15:1;
    } bits;
} WORD_VAL;
typedef union _SHORT_VAL
{
    SHORT Val;
```

```
BYTE  v[2];

struct

{

   BYTE LB;

   BYTE HB;

} byte;

struct

{

   unsigned char b0:1;

   unsigned char b1:1;

   unsigned char b2:1;

   unsigned char b3:1;

   unsigned char b4:1;

   unsigned char b5:1;

   unsigned char b6:1;

   unsigned char b7:1;
```

```
        unsigned char b8:1;

        unsigned char b9:1;

        unsigned char b10:1;

        unsigned char b11:1;

        unsigned char b12:1;

        unsigned char b13:1;

        unsigned char b14:1;

        unsigned char b15:1;

    } bits;

} SHORT_VAL;

typedef union _DWORD_VAL

{

    DWORD Val;

WORD w[2];

    BYTE v[4];

    struct
```

```
{
    WORD LW;
    WORD HW;
} word;
struct
{
    BYTE LB;
    BYTE HB;
    BYTE UB;
    BYTE MB;
} byte;
struct
{
    unsigned char b0:1;
    unsigned char b1:1;
    unsigned char b2:1;
```

```
unsigned char b3:1;
unsigned char b4:1;
unsigned char b5:1;
unsigned char b6:1;
unsigned char b7:1;
unsigned char b8:1;
unsigned char b9:1;
unsigned char b10:1;
unsigned char b11:1;
unsigned char b12:1;
unsigned char b13:1;
unsigned char b14:1;
unsigned char b15:1;
unsigned char b16:1;
unsigned char b17:1;
unsigned char b18:1;
```

```
        unsigned char b19:1;

        unsigned char b20:1;

        unsigned char b21:1;

        unsigned char b22:1;

        unsigned char b23:1;

        unsigned char b24:1;

        unsigned char b25:1;

        unsigned char b26:1;

        unsigned char b27:1;

        unsigned char b28:1;

        unsigned char b29:1;

        unsigned char b30:1;

        unsigned char b31:1;

    } bits;

} DWORD_VAL;

typedef union _LONG_VAL
```

```
{
    LONG Val;
WORD w[2];
    BYTE v[4];
    struct
    {
        WORD LW;
        WORD HW;
    } word;
    struct
    {
        BYTE LB;
        BYTE HB;
        BYTE UB;
        BYTE MB;
    } byte;
```

```
struct
{
    unsigned char b0:1;
    unsigned char b1:1;
    unsigned char b2:1;
    unsigned char b3:1;
    unsigned char b4:1;
    unsigned char b5:1;
    unsigned char b6:1;
    unsigned char b7:1;
    unsigned char b8:1;
    unsigned char b9:1;
    unsigned char b10:1;
    unsigned char b11:1;
    unsigned char b12:1;
    unsigned char b13:1;
```

```
unsigned char b14:1;
unsigned char b15:1;
unsigned char b16:1;
unsigned char b17:1;
unsigned char b18:1;
unsigned char b19:1;
unsigned char b20:1;
unsigned char b21:1;
unsigned char b22:1;
unsigned char b23:1;
unsigned char b24:1;
unsigned char b25:1;
unsigned char b26:1;
unsigned char b27:1;
unsigned char b28:1;
unsigned char b29:1;
```

```
        unsigned char b30:1;

        unsigned char b31:1;

    } bits;

} LONG_VAL;

typedef union _QWORD_VAL

{

    QWORD Val;

          DWORD d[2];

WORD w[4];

    BYTE v[8];

    struct

    {

        DWORD LD;

        DWORD HD;

    } dword;

    struct
```

```
{
    WORD LW;
    WORD HW;
    WORD UW;
    WORD MW;
} word;
struct
{
    unsigned char b0:1;
    unsigned char b1:1;
    unsigned char b2:1;
    unsigned char b3:1;
    unsigned char b4:1;
    unsigned char b5:1;
    unsigned char b6:1;
    unsigned char b7:1;
```

```
        unsigned char b8:1;
        unsigned char b9:1;
        unsigned char b10:1;
        unsigned char b11:1;
        unsigned char b12:1;
        unsigned char b13:1;
        unsigned char b14:1;
        unsigned char b15:1;
        unsigned char b16:1;
        unsigned char b17:1;
        unsigned char b18:1;
        unsigned char b19:1;
        unsigned char b20:1;
        unsigned char b21:1;
        unsigned char b22:1;
        unsigned char b23:1;
```

```
unsigned char b24:1;
unsigned char b25:1;
unsigned char b26:1;
unsigned char b27:1;
unsigned char b28:1;
unsigned char b29:1;
unsigned char b30:1;
unsigned char b31:1;
unsigned char b32:1;
unsigned char b33:1;
unsigned char b34:1;
unsigned char b35:1;
unsigned char b36:1;
unsigned char b37:1;
unsigned char b38:1;
unsigned char b39:1;
```

```
unsigned char b40:1;
unsigned char b41:1;
unsigned char b42:1;
unsigned char b43:1;
unsigned char b44:1;
unsigned char b45:1;
unsigned char b46:1;
unsigned char b47:1;
unsigned char b48:1;
unsigned char b49:1;
unsigned char b50:1;
unsigned char b51:1;
unsigned char b52:1;
unsigned char b53:1;
unsigned char b54:1;
unsigned char b55:1;
```

```
    unsigned char b56:1;
    unsigned char b57:1;
    unsigned char b58:1;
    unsigned char b59:1;
    unsigned char b60:1;
    unsigned char b61:1;
    unsigned char b62:1;
    unsigned char b63:1;
  } bits;
} QWORD_VAL;
```

En esta estructura, se colocará la información necesaria para el correcto funcionamiento del protocolo HTTP; así como, un indicador de estado de dicho protocolo, esto se encuentra localizado en el archivo HTTP2.h del stack TCP/IP.

```
typedef enum _HTTP_STATUS
{
 HTTP_GET = 0u,
```

```
 HTTP_POST,

HTTP_UNAUTHORIZED,

    HTTP_NOT_FOUND,

                    HTTP_OVERFLOW,

                    HTTP_INTERNAL_SERVER_ERROR,

HTTP_NOT_IMPLEMENTED,

#if defined(HTTP_MPFS_UPLOAD)

HTTP_MPFS_FORM,

HTTP_MPFS_UP,

HTTP_MPFS_OK,

HTTP_MPFS_ERROR,

#endif

HTTP_REDIRECT,

HTTP_SSL_REQUIRED

} HTTP_STATUS;

typedef enum _SM_HTTP2
```

```
{
 SM_HTTP_IDLE = 0u,
 SM_HTTP_PARSE_REQUEST,
 SM_HTTP_PARSE_HEADERS,
 SM_HTTP_AUTHENTICATE,
 SM_HTTP_PROCESS_GET,
 SM_HTTP_PROCESS_POST,
 SM_HTTP_PROCESS_REQUEST,
 SM_HTTP_SERVE_HEADERS,
 SM_HTTP_SERVE_COOKIES,
 SM_HTTP_SERVE_BODY,
 SM_HTTP_SEND_FROM_CALLBACK,
 SM_HTTP_DISCONNECT,
 SM_HTTP_WAIT
} SM_HTTP2;
typedef enum _HTTP_IO_RESULT
```

```
{
 HTTP_IO_DONE = 0u,
 HTTP_IO_NEED_DATA,
 HTTP_IO_WAITING,
} HTTP_IO_RESULT;
typedef enum _HTTP_FILE_TYPE
{
 HTTP_TXT = 0u,
 HTTP_HTM,
 HTTP_HTML,
 HTTP_CGI,
 HTTP_XML,
 HTTP_CSS,
 HTTP_GIF,
 HTTP_PNG,
 HTTP_JPG,
```

```
 HTTP_JAVA,

 HTTP_WAV,

 HTTP_UNKNOWN

} HTTP_FILE_TYPE;

typedef struct _HTTP_STUB

{

    SM_HTTP2 sm;

    TCP_SOCKET socket;

} HTTP_STUB;

typedef struct _HTTP_CONN

{

 DWORD byteCount;

 DWORD nextCallback;

                        DWORD callbackID;

 DWORD callbackPos;

                        BYTE *ptrData;
```

```
                    BYTE *ptrRead;

                    MPFS_HANDLE file;

                    MPFS_HANDLE offsets;

 BYTE hasArgs;

                    BYTE isAuthorized;

 HTTP_STATUS httpStatus;

                    HTTP_FILE_TYPE fileType;

 BYTE data[HTTP_MAX_DATA_LEN];

} HTTP_CONN;
```

En este tipo de información se definen las diferentes partes de la trama IP como dirección IP destino, dirección IP fuente, sumatoria, bandera, etc; que serán usadas cuando el programa requiera del protocolo IP, para el encapsulamiento o desencapsulamiento de los datos al momento de la transmisión y recepción de los mismos. Esto se encuentra localizado en el archivo IP.h del stack TCP/IP.

```
typedef struct _IP_HEADER

{
```

```
    BYTE    VersionIHL;

    BYTE    TypeOfService;

    WORD    TotalLength;

    WORD    Identification;

    WORD    FragmentInfo;

    BYTE    TimeToLive;

    BYTE    Protocol;

    WORD    HeaderChecksum;

    IP_ADDR SourceAddress;

    IP_ADDR DestAddress;

} IP_HEADER;



typedef struct _PSEUDO_HEADER

{

    IP_ADDR SourceAddress;

    IP_ADDR DestAddress;
```

```
   BYTE Zero;

   BYTE Protocol;

   WORD Length;

} PSEUDO_HEADER;
```

Como podemos apreciar en esta estructura se definen las variables que se utilizarán para el direccionamiento de control de acceso al medio. Esto se encuentra localizado en el archivo MAC.h del stack TCP/IP.

```
typedef struct  __attribute__((aligned(2), packed))

{

MAC_ADDR        DestMACAddr;

MAC_ADDR        SourceMACAddr;

WORD_VAL        Type;

} ETHER_HEADER;
```

En esta estructura se definen las variables que utilizaremos para guardar la configuración de red, necesaria para la conexión entre nuestro sistema y la red externa. Esto se encuentra localizado en StackTsk.h del stack TCPIP.

```
typedef struct __attribute__((__packed__)) _MAC_ADDR

{

   BYTE v[6];

} MAC_ADDR;

#define IP_ADDR  DWORD_VAL

typedef struct __attribute__((__packed__)) _NODE_INFO

{

   IP_ADDR     IPAddr;

   MAC_ADDR    MACAddr;

} NODE_INFO;



typedef struct __attribute__((__packed__)) _APP_CONFIG

{

IP_ADDR  MyIPAddr;

IP_ADDR  MyMask;

IP_ADDR  MyGateway;
```

```
IP_ADDR PrimaryDNSServer;

IP_ADDR SecondaryDNSServer;

IP_ADDR DefaultIPAddr;

IP_ADDR DefaultMask;

BYTE NetBIOSName[16];

struct

{

 unsigned char : 6;

 unsigned char bIsDHCPEnabled : 1;

 unsigned char bInConfigMode : 1;

} Flags;

MAC_ADDR MyMACAddr;

} APP_CONFIG;
```

El siguiente tipo de datos se utiliza para la transportación de los paquetes de forma segura por la red de internet, es decir, es necesario cuando el programa utilice el protocolo TCP. A más de los estados en los enlaces que se establecen durante la comunicación. Esto se encuentra localizado en el archivo TCP.h del stack TCP/IP.

```
typedef enum _TCP_STATE
{
    TCP_LOOPBACK = 0,
    TCP_LOOPBACK_CLOSED,
          TCP_GET_DNS_MODULE,
TCP_DNS_RESOLVE,
TCP_GATEWAY_SEND_ARP,
TCP_GATEWAY_GET_ARP,

    TCP_LISTEN,
    TCP_SYN_SENT,
    TCP_SYN_RECEIVED,
    TCP_ESTABLISHED,
    TCP_FIN_WAIT_1,
```

```
   TCP_FIN_WAIT_2,

    TCP_CLOSING,

TCP_CLOSE_WAIT,

    TCP_LAST_ACK,

   TCP_CLOSED

} TCP_STATE;

typedef struct _TCB_STUB

{

PTR_BASE bufferTxStart;

PTR_BASE bufferRxStart;

PTR_BASE bufferEnd;

PTR_BASE txHead;

PTR_BASE txTail;

PTR_BASE rxHead;

PTR_BASE rxTail;

   DWORD eventTime;
```

```
WORD eventTime2;

{

 WORD delayedACKTime;

 WORD closeWaitTime;

} OverlappedTimers;

    TCP_STATE smState;

    struct

    {

         unsigned char bServer : 1;

           unsigned char bTimerEnabled: 1;

unsigned char bTimer2Enabled : 1;

unsigned char bDelayedACKTimerEnabled : 1;

unsigned char bOneSegmentReceived : 1;

unsigned char bHalfFullFlush : 1;

unsigned char bTXASAP : 1;

unsigned char bTXFIN : 1;
```

```
unsigned char bSocketReset : 1;

unsigned char filler : 7;

   } Flags;

   WORD_VAL remoteHash;

    BYTE vMemoryMedium;

} TCB_STUB;

typedef struct _TCB

{

DWORD retryInterval;

DWORD MySEQ;

DWORD RemoteSEQ;

PTR_BASE txUnackedTail;

   WORD_VAL remotePort;

   WORD_VAL localPort;

WORD remoteWindow;

WORD wFutureDataSize;
```

```
union

{

 NODE_INFO niRemoteMACIP;

 DWORD dwRemoteHost;

} remote;

SHORT sHoleSize;

    struct

    {

          unsigned char bFINSent : 1;

 unsigned char bSYNSent : 1;

 unsigned char bRemoteHostIsROM : 1;

 unsigned char filler : 5;

    } flags;

    BYTE retryCount;

    BYTE vSocketPurpose;

} TCB;
```

```
typedef struct _SOCKET_INFO

{

NODE_INFO remote;

WORD_VAL remotePort;

} SOCKET_INFO;
```

En esta estructura declaramos las variables necesarias, para la transferencia de archivos en forma simple; es decir, útiles al momento de usar el protocolo TFTP. Esto se encuentra localizado en el archivo TFTPc.h del stack TCP/IP.

```
typedef enum _TFTP_RESULT

{

   TFTP_OK = 0,

   TFTP_NOT_READY,

   TFTP_END_OF_FILE,

   TFTP_ERROR,

   TFTP_RETRY,

   TFTP_TIMEOUT
```

```
} TFTP_RESULT;

typedef enum _TFTP_FILE_MODE

{

   TFTP_FILE_MODE_READ = 1,

   TFTP_FILE_MODE_WRITE = 2

} TFTP_FILE_MODE;

typedef enum _TFTP_ACCESS_ERROR

{

   TFTP_ERROR_NOT_DEFINED = 0,

   TFTP_ERROR_FILE_NOT_FOUND,

   TFTP_ERROR_ACCESS_VIOLATION,

   TFTP_ERROR_DISK_FULL,

   TFTP_ERROR_INVALID_OPERATION,

   TFTP_ERROR_UNKNOWN_TID,

   TFTP_ERROR_FILE_EXISTS,

   TFTP_ERROR_NO_SUCH_USE
```

```
} TFTP_ACCESS_ERROR;
```

Finalmente, en esta estructura se hace referencia al protocolo UDP, declaramos las variables necesarias para el transporte de paquetes por medio de dicho protocolo. Esto se encuentra ubicado en el archivo UDP.h del stack TCP/IP.

```
typedef WORD UDP_PORT;

typedef BYTE UDP_SOCKET;

typedef struct _UDP_SOCKET_INFO

{

   NODE_INFO   remoteNode;

   UDP_PORT    remotePort;

   UDP_PORT    localPort;

} UDP_SOCKET_INFO;

typedef struct _UDP_HEADER

{

   UDP_PORT    SourcePort;

   UDP_PORT    DestinationPort;
```

```
    WORD        Length;

    WORD        Checksum;

} UDP_HEADER;
```

4.2.2 Cabeceras

Debido a que el programa necesita una serie de funciones que se llaman de manera automática unas a otras, para cumplir con una tarea específica, hemos utilizado cabeceras o .h. Las cabeceras, contienen única y exclusivamente los prototipos de las funciones y en algunos casos contienen las estructuras de datos que ya hemos mencionado. En esta sección hemos puesto los principales .h que utilizaremos en el código para un mejor manejo del mismo.

- ARP.h: Funciones que permiten encontrar la dirección hardware o MAC correspondientes a una determinada dirección IP.

- Delay.h: Declaraciones que establecen retardos específicos, que son de utilidad para el buen desempeño del programa.

- DHCP.h: Funciones que permite a las estaciones de trabajo de una red IP obtener sus parámetros de configuración de manera automática.

- ETH97J60.h: En esta cabecera definimos las estructuras útiles para el manejo del PIC18F97J60, así como los todos los registros que posee dicho PIC.

- FTP.h: Funciones que permiten la comunicación tipo FTP entre el usuario y el Microcontrolador.

- funciones.h: Declaraciones que permiten la configuración inicial y posterior del microcontrolador, y de los protocolos TCP/IP que se utilizan en nuestra aplicación.

- GenericTCPServer.h: Función que utiliza el stack periódicamente para escuchar las conexiones entrantes.

- HTTP2.h: Funciones y estructuras que permiten utilizar de forma adecuada el protocolo HTTP.

- ICMP.h: Definiciones que se establecen para el control y notificación de errores del protocolo IP.

- IP.h: Estructuras y funciones del protocolo IP, usadas por el programa, para la comunicación de datos a través de una red de paquetes conmutados.
- MAC.h: Estas funciones proporcionan acceso al controlador Ethernet del PIC18F97J60.
- MPFS.h: Funciones utilizadas para acceder a la página web, y a otros archivos de la memoria de programas internos, y de una memoria serial EEPROM externa.
- Random.h: Función que permite reiniciar el microcontrolador de forma remota.
- SNTP.h: Definiciones que actualizan el tiempo interno del programa cada 10 minutos usando la dirección "pool.ntp.org", que es un servidor de tiempo de red.
- StackTsk.h: Funciones que permiten manejar los paquetes de recepción realizando un pre procesamiento para luego ser enviados a las capas superiores.
- TCP.h: Funciones que proporcionan confiabilidad, ofreciendo transporte en la transferencia de aplicaciones orientadas a datos con control de flujo.

- TCPIP.h: En esta cabecera se encuentran declaradas todos los ‘.h’ del programa, permitiendo acceder a todos las cabeceras mediante una sola llamada.

- TCPPerformance.h: Funciones que realizan únicamente pruebas, para establecer una conexión, enviando imitaciones de paquetes de la memoria ROM.

- Telnet.h: Declaraciones que establecen los servicios de comunicación de telnet.

- TFTPc.h: Funciones que establecen servicios poco confiables de carga y descarga de archivos a otras aplicaciones.

- Tick.h: Funciones útiles para la administración del cronómetro interno del microcontrolador, así como la inicialización de los Relojes (Timers), conversión a escala del tiempo, entre otras.

- UART2TCPBridge.h: Declaraciones útiles que permiten controlar y configurar los periféricos UART para esta aplicación.

- UDP.h: Funciones que establecen un transporte de paquetes de forma rápida pero de poca confiabilidad.

4.2.3 Archivos Fuente

Los archivos fuente que presentamos, fueron creados para la aplicación. Con este código realizamos las llamadas a los registros del PIC18F97J60 y al stack TCP/IP, permitiéndonos hacer las configuraciones respectivas en el PIC. Así, aseguramos un buen desempeño del sistema, tanto en el proceso interno del Microcontrolador, como en las diferentes interfaces que se manejan. Las funciones que permiten llamar, y utilizar las distintas funciones, y estructuras del stack, ayudan a que la tarjeta se pueda comunicar con la red externa o con una PC conectada directamente.

```
/*

 ********************************************************************

 * Archivo:        Main.c

 *******************************************************************/

/*

 * main() es la función principal de nuestro programa.
```

```
 *La función main() es el punto de partida, para una serie
 *       de llamadas a funciones que cumplen con tareas
 *       especificas, que ayuda al PIC a desempeñar nuestra
*         aplicación requerida.
 */
#define THIS_IS_STACK_APPLICATION
// Esta cabecera incluye todas las cabeceras para habilitar
//  cualquier funcion del Stack TCP/IP
#include "Include/TCPIP Stack/TCPIP.h"
// Esto es usado por otros elementos del stack.
// La aplicacion Main debe definir esto e inicializar con los
// propios valores.
APP_CONFIG AppConfig;
BYTE myDHCPBindCount = 0xFF;
#define DHCPBindCount (0xFF)
// Fijar los Bits de configuracion
```

```
// Configuracion de Bits para nuestra aplicacion
#pragma config XINST=OFF, WDT=OFF, FOSC2=ON, FOSC=HSPLL, ETHLED=ON
//
// Servicio de rutinas de interrupción del PIC18
//
#pragma interruptlow LowISR
void LowISR(void)
{
   TickUpdate();
}
#pragma interruptlow HighISR
void HighISR(void)
{
UART2TCPBridgeISR();
}
```

```
#pragma code lowVector=0x18

void LowVector(void){_asm goto LowISR _endasm}

#pragma code highVector=0x8

void HighVector(void){_asm goto HighISR _endasm}

#pragma code

//

// Punto de entrada a la aplicación Main.

//

void main(void)

{

   static TICK t = 0;

BYTE i;

  // Inicializa cualquier hardware especifico de la aplicación.

   Inicializar_Tarjeta();

// Inicializa y muestra la versión del stack sobre el LCD

LCDInit();
```

```
for(i = 0; i < 100; i++)

 DelayMs(1);

strcpypgm2ram((char*)LCDText, "Sandoya-Montene");

LCDUpdate();

   // Inicializa todos los componentes relacionados con el stack.

 // Los siguientes pasos deben ser ejecutados para todas las

// aplicaciones que usan el Stack TCP/IP de Microchip.

   TickInit();

// Inicializa el módulo de sistema del archivo de Microchip

   MPFSInit();

// Inicializa el Stack y aplicaciones relacionadas con las

//  variables NV dentro del AppConfig.

Inicializar_AppConfig();

// Inicia el proceso de la tarjeta si el botón R3 es presionado

// en arranque

   if(BUTTON0_IO == 0u)
```

```
   {
// Anula el contenido de la EEPROM si el BUTTON0 es retenido
por más de 4 segundos
TICK Tiempo_Inicio = TickGet();


 while(BUTTON0_IO == 0u)
 {
                    if(TickGet()     -     Tiempo_Inicio     >
                    4*TICK_SECOND)
  {
     XEEBeginWrite(0x0000);
     XEEWrite(0xFF);
     XEEEndWrite();
   #if defined(STACK_USE_UART)
                      putrsUART("\r\n\r\nRetener  el  BUTTON0
                      por mas de 4 segundos.  Contenido de la
                      EEPROM borrado.\r\n\r\n");
```

```
#endif
 LED0_TRIS = 0;
 LED1_TRIS = 0;
 LED2_TRIS = 0;
 LED3_TRIS = 0;
 LED0_IO = 1;
 LED1_IO = 1;
 LED2_IO = 1;
 LED3_IO = 1;
                    while((LONG)(TickGet()-
                    Tiempo_Inicio)<=(LONG)
                    (9*TICK_SECOND/2));
 Reset();
 break;
}
}
```

```
   Seteo_Config();
}



      // Inicializa el nucleo de las capas del stack (MAC, ARP,
      TCP, UDP)
   StackInit();
   UART2TCPBridgeInit();
   HTTPInit();
   if(!AppConfig.Flags.bIsDHCPEnabled)
   {  // Forza la dirección IP mostrar la actualización.
      myDHCPBindCount = 1;
      DHCPDisable();
   }
// Una vez que todos los puntos son inicializados, va dentro de
// un lazo infinito y permite que los elementos del stack lleven a
// cabo sus tareas. Si las aplicaciones necesitan ejecutar sus
// propias tareas, esto debe ser hecho en el fin del lazo.
```

```
// Note que esto es un mecanismo de cooperación de
// multitareas donde cada tarea ejecuta sus procesos (tanto si
// todo en un intento  o parte de esto) y retorna para que otras
// tareas puedan hacer su trabajo. Si una tarea necesita una
// gran cantidad de tiempo para hacer su trabajo, esto debe ser
// interrumpido  dentro de pequeñas piezas para que otras
// tareas puedan tener tiempo de CPU.
  while(1)
  {
    // Parpadeo del LED0 cada segundo.
    if(TickGet() - t >= TICK_SECOND/2ul)
    {
      t = TickGet();
      LED0_IO ^= 1;
    }


    // Esta labor normal ejecuta stack task incluyendo el
    // chequeo para paquetes entrantes, tipo de paquete y la
```

```
      // llamada apropiada de la entidad del stack para procesar
      // esto.
       StackTask();
       UART2TCPBridgeTask();



      // Esta es una aplicación TCP. Esto escucha al puerto TCP
      // 80 con uno o más zócalos y responde a las peticiones
      // remotas.
        HTTPServer();
          DiscoveryTask();
NBNSTask();
DHCPServerTask();
GenericTCPClient();
GenericTCPServer();
TelnetTask();
RebootTask();
```

```
SNTPClient();



     // Para la información DHCP, muestra cuantas veces
     // nosotros hemos renovado la configuración de la IP desde
     // el ultimo reset.

      if(DHCPBindCount != myDHCPBindCount)

      {

          myDHCPBindCount = DHCPBindCount;

                        putrsUART((ROM char*)"Nueva Direccion
                        IP: ");

   Mostrar_Valor_IP(AppConfig.MyIPAddr);

 putrsUART((ROM char*)"\r\n");

 AnnounceIP();

 }

   }

}

/*
```

```
 ************************************************************************

 * Archivo:        funcion.h

 ***********************************************************************/

/*

 * Declaraciones que permiten la configuración  inicial y

 * posterior del Microcontrolador, y de los protocolos TCP/IP

 * que utilizamos en nuestra aplicación.

 */

#ifndef __FUNCIONES_H

#define __FUNCIONES_H

// Esto puede o no estar presente en todas las aplicaciones.

void Mostrar_Valor_IP(IP_ADDR IPVal);

void Inicializar Tarjeta(void);

void Inicializar_AppConfig(void);

void Formato_NetBIOS_Nombre(BYTE Nombre[16]);

void Grabar_AppConfig(void);
```

```
void Seteo_Config(void);

#endif

/*

 *********************************************************************

 * Archivo:       funcion.c

********************************************************************/

/*

 * Estas funciones nos permiten cumplir con tareas
 *específicas, utilizando los protocolos TCP/IP y los            *
 registros del Microcontrolador. Con estas funciones        *
 podemos realizar las configuraciones iniciales de la       *
 tarjeta, inicializar el tipo de dato AppConfig, o mostrar la    *
 dirección IP en el LCD, entre otras.

*/

#include "Include/TCPIP Stack/TCPIP.h"

BYTE AN0String[8];

#define BAUD_RATE       (19200) // bps
```

```
/*****************************************************************
 * Función:    void DisplayIPValue(IP_ADDR IPVal)
 * Nota:       Función que permite imprimir en el LCD la
 *    dirección IP actual de la tarjeta.
 *****************************************************************/
void Mostrar_Valor_IP(IP_ADDR IPVal)
{
   BYTE IPDigit[4];
BYTE i;
BYTE j;
BYTE LCDPos=16;
for(i = 0; i < sizeof(IP_ADDR); i++)
{
   uitoa((WORD)IPVal.v[i], IPDigit);
 putsUART(IPDigit);
 for(j = 0; j < strlen((char*)IPDigit); j++)
```

```
 {
  LCDText[LCDPos++] = IPDigit[j];
 }
 if(i == sizeof(IP_ADDR)-1)
  break;
 LCDText[LCDPos++] = '.';
 while(BusyUART());
 WriteUART('.');
}
if(LCDPos < 32)
 LCDText[LCDPos] = 0;
LCDUpdate();
}
/*********************************************************************
 * Función:    void InitializeBoard(void)
 * Nota:       Función que permite inicializar la tarjeta.
```

```
*****************************************************************************/

void Inicializar_Tarjeta(void)

{

// LEDs

LED0_TRIS = 0;

LED1_TRIS = 0;

LED2_TRIS = 0;

LED3_TRIS = 0;

LED4_TRIS = 0;

LED5_TRIS = 0;

LED6_TRIS = 0;

LED7_TRIS = 0;

LED0_IO = 0;

LED1_IO = 0;

LED2_IO = 0;

LED3_IO = 0;
```

```
LED4_IO = 0;
LED5_IO = 0;
LED6_IO = 0;
LED7_IO = 0;
// Habilita 4x/5x PLL sobre el PIC18F97J60.
   OSCTUNE = 0x40;
// Habilita los pull-ups internos del PORTB
   INTCON2bits.RBPU = 0;
// Configura el USART
   TXSTA = 0x20;   // Habilita la Tx y el modo asincrónico en el
                   // modulo de EUSART
   RCSTA = 0x90;   // Habilita el puerto serial y el receptor en el
                   // modulo EUSART
// Nosotros podemos usar la configuración de tasa alta de
// baudios
SPBRG = (INSTR_FREQ+2*BAUD_RATE)/BAUD_RATE/4 - 1;
```

```
TXSTAbits.BRGH = 1; //Alta velocidad en la tasa de
// Baudios.

        // Habilita las Interrupciones

           RCONbits.IPEN = 1;            // Habilita las prioridades de
                                         // interrupción

           INTCONbits.GIEH = 1;  // Habilita todas las interrupciones de
                                 // prioridad alta.

           INTCONbits.GIEL = 1;  // Habilita todas las interrupciones de
                                 // prioridad baja.

        }

        /*******************************************************************

         * Función: void InitAppConfig(void)

          * Nota: Esta función nos permite inicializar la variable

         *   AppConfig.

        *******************************************************************/

        void Inicializar_AppConfig(void)

        {
```

```
   BYTE c;

   BYTE *p;

AppConfig.Flags.bIsDHCPEnabled = TRUE;

AppConfig.Flags.bInConfigMode = TRUE;

AppConfig.MyMACAddr.v[0]=MY_DEFAULT_MAC_BYTE1;

AppConfig.MyMACAddr.v[1] = MY_DEFAULT_MAC_BYTE2;

AppConfig.MyMACAddr.v[2] = MY_DEFAULT_MAC_BYTE3;

AppConfig.MyMACAddr.v[3] = MY_DEFAULT_MAC_BYTE4;

AppConfig.MyMACAddr.v[4] = MY_DEFAULT_MAC_BYTE5;

AppConfig.MyMACAddr.v[5] = MY_DEFAULT_MAC_BYTE6;

AppConfig.MyIPAddr.Val = MY_DEFAULT_IP_ADDR_BYTE1 |
MY_DEFAULT_IP_ADDR_BYTE2<<8ul                     |
MY_DEFAULT_IP_ADDR_BYTE3<<16ul                    |
MY_DEFAULT_IP_ADDR_BYTE4<<24ul;

AppConfig.DefaultIPAddr.Val = AppConfig.MyIPAddr.Val;

AppConfig.MyMask.Val   =   MY_DEFAULT_MASK_BYTE1  |
MY_DEFAULT_MASK_BYTE2<<8ul                        |
```

```
MY_DEFAULT_MASK_BYTE3<<16ul |
MY_DEFAULT_MASK_BYTE4<<24ul;

AppConfig.DefaultMask.Val = AppConfig.MyMask.Val;

AppConfig.MyGateway.Val = MY_DEFAULT_GATE_BYTE1 |
MY_DEFAULT_GATE_BYTE2<<8ul |
MY_DEFAULT_GATE_BYTE3<<16ul |
MY_DEFAULT_GATE_BYTE4<<24ul;

AppConfig.PrimaryDNSServer.Val =
MY_DEFAULT_PRIMARY_DNS_BYTE1 |
MY_DEFAULT_PRIMARY_DNS_BYTE2<<8ul |
MY_DEFAULT_PRIMARY_DNS_BYTE3<<16ul |
MY_DEFAULT_PRIMARY_DNS_BYTE4<<24ul;

AppConfig.SecondaryDNSServer.Val =
MY_DEFAULT_SECONDARY_DNS_BYTE1 |
MY_DEFAULT_SECONDARY_DNS_BYTE2<<8ul |
MY_DEFAULT_SECONDARY_DNS_BYTE3<<16ul |
MY_DEFAULT_SECONDARY_DNS_BYTE4<<24ul;



// Carga el NetBIOS del nombre de host
```

```
memcpypgm2ram(AppConfig.NetBIOSName,(ROMvoid*)MY_D
EFAULT_HOST_NAME, 16);
    Formato_NetBIOS_Nombre(AppConfig.NetBIOSName);
    p = (BYTE*)&AppConfig;
    XEEBeginRead(0x0000);
    c = XEERead();
    XEEEndRead();
 // Cuando un registro es grabado, los primeros byte están
escritos
// como 0x60 para indicar que un registro válido fue grabado.
    if(c == 0x60u)
    {
        XEEBeginRead(0x0001);
        for ( c = 0; c < sizeof(AppConfig); c++ )
            *p++ = XEERead();
        XEEEndRead();
```

```
    }
    else
        Grabar_AppConfig();
}
void Grabar_AppConfig(void)
{
    BYTE c;
    BYTE *p;
    p = (BYTE*)&AppConfig;
    XEEBeginWrite(0x0000);
    XEEWrite(0x60);
    for ( c = 0; c < sizeof(AppConfig); c++ )
    {
        XEEWrite(*p++);
    }
```

```
    XEEEndWrite();

}

#define MAX_USER_RESPONSE_LEN   (20u)

void Seteo_Config(void)

{

    BYTE response[MAX_USER_RESPONSE_LEN];

    IP_ADDR tempIPValue;

    IP_ADDR *destIPValue;

    WORD_VAL wvTemp;

    BOOL bQuit = FALSE;

while(!bQuit)

{

 // Display the menu

              putrsUART("\r\n\r\n\rAplicacion de Configuracion
              TCP/IP (" __DATE__ ")\r\n\r\n");

  putrsUART("\r\n\t1: Cambiar nombre de host:\t\t\t");
```

```
putsUART(AppConfig.NetBIOSName);

putrsUART("\r\n\t2: Cambiar direccion IP estatica:\t\t");

Mostrar_Valor_IP(AppConfig.MyIPAddr);

        putrsUART("\r\n\t3: Cambiar direccion de puerta de
        enlace estatica:\t");

Mostrar_Valor_IP(AppConfig.MyGateway);

 putrsUART("\r\n\t4: Cambiar la mascara de subred:\t\t");

Mostrar_Valor_IP(AppConfig.MyMask);

        putrsUART("\r\n\t5: Cambiar la direccion del servidor
        DNS primario:\t");

Mostrar_Valor_IP(AppConfig.PrimaryDNSServer);

        putrsUART("\r\n\t6: Cambiar la direccion del servidor
        DNS secundario:\t");

Mostrar_Valor_IP(AppConfig.SecondaryDNSServer);

putrsUART("\r\n\t7: ");

        putrsUART((ROMBYTE*)(AppConfig.Flags.bIsDHCPEna
        bled ? "Des" : ""));
```

```
        putrsUART("habilitado DHCP \t\tDHCP esta actualmente
        ");
        putrsUART((ROMBYTE*)(AppConfig.Flags.bIsDHCPEna
        bled ? "habilitado" : "deshabilitado"));
putrsUART("\r\n\t0: Grabar & Quitar.");
putrsUART("\r\nIntroduzca una opción del menu: ");
// Espera por el usuario hasta que presione una tecla
 while(!DataRdyUART());
putrsUART((ROM char*)"\r\n");
// Ejecuta la seleccion del usuario
   switch(ReadUART())
   {
   case '1':
  putrsUART("New setting: ");
                    ReadStringUART(response, sizeof(response)
                    > sizeof(AppConfig.NetBIOSName) ?
```

```
                    sizeof(AppConfig.NetBIOSName)                :
                    sizeof(response));
if(response[0] != '\0')
{
                        memcpy(AppConfig.NetBIOSName,
                        (void*)response,
                        sizeof(AppConfig.NetBIOSName));
   Formato_NetBIOS_Nombre(AppConfig.NetBIOSName);
}
break;
                case '2':
     destIPValue = &AppConfig.MyIPAddr;
     goto ReadIPConfig;
 case '3':
     destIPValue = &AppConfig.MyGateway;
     goto ReadIPConfig;
  case '4':
```

```
            destIPValue = &AppConfig.MyMask;

            goto ReadIPConfig;

         case '5':

            destIPValue=&AppConfig.PrimaryDNSServer;

            goto ReadIPConfig;

              case '6':

                             destIPValue=&

                             AppConfig.SecondaryDNSServer;

            goto ReadIPConfig;

ReadIPConfig:

 putrsUART("Nueva Configuracion: ");

                             ReadStringUART(response,

                             sizeof(response));

                             if(StringToIPAddress(response,

                             &tempIPValue))

           destIPValue->Val = tempIPValue.Val;

 else
```

```
            putrsUART("Entrada Invalida.\r\n");

          break;

        case '7':

                          AppConfig.Flags.bIsDHCPEnabled=

                          !AppConfig.Flags.bIsDHCPEnabled;

          break;

        case '0':

         bQuit = TRUE;

//#if defined(MPFS_USE_EEPROM) &&
(defined(STACK_USE_MPFS) ||
defined(STACK_USE_MPFS2))

           Grabar_AppConfig();

 putrsUART("Configuracion Grabada.\r\n");

 //#else

                // putrsUART("MPFS externo no habilitado --

                // configuracion seria perdida en el reseteo.\r\n");

 //#endif
```

```
        break;
 }
}
}
////////////////////////////////////////////////////////////////////////////////////////
// NOTE: El siguiente código nos ayuda a restringir el acceso de
// cualquier persona a las configuraciones de red.
////////////////////////////////////////////////////////////////////////////////////////
// NOTA: Name [] debe ser de al menos 16 caracteres de
longitud.
// Esto debe ser exactamente de 16 caracteres, como se definió
por especifico NetBIOS.
void Formato_NetBIOS_Nombre(BYTE Nombre[])
{
BYTE i;
Nombre[15] = '\0';
strupr((char*)Nombre);
```

```
i = 0;

while(i < 15u)

{

 if(Nombre[i] == '\0')

 {

  while(i < 15u)

  {

   Nombre[i++] = ' ';

  }

  break;

 }

 i++;

}

}
```

4.3 Diseño de Página Web

Una **página web,** es una fuente de información adaptada para la Red Mundial (World Wide Web) y accesible mediante un navegador de Internet. Esta información se presenta generalmente en formato HTML y puede contener hiper-enlaces a otras páginas web, constituyendo la red enlazada de la World Wide Web.

Las páginas web pueden ser cargadas de un computador local o remoto, llamado Servidor Web, el cual servirá de CLIENTE. El servidor web puede restringir las páginas a una red privada, por ejemplo, una intranet, o puede publicar las páginas en LA Red Mundial. Las páginas web son solicitadas y transferidas de los servidores usando el Protocolo de Transferencia de Hipertexto (HTTP). La acción del Servidor CLIENTE de guardar la página web, se denomina "HOSTING".

Las páginas web pueden consistir en archivos de texto estático, o se pueden leer una serie de archivos con código que instruya al servidor cómo construir el HTML para cada página que es solicitada, a esto se le conoce como Página Web Dinámica.

Las páginas estáticas, generalmente usan la extensión de archivo.htm o.html. Las páginas dinámicas, usan extensiones que generalmente reflejan el lenguaje o tecnología que se utilizó para crear el código, como.php (PHP),.jsp (JavaServer), etc. En estos casos, el servidor debe estar configurado para esperar y entender estas tecnologías.

Las páginas web, generalmente incluyen instrucciones para el tamaño, el color del texto y el fondo, así como hipervínculos a imágenes y algunas veces otros tipos de archivos multimedia.

La estructura tipográfica y el esquema de color es definida por instrucciones de Hojas de Estilo (CSS-Cascading Style Sheet), que pueden estar adjuntas al HTML o pueden estar en un archivo por separado, al que se hace referencia desde el HTML.

Las imágenes son almacenadas en el servidor web como archivos separados.

Otros archivos multimedia, como sonido o video pueden ser incluidos también en las páginas web, como parte de la página o mediante hipervínculos. Juegos y animaciones también pueden ser adjuntados a la página mediante tecnologías como Adobe Flash y Java. Este tipo de material depende de la habilidad del navegador para manejarlo y que el usuario permita su visualización.

Código del lado del cliente como JavaScript o AJAX pueden incluirse adjuntos al HTML o por separado, ligados con el código específico en el HTML. Este tipo de código necesita correr en la computadora cliente, si el usuario lo permite, y puede proveer de un alto grado de interactividad entre el usuario y la página web.

Las páginas web dinámicas, son aquellas que pueden acceder a bases de datos para extraer información que pueda ser presentada al visitante dependiendo de determinados criterios. Ejemplo de esto son páginas que tienen sistemas de administración de contenido o CMS. Estos sistemas permiten cambiar el contenido de la página web sin tener que utilizar un programa de ftp para subir los cambios.

Existen diversos lenguajes de programación, que permiten agregar dinamismo a una página web, tal es el caso de ASP, PHP, JSP y varios más.

Elementos de una página web

Una página web, tiene contenido que puede ser visto o escuchado por el usuario final. Estos elementos incluyen, pero no exclusivamente:

- Texto. El texto editable se muestra en pantalla con alguna de las fuentes que el usuario tiene instaladas.
- Imágenes. Son ficheros enlazados desde el fichero de la página propiamente dicho. Se puede hablar de tres formatos casi exclusivamente: GIF, JPG y PNG.

- Audio, generalmente en MIDI, WAV y MP3.
- Adobe Flash.
- Adobe Shockwave.
- Gráficas Vectoriales (SVG - Scalable Vector Graphics).
- Hipervínculos, Vínculos y Marcadores.

La página web, también puede traer contenido que es interpretado de forma diferente dependiendo del navegador y generalmente no es mostrado al usuario final. Estos elementos incluyen, pero no exclusivamente:

- Scripts, generalmente JavaScript.
- Meta tags.
- Hojas de Estilo (CSS - Cascading Style Sheets).

Crear una página web

Para crear una página web, es necesario un editor de texto o un editor de HTML. Para cargar la información al servidor generalmente se utiliza un cliente FTP.

El diseño de una página web, es completamente personal. El diseño se puede hacer de acuerdo a las preferencias personales o se puede utilizar una plantilla. Mucha gente pública tiene sus propias páginas web usando sitios como GeoCities de Yahoo, Tripod o Angelfire. Estos sitios ofrecen hospedaje gratuito a cambio de un espacio limitado y publicidad.

El diseño de una página web, debe cumplir una serie de requisitos para que pueda ser accesible por cualquier persona independientemente de sus limitaciones físicas o de su entorno. Es lo que llamamos accesibilidad web. Normalmente viene recogidas en normativas, en España UNE:139803, que hacen referencia a las Pautas de Accesibilidad del Contenido Web 1.0 WCAG, desarrolladas por el W3C.

Cuando se crea una página web, es importante asegurarse que cumple con los estándares del Consorcio World Wide Web (W3C) para el HTML, CSS, XML, etc. Los estándares aseguran que todos los navegadores mostrarán información idéntica sin ninguna consideración especial. Una página propiamente codificada será accesible para diferentes navegadores, ya sean nuevos o antiguos, resoluciones, así como para usuarios con incapacidades auditivas y visuales.

Diseño de la página web del Microcontrolador

El diseño de la página web del Microcontrolador que será grabada posteriormente en una EEPROM se lo hará en Dreamweaver versión 8.0 y constará con las siguientes partes:

Identificación clara del Logotipo de la Facultad, de la ESPOL; se importarán las imágenes de las páginas autorizadas de la universidad que se utilizarán en dicho propósito y se ubicarán en la carpeta de imágenes de dónde serán importadas para mostrar en la aplicación final.

Nombre y dirección electrónica de los Integrantes: Se incluirán los nombres de los integrantes del Proyecto de Tesis en el diseño web.

Título y Objetivos del Proyecto: Debido a que es una aplicación académica, hemos decidido nombrar los objetivos del proyecto para verificar su cumplimiento y el buen funcionamiento del mismo.

Hipervínculos: Al utilizar solo una página web, haremos un hipervínculo entre ellas para pasar de una a otra sin ningún problema.

Estado de Luces: En la primera página mostraremos el estado de las luces, en esta sección también tendremos la facilidad de interactuar con ellas encendiéndolas o apagándolas según sea el caso; se han implementado cuatro luces con su respectiva ventana para elegir el estado de la misma.

4.3.1 Dreamweaver

El programa Dreamweaver es una aplicación muy completa, de gran funcionalidad y muy potente, pero a la vez está diseñada para hacer más cómodo el trabajo al usuario.

Dreamweaver combina potencia y comodidad permitiendo que el usuario personalice a su gusto el entorno de trabajo.

Las funciones de edición visual de Dreamweaver permiten crear páginas Web de forma rápida, sin escribir una sola línea de código. Podemos ver todos los elementos o activos del sitio y arrastrarlos desde un panel fácil de usar directamente hasta un documento. Agilizamos el flujo de trabajo de desarrollo mediante la creación y edición de imágenes en Macromedia Fireworks o en otra aplicación de gráficos y su posterior importación directa a Dreamweaver. Este programa también contiene herramientas que facilitan la adición de activos de Flash a las páginas web.

Además de las funciones de arrastrar y soltar que le ayudan a crear páginas web, Dreamweaver ofrece un entorno de codificación con todas las funciones, que incluye herramientas para la edición de código (tales como coloreado de código, terminación automática de etiquetas, barra de herramientas para codificación y contracción de código) y material de referencia para lenguajes sobre hojas de estilos en cascada (CSS), JavaScript y ColdFusion Markup Language (CFML) entre otros. La tecnología Roundtrip HTML de Macromedia importa los documentos con código manual HTML sin modificar el formato del código. Posteriormente, podemos formatear el código con el estilo que prefiera. Además, permite crear aplicaciones Web dinámicas de bases de datos empleando tecnologías de servidor como CFML, ASP.NET, ASP, JSP y PHP. Si se prefiere trabajar con datos en XML, incorpora herramientas que le permiten crear fácilmente páginas XSLT, adjuntar archivos XML y mostrar datos XML en sus páginas.

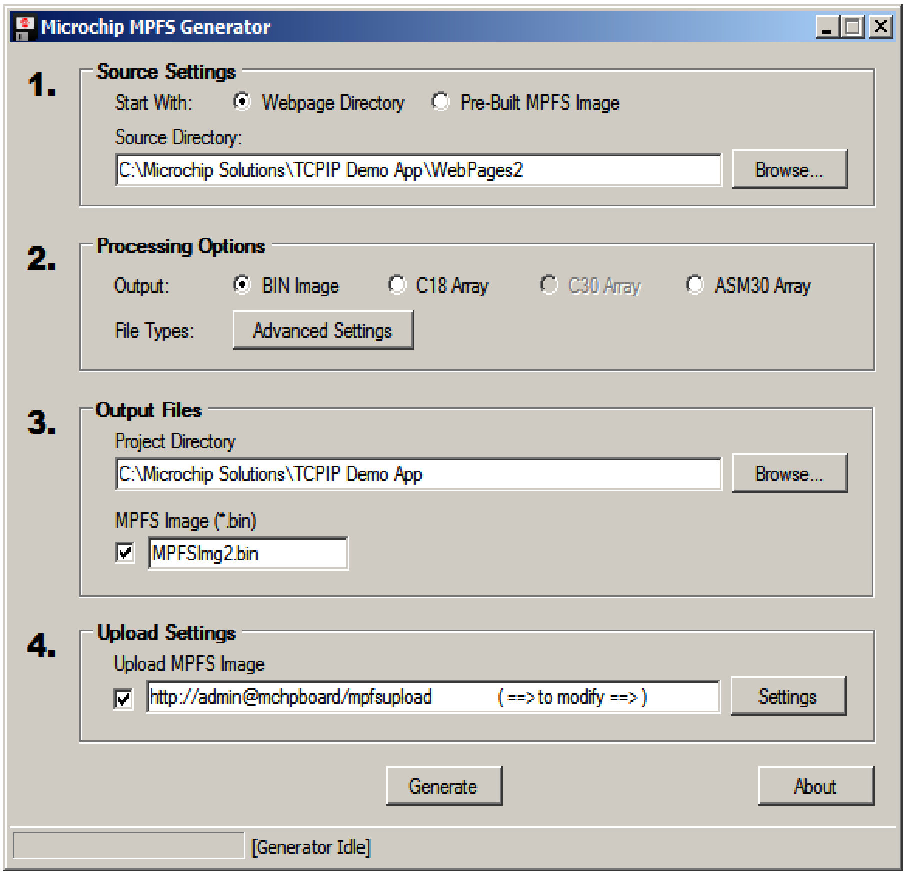
Podemos crear nuestros propios objetos y comandos, modificar métodos abreviados de teclado e incluso escribir código JavaScript para ampliar las posibilidades que ofrece Dreamweaver con nuevos comportamientos, inspectores de propiedades e informes de sitios.

4.3.2 MPFS2 Utility

La utilidad MPFS 2 tiene muchas características. El propósito principal es empaquetar páginas web en un formato para el almacenamiento eficiente en un sistema embebido. Esto también adhiere variables dinámicas y genera el archivo HTTPPrint.h para asegurar que las llamadas a las mismas se realicen el número necesario de veces. Finalmente, cuando se desarrolla una aplicación que usa almacenamiento en una EEPROM, la utilidad MPFS2 puede cargar imágenes hacia la EEPROM usando la funcionalidad de carga MPFS construida en el servidor http.

Construyendo Imágenes MPFS2

La utilidad MPFS 2 tiene cuatro pasos, denotados en el margen izquierdo de la ventana de diálogo. Para construir una imagen MPFS, seleccionamos “Comenzar con: Directorio de Página Web” en el paso uno y seleccionamos el directorio en que la página se almacenará.



Microchip MPFS Generator

1. Source Settings

Start With: ● Webpage Directory ○ Pre-Built MPFS Image

Source Directory: C:\Microchip Solutions\TCPIP Demo App\WebPages2

2. Processing Options

Output: ● BIN Image ○ C18 Array ○ C30 Array ○ ASM30 Array

3. Output Files

Project Directory: C:\Microchip Solutions\TCPIP Demo App

MPFS Image (*.bin): ☑ MPFSImg2.bin

4. Upload Settings

Upload MPFS Image: ☑ http://admin@mchpboard/mpfsupload (==> to modify ==>)

[Generator Idle]

Figura 4.12. Ventana de Generador de Imágenes de MPFS

El segundo paso, selecciona el formato de salida. Si el almacenamiento es en una EEPROM, seleccionamos “Imagen BIN” como formato de salida. Si utilizamos la memoria interna de programa, indicamos “Arreglo C18” para usar con secciones de 8 bits, y “Arreglo ASM 30” para aplicaciones de 16 bits.

La opción “Arreglo C30” es para aplicaciones con el compilador C30 de Microchip. El método más eficiente de almacenamiento de datos en la memoria de programa para secciones de 16 bits requiere ensamblaje.

El tercer paso, nos pregunta sobre el directorio del proyecto en MPLAB IDE. La herramienta MPFS escribirá el archivo de la imagen en el directorio del proyecto, y también actualizará el archivo HTTPPrint.h, si es necesario.

El cuarto paso, controla las características de carga. Cuando es usada la EEPROM para almacenamiento, la opción para cargar la reciente imagen creada estará disponible. Revisamos el casillero seguido a “Cargar Imagen MPFS” para habilitar esta característica. El botón “Ajustes” puede ser usado para configurar correctamente el nombre de dispositivo.

Si está siendo usada la memoria interna del programa, la imagen será compilada con el proyecto y debido a eso las cargas no estarán disponibles. Debe asegurarse que la salida del archivo fuente indicado, está incluida en el proyecto.

Una vez listo todo, seleccionamos el botón “Generar” para crear la imagen y cargarla en la tarjeta destino.

Cargando una Imagen Pre-Existente

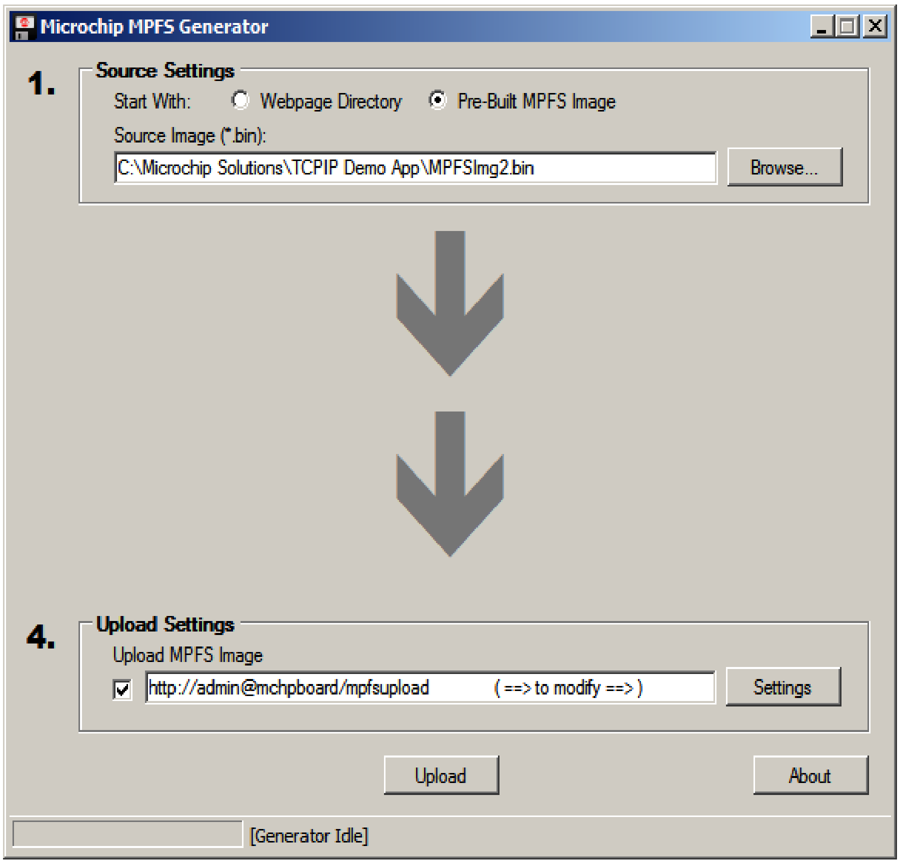
Existen dos maneras para cargar una imagen pre existente a la EEPROM: la primera. es cargándola directamente desde el directorio en el buscador y la segunda, es usar la utilidad MPFS2 para cargar la imagen.

Para usar la utilidad MPFS2 con la finalidad de cargar una imagen, comenzamos por seleccionar “Comenzar con: Pre-construir imagen” en la parte superior del paso 1. Seleccionamos la imagen a cargar.

Este procedimiento trabaja tanto para imágenes de MPFS2 como para imágenes MPFS clásicas. La utilidad MPFS2 detectará automáticamente el tipo de imagen y usará FTP para imágenes MPFS clásicas y HTTP para imágenes MPFS2.

Los pasos dos y tres no son largamente necesarios, y pueden ser removidos. Procedemos directamente al cuarto paso y verificamos que las características de carga estén correctas.

Una vez que todo esté correctamente configurado, seleccionamos el botón cargar. La imagen será cargada a la tarjeta.



Source Settings

Start With: ○ Webpage Directory ● Pre-Built MPFS Image

Source Image (*.bin): C:\Microchip Solutions\TCPIP Demo App\MPFSImg2.bin

Upload Settings

Upload MPFS Image

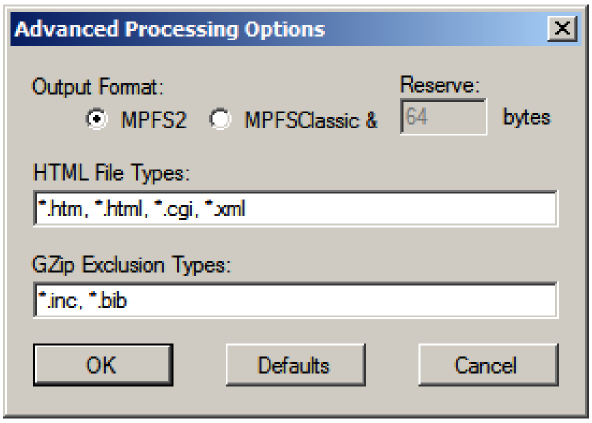
☑ http://admin@mchpboard/mpfsupload

[Generator Idle]

Figura 4.13 Ventana de Carga de Imagen Pre-existente

Características Avanzadas de Configuración.

Las características avanzadas de procesamiento encontradas en el segundo paso, nos proveen de mayor control sobre el procesamiento de archivos.



Advanced Processing Options

Output Format: ● MPFS2 ○ MPFSClassic & Reserve: 64 bytes

HTML File Types: *.htm, *.html, *.cgi, *.xml

GZip Exclusion Types: *.inc, *.bib

Figura 4.14 Ventana de Opciones de Procesamiento

Avanzado con MPFS 2

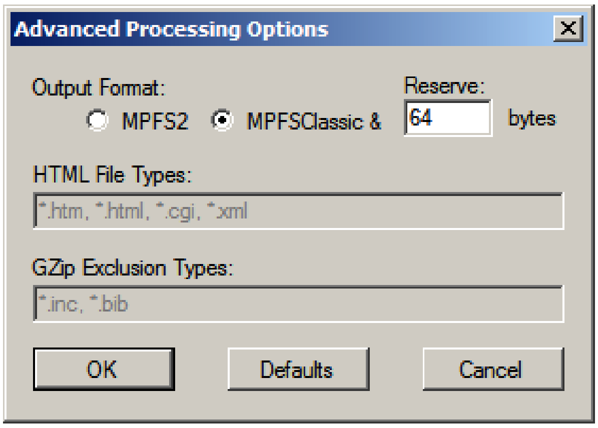
El formato de salida permite seleccionar entre los formatos MPFS2 y MPFS clásico. El servidor HTTP2 es solamente compatible con MPFS2.

Los tipos de archivo HTML indican los tipos de archivos a ser analizados para variables dinámicas. Por defecto, todos los archivos con extensión htm, html, cgi o xml son analizados. Si una aplicación contiene variables dinámicas en otros tipos de archivo, estos tipos deben ser añadidos a la lista.

Los tipos de exclusión Gzip indican que tipos de archivos nunca deberían ser compresos. Comprimir archivos con Gzip guarda ambos, espacios de almacenamiento y tiempo de transmisión. Sin embargo, esto es solo adecuado para contenido estático como CCS o JavaScript. Los archivos con variables dinámicas serán excluidos automáticamente. Los archivos incluidos vía ~inc:filename~ no deberían ser comprimidos, no se debería usar ningún archivo BIB para el módulo SNMP. Los tipos de archivo adicionales pueden ser añadidos a las lista si alguna aplicación accede a MPFS.

Trabajando con Imágenes MPFS clásicas.

Para generar una imagen MPFS clásica, seleccionamos "Características avanzadas" , además seleccionamos "MPFS Classic" como tipo de salida.



Advanced Processing Options

Output Format: ○ MPFS2 ● MPFSClassic &

Reserve: 64 bytes

HTML File Types: *.htm, *.html, *.cgi, *.xml

GZip Exclusion Types: *.inc, *.bib

Figura 4.15 Ventana de Opciones de Procesamiento

Avanzado con MPFS clásico

Para imágenes MPFS clásicas, el generador debe conocer las características de MPFS_RESERVE_BLOCK en *TCPIPConfig.h*. Esta opción controla la cantidad de espacio que es seleccionada en la EEPROM para el uso de la aplicación. Si la aplicación no usa EEPROM, está configuración es ignorada.

El análisis de variables dinámicas y la compresión GZIP son característicos del servidor HTTP2, que requiere MPFS2. Solamente el servidor original HTTP es compatible con MPFS.

Si el almacenamiento en la EEPROM está siendo usado, la utilidad MPFS2 puede aún cargar imágenes hacia la tarjeta. Esta funcionalidad requiere que el servidor FTP sea habilitado en el proyecto, pero el uso en la utilidad es idéntico. Cuando las imágenes MPFS clásicas son cargadas, la utilidad automáticamente selecciona el protocolo FTP.

4.4 Diseño y Desarrollo de la Tarjeta

Hemos basado el diseño de nuestra tarjeta controladora en una tecnología, que actualmente está en auge cómo es la de sistemas embebidos, específicamente el Microcontrolador que estamos utilizando, el PIC18F97J60, que posee un módulo de Ethernet integrado en su circuitería.

El objetivo principal de nuestro diseño, es la de crear una aplicación que permita controlar y monitorear las luminarias de una vivienda, por medio de Internet. Por ello, hemos pensado en las características antes mencionadas del PIC18F97J60 que será el elemento central de nuestra tarjeta.

La tarjeta controladora permite la conectividad a Internet, a través del conjunto de protocolos del modelo TCP/IP que estarán manejados por el módulo Ethernet de nuestro Microcontrolador, el cual tiene entradas y salidas específicas para el propósito de transmisión y recepción de datos; estas a su vez pasan por un circuito de transformadores que sirven de acoplamiento entre el Microcontrolador y la red externa.

Cabe recalcar, que este conjunto de transformadores está integrado junto a un conector RJ45 hembra en un solo elemento, que además posee dos leds, uno al lado derecho y otro al lado izquierdo del elemento, que nos indican la actividad y el proceso de enlace respectivamente. Esto muestra si una aplicación de Ethernet está transmitiendo o recibiendo un paquete y si la conexión está activa.

Para la programación de la tarjeta utilizaremos un conector RJ 11 hembra que servirá de interfaz entre el PIC y el programador MPLAB ICD 2, del que ya hablamos en el capítulo anterior.

Hemos implementado un puerto RS232 con un conector DB9, con el fin de facilitar el monitoreo la configuración de parámetros de red como nombre de cliente, dirección IP, máscara de subred, número de puerta de enlace y permitir o no el control de direccionamiento dinámico del PIC.

Con la ayuda del código fuente que nos provee Microchip, el TCP/IP stack, podemos desarrollar y experimentar diferentes aplicaciones, para nuestro caso, la conectividad entre ellas.

Nuestro diseño se alimenta directamente de una línea residencial de 120 v. a 60 Hz, por medio de un adaptador a 9 v. dc obtenemos el potencial necesario para el óptimo funcionamiento de nuestros reguladores, uno a 3.3 v. dc y el otro a 5 v. dc que posteriormente nos darán el voltaje necesario que requieren los diferentes elementos que conforman nuestra tarjeta.

Observaremos en un LCD la configuración actual de la dirección IP, y algún otro parámetro o frase que incrementemos en la programación.

Finalmente, vamos a lograr la comunicación con diferentes adaptadores, en nuestro caso el de potencia para las luminarias, a través de tres conectores macho de 16 pines que serán unidos por medio de cables planos a los adaptadores antes mencionados. En estos adaptadores hemos incluido varios pines de entrada y/o salida para dar una escalabilidad a nuestro proyecto.

4.4.1 Elementos Utilizados en la Tarjeta

En la elaboración de la tarjeta hemos utilizado varios elementos que cumplen algunas funciones conforme lo detallamos a continuación.

El Microcontrolador PIC18F97J60: Es la parte principal de nuestro diseño, a través de él podremos lograr los objetivos propuestos; tiene la particularidad de que incluye un módulo de Ethernet embebido a más de diferentes entradas y salidas, diferentes módulos para utilizar y configurar tales como los de oscilación, memoria, comunicación serial, etc.

La memoria EEPROM 25LC256: Esta nos dará la capacidad de almacenamiento al proveernos de 256 Kbits de memoria, la cual utilizaremos para guardar las páginas web de monitoreo y control de nuestra aplicación.

El oscilador: Mediante un cristal de 25 MHz daremos sincronismo al PIC, para la transmisión y recepción de datos del módulo Ethernet.

El LCD: Con sus dos líneas de 16 caracteres nos mostrará la configuración de la dirección IP y algún otro mensaje que necesitemos el mismo que será ingresado mediante programación del PIC.

Las botoneras: Estas nos ayudarán en diferentes funciones de nuestra aplicación, como es el caso para inicializar nuevamente el PIC a través de la señal MCLR, habilitar la comunicación serial a través del RS232.

Conector Modular RJ45: Permiten la conectividad física con la red, el módulo nos provee de un conector hembra modular, transformadores integrados para acoplar magnéticamente la entrada y eliminar cualquier tipo de corriente inducida; además, posee dos leds que nos indicarán la actividad y enlace respectivamente.

Conector Modular RJ11: Nos permitirá la comunicación y programación del PIC a través del MPLAB ICD 2 que utilizaremos para nuestro proyecto.

Puerto Serial: Tenemos un puerto RS232 con un conector DB9 y el respectivo desplazador de nivel MAX3232 que nos permitirá configurar la dirección IP a través de una comunicación serial estándar.

Reguladores: Aplicaremos dos reguladores, de 3.3v. y 5 v. respectivamente para energizar el PIC y los demás elementos de la tarjeta

Conectores de salida: Hemos utilizado tres conectores de 16 pines que darán escalabilidad a nuestro proyecto, estos nos permiten llevar las salidas del PIC a otras aplicaciones o a su vez importar señales de entradas de las mismas.

Capacitores y Resistencias: Los capacitadores y resistencias utilizadas en nuestro proyecto nos sirven en diferentes casos como protección, filtros, divisores de voltaje, etc.

A continuación, podemos observamos el esquemático de nuestra tarjeta, realizado en el software Altium Designer, en el cual notamos todos los elementos antes mencionados.

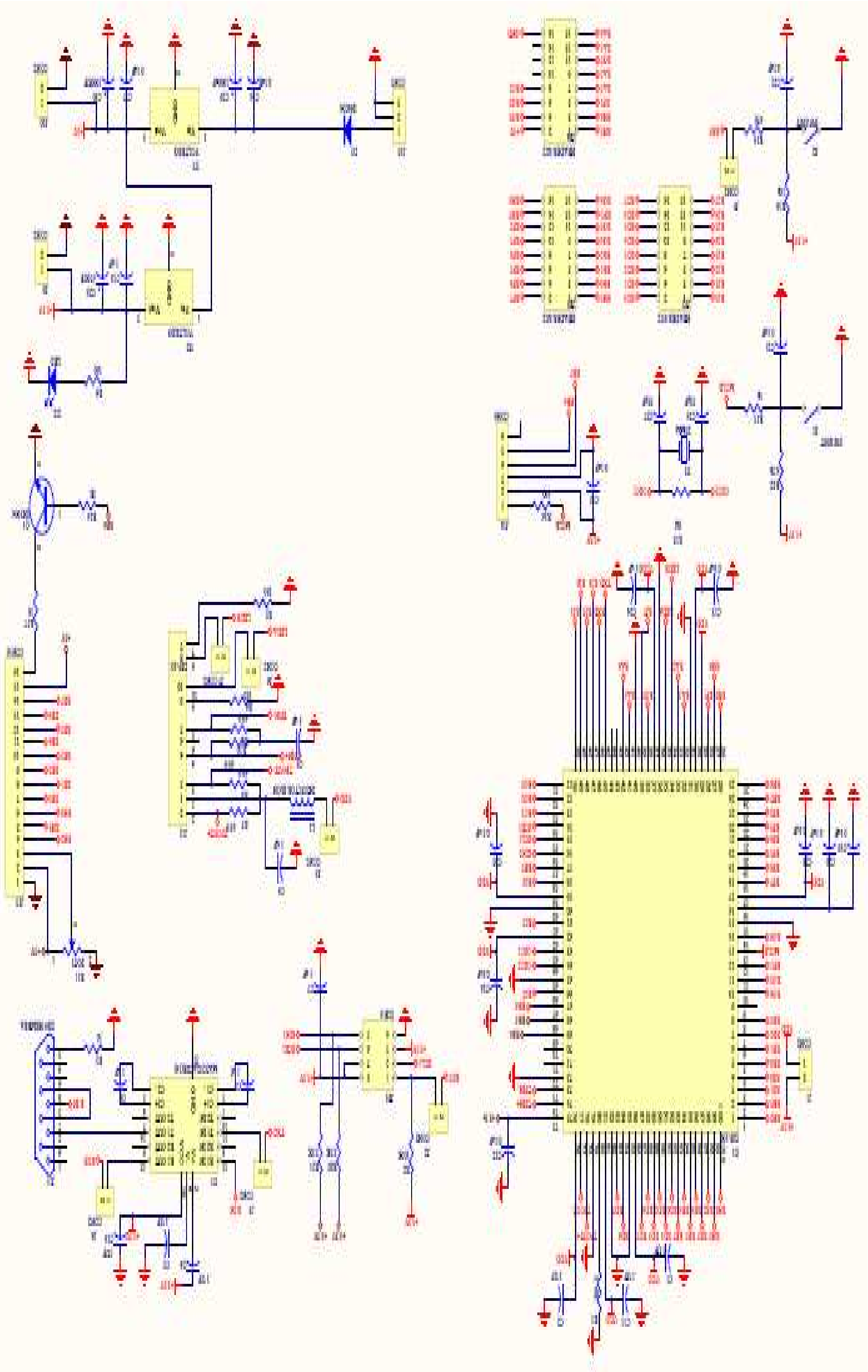


Figura 4.16 Esquema de Tarjeta usando Altium Designer

4.4.2 Detalles del Diseño de la Tarjeta

En nuestro diseño existen ciertas partes o módulos importantes que citaremos a continuación con la respectiva descripción y funcionamiento:

En el módulo o etapa de energía tenemos la entrada principal de la misma por medio de un adaptador de 9v conectado a una línea de 110 v. a 60 Hz., estos 9v pasarán por medio de un juego de resistencias y capacitores hasta llegar al regulador LM2940S-5.0 de 5v. para limitar el voltaje a esta cantidad. Esta salida de voltaje tendrá dos destinos, el primero, hacia el LCD para su alimentación y el segundo, pasará al regulador LM2937ET de 3.3 v. cuya salida de dicho voltaje será la que alimente al Microcontrolador y a otros elementos como la memoria EEPROM, el adaptador de niveles MAX3232, conector ICD y botoneras o pulsadores.

Tenemos también una interfaz serial RS232, mediante la cual podremos comunicar un ordenador con la tarjeta controladora. Esta interfaz, consta de un conector DB9 hembra mediante el cual recibimos las señales del computador que van de +15 v. a -15 v. e ingresan a un adaptador de niveles MAX3232 que junto a 5 capacitores electrolíticos de 0.1 microfaradios podremos obtener los niveles lógicos de 0 v. y 3.3 v. con los cuales trabajará el Microcontrolador.

En la siguiente figura podemos observar cómo trabaja la interfaz RS232, adaptando los niveles lógicos entre el computador y el Microcontrolador.

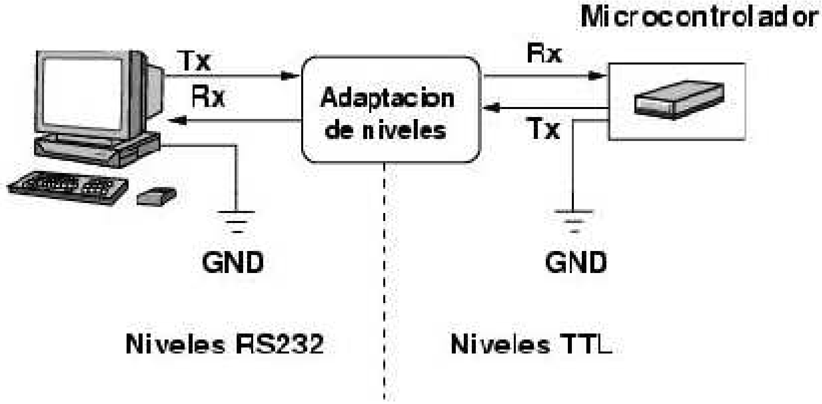


Figura 4.17 Esquema de Funcionamiento de RS232

Hemos implementado también un conector ICD que permitirá programar el Microcontrolador con el MPLAB ICD 2, esta conexión consta de un resistor y un capacitor que actúa como filtro. Dos de las entradas de este conector irán directamente a los pines de programación RB6 y RB7 del Microcontrolador.

Con el fin de almacenar la página web de nuestra aplicación, utilizamos una memoria de 256Kbit EEPROM 25LC256 que se comunicará con el modulo del puerto serial sincrónico maestro

MSSP del Microcontrolador, a través de los pines de entrada y salida serial digital RC4/SDI1 y RC5/SDO1 respectivamente. Además la memoria se sincronizará a través del pino RC3/SCK1 en el mismo módulo del Microcontrolador.

Para poder ayudarnos durante la configuración de los parámetros de red hemos colocado una pantalla de cristal líquido LCD de dos líneas de 16 caracteres, muestra la IP actual que tiene nuestra tarjeta controladora, esto es de gran importancia para nosotros, una vez que, con esta información podemos comunicarnos con nuestra tarjeta a través del ordenador en una conexión directa o por medio de un acceso remoto. La información, le llegará al LCD por los pines RE0, RE1, RE2, RE3, RE4, RE5, RE6, RE7, RH0, RH1 y RH2 del PIC.

Nuestro diseño también comprende un conector modular hembra RJ45 08B0-1X1T-36-F0724, que tiene la propiedad de acoplar magnéticamente la señal en la red con la tarjeta, mediante un conjunto de transformadores empaquetados en su interior. Posee dos LEDs que nos mostraran el estado del enlace con la red los cuales reciben su señal de los pines RA0 y RA1 respectivamente. Este elemento permite realizar la comunicación con la red a través de los pines de transmisión y recepción de datos, TPIN+, TPIN- y TPOUT+, TPOUT- del PIC correspondientemente. Además en su configuración consta de resistores, capacitores y un inductor que son utilizados con el propósito de reducir las interferencias electromagnéticas.

Finalmente, tenemos los puertos de interconexión con otras aplicaciones que dará escalabilidad a nuestra tarjeta controladora, como citamos anteriormente en otros capítulos, en nuestro caso se conectará la misma a una tarjeta de potencia que hará posible el monitoreo y control de las luces de una vivienda; para este fin utilizaremos únicamente las salidas RJ0, RJ1, RJ2, RJ3, RJ4, RJ5, RJ6, RJ7.

4.5 Diseño y Desarrollo del Adaptador

Para demostrar el funcionamiento de nuestro proyecto, hemos diseñado una tarjeta de potencia, cuyo objetivo será el de amplificar el nivel de voltaje que se obtiene a la salida de la tarjeta controladora, con el propósito de que ésta aplicación se use sin problemas en las luminarias de una vivienda y automatizar su funcionamiento.

Podemos ver a continuación, un diagrama de bloques donde se denota la estructura de la tarjeta de potencia de una manera muy básica y que será descrita en las siguientes secciones.

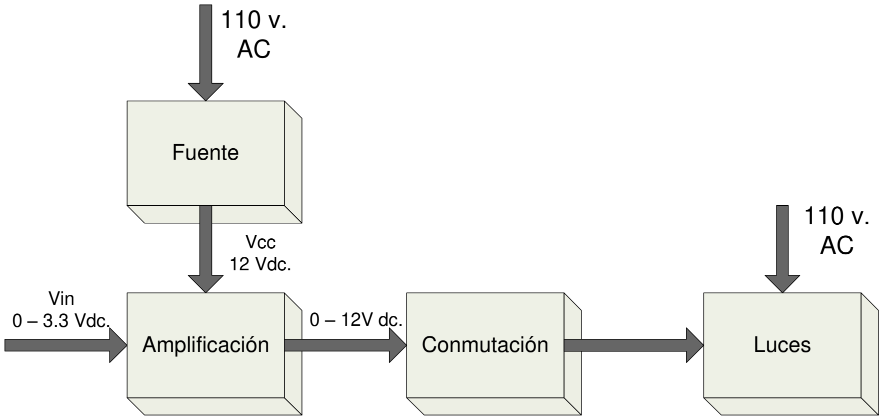


Figura 4.18 Diagrama de bloques de Tarjeta de Potencia

Cabe recalcar, que este esquema es totalmente escalable, se ha diseñado para cuatro luces para demostración, pero se puede expandir sin ninguna dificultad.

4.5.1 Elementos Utilizados en el Adaptador

Al elaborar la tarjeta hemos utilizado los siguientes elementos que nos han permito llevar a cabo el correcto funcionamiento de la misma:

Transformador: Utilizamos un transformador de 12 v. para reducir el voltaje al nivel necesario de los demás elementos de la tarjeta.

Puente de Diodos: Permitirá rectificar el voltaje alterno y llevarlo a un nivel dc para la alimentación requerida por los demás elementos.

Capacitores: Tenemos capacitores que eliminarán en gran medida el rizado que pudiese causar cualquier daño o mal funcionamiento.

Circuito Integrado LM2803: Nos permitirá amplificar el voltaje de 3.3 v. dc a 12 v. dc.

Relés de 12 v. dc.: Servirán de conmutación para llevar a cabo el encendido de las luces que demostrarán el correcto funcionamiento de nuestro proyecto.

Leds: Servirán de indicadores de estado de los relés.

Podemos observar en la siguiente figura la distribución de elementos en la placa:

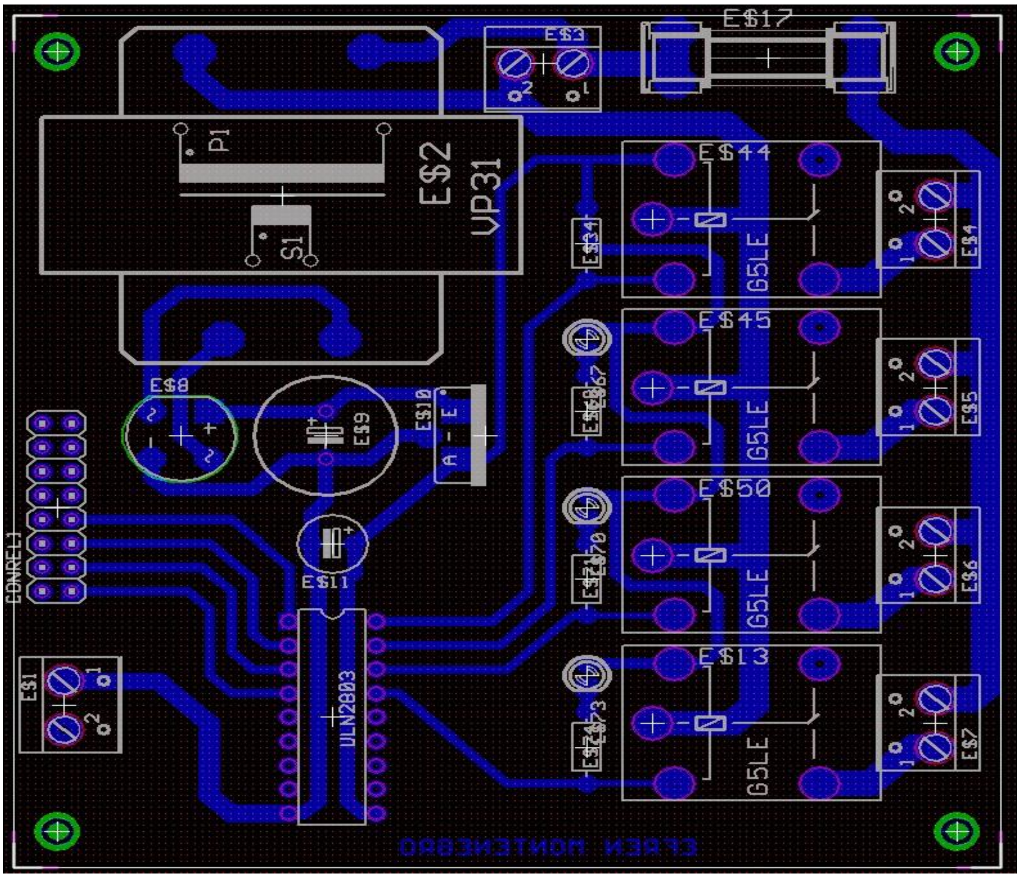


Figura 4.19 Distribución de Elementos de Tarjeta de Potencia

4.5.2 Detalles del Diseño del Adaptador

El adaptador o tarjeta de potencia ha sido diseñado, con elementos fáciles de conseguir en el mercado y a su vez estos han sido montados de forma tal que aprovechen el reducido espacio de la placa para compactar su funcionamiento.

Empezamos por la etapa de alimentación que consta de una fuente, la cual será energizada con 120 voltios a 60 Hz por medio de una bornera y su respectivo fusible de protección; mediante un transformador reducimos el voltaje a 12 voltios y lo rectificamos mediante un puente de diodos para convertirlo en DC. Se utilizan también, dos capacitores y un diodo adicional para eliminar el rizado y proteger el circuito respectivamente.

Este nivel de voltaje de 12 v. dc será utilizado para polarizar un integrado que hará las veces de amplificador.

En la siguiente etapa del adaptador encontramos el circuito integrado LN2803 que servirá para amplificar el nivel de 3.3 voltios dc provenientes de la tarjeta controladora a través de un cable plano de 16 pines en 12 voltios dc que activarán cuatro relés en la siguiente etapa.

La etapa final de nuestro adaptador consta de cuatro relés cuyos contactos se juntarán al momento de que les llegue la señal de 12 voltios dc del LN2803 dependiendo de las órdenes de la tarjeta controladora. Al juntarse los contactos de los relés, estos cerrarán el circuito de cada una de las luces y harán que estas se enciendan o se apaguen según sea el caso. Cada relé tiene su respectivo led indicador de estado.

CAPÍTULO 5

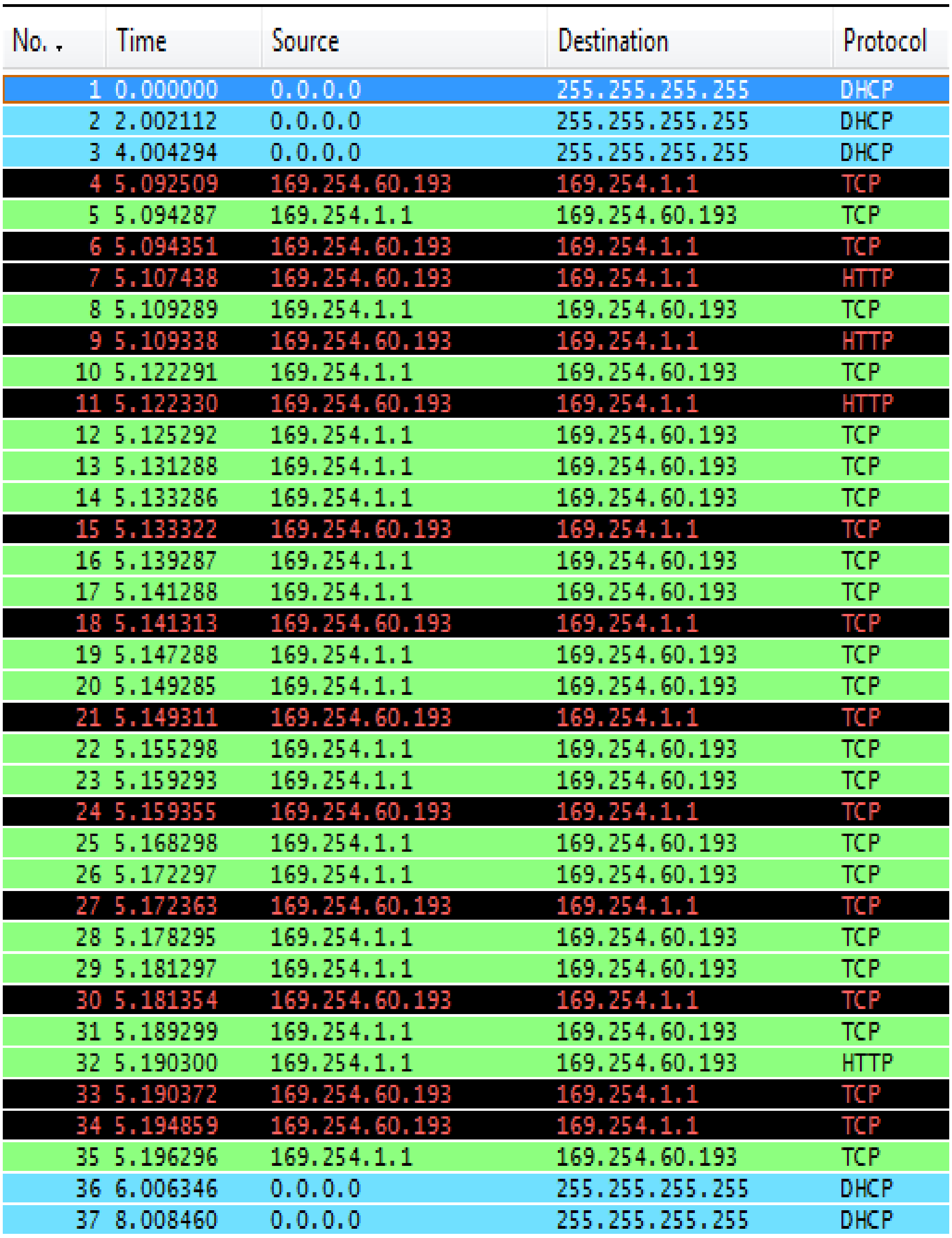
5. ANÁLISIS DE FUNCIONAMIENTO

Para realizar el análisis de funcionamiento, hemos utilizado el software Wireshark para el conteo de paquetes tanto en la transmisión como en la recepción; además, se realizarán con el mismo software gráficos estadísticos en base al número de paquetes y la clase de los mismos.

5.1 Transmisión

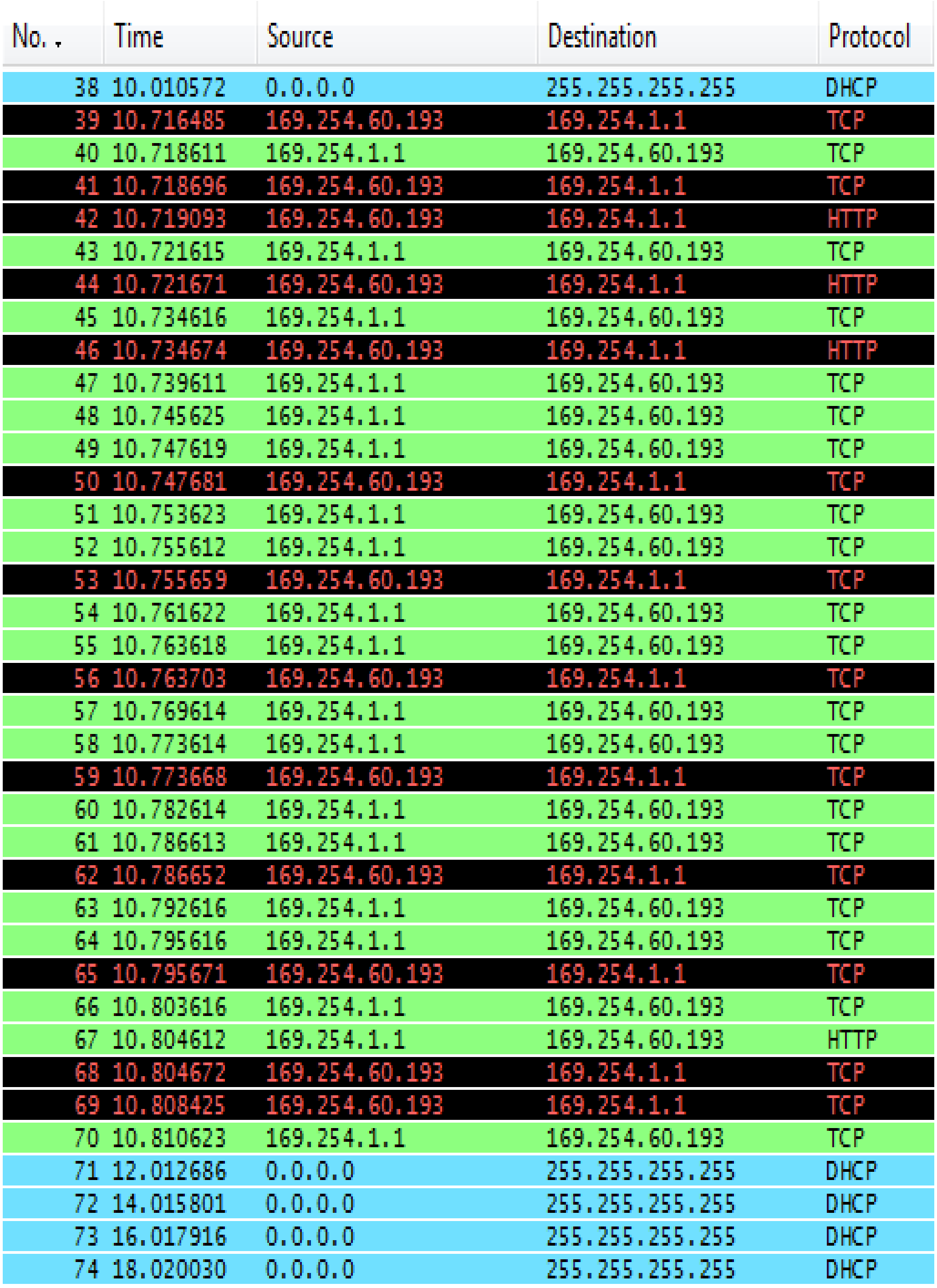
Al realizar la transmisión de datos pudimos obtener los siguientes resultados: en cuanto al conteo de paquetes que analizaremos a continuación.

Primeramente, el software realizó la captura de paquetes que vemos en la siguiente figura:



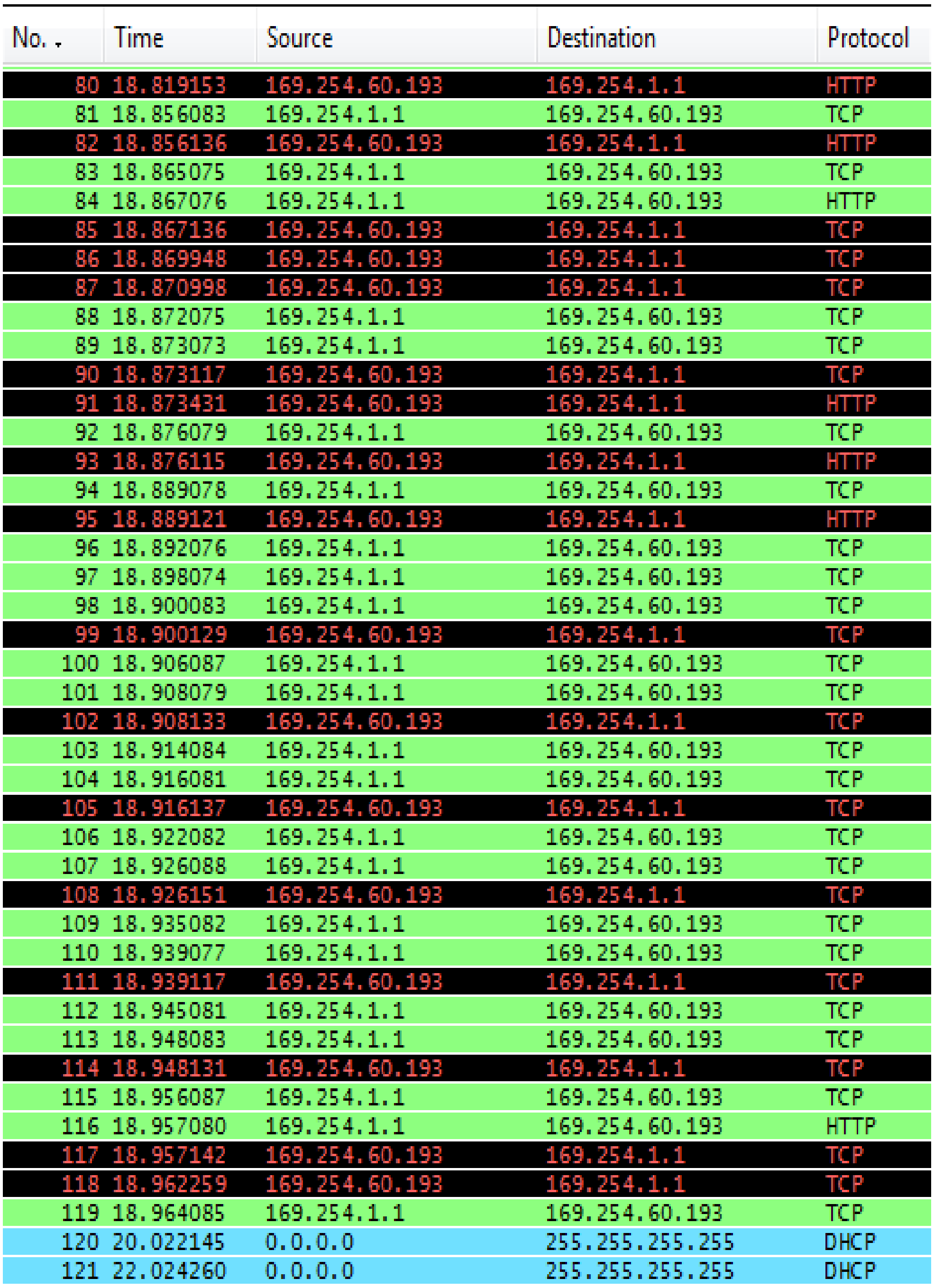
No.	Time	Source	Destination	Protocol
1	0.000000	0.0.0.0	255.255.255.255	DHCP
2	2.002112	0.0.0.0	255.255.255.255	DHCP
3	4.004294	0.0.0.0	255.255.255.255	DHCP
4	5.092509	169.254.60.193	169.254.1.1	TCP
5	5.094287	169.254.1.1	169.254.60.193	TCP
6	5.094351	169.254.60.193	169.254.1.1	TCP
7	5.107438	169.254.60.193	169.254.1.1	HTTP
8	5.109289	169.254.1.1	169.254.60.193	TCP
9	5.109338	169.254.60.193	169.254.1.1	HTTP
10	5.122291	169.254.1.1	169.254.60.193	TCP
11	5.122330	169.254.60.193	169.254.1.1	HTTP
12	5.125292	169.254.1.1	169.254.60.193	TCP
13	5.131288	169.254.1.1	169.254.60.193	TCP
14	5.133286	169.254.1.1	169.254.60.193	TCP
15	5.133322	169.254.60.193	169.254.1.1	TCP
16	5.139287	169.254.1.1	169.254.60.193	TCP
17	5.141288	169.254.1.1	169.254.60.193	TCP
18	5.141313	169.254.60.193	169.254.1.1	TCP
19	5.147288	169.254.1.1	169.254.60.193	TCP
20	5.149285	169.254.1.1	169.254.60.193	TCP
21	5.149311	169.254.60.193	169.254.1.1	TCP
22	5.155298	169.254.1.1	169.254.60.193	TCP
23	5.159293	169.254.1.1	169.254.60.193	TCP
24	5.159355	169.254.60.193	169.254.1.1	TCP
25	5.168298	169.254.1.1	169.254.60.193	TCP
26	5.172297	169.254.1.1	169.254.60.193	TCP
27	5.172363	169.254.60.193	169.254.1.1	TCP
28	5.178295	169.254.1.1	169.254.60.193	TCP
29	5.181297	169.254.1.1	169.254.60.193	TCP
30	5.181354	169.254.60.193	169.254.1.1	TCP
31	5.189299	169.254.1.1	169.254.60.193	TCP
32	5.190300	169.254.1.1	169.254.60.193	HTTP
33	5.190372	169.254.60.193	169.254.1.1	TCP
34	5.194859	169.254.60.193	169.254.1.1	TCP
35	5.196296	169.254.1.1	169.254.60.193	TCP
36	6.006346	0.0.0.0	255.255.255.255	DHCP
37	8.008460	0.0.0.0	255.255.255.255	DHCP

Figura 5.1 Paquetes del 1 al 37 Capturados en Transmisión



No.	Time	Source	Destination	Protocol
38	10.010572	0.0.0.0	255.255.255.255	DHCP
39	10.716485	169.254.60.193	169.254.1.1	TCP
40	10.718611	169.254.1.1	169.254.60.193	TCP
41	10.718696	169.254.60.193	169.254.1.1	TCP
42	10.719093	169.254.60.193	169.254.1.1	HTTP
43	10.721615	169.254.1.1	169.254.60.193	TCP
44	10.721671	169.254.60.193	169.254.1.1	HTTP
45	10.734616	169.254.1.1	169.254.60.193	TCP
46	10.734674	169.254.60.193	169.254.1.1	HTTP
47	10.739611	169.254.1.1	169.254.60.193	TCP
48	10.745625	169.254.1.1	169.254.60.193	TCP
49	10.747619	169.254.1.1	169.254.60.193	TCP
50	10.747681	169.254.60.193	169.254.1.1	TCP
51	10.753623	169.254.1.1	169.254.60.193	TCP
52	10.755612	169.254.1.1	169.254.60.193	TCP
53	10.755659	169.254.60.193	169.254.1.1	TCP
54	10.761622	169.254.1.1	169.254.60.193	TCP
55	10.763618	169.254.1.1	169.254.60.193	TCP
56	10.763703	169.254.60.193	169.254.1.1	TCP
57	10.769614	169.254.1.1	169.254.60.193	TCP
58	10.773614	169.254.1.1	169.254.60.193	TCP
59	10.773668	169.254.60.193	169.254.1.1	TCP
60	10.782614	169.254.1.1	169.254.60.193	TCP
61	10.786613	169.254.1.1	169.254.60.193	TCP
62	10.786652	169.254.60.193	169.254.1.1	TCP
63	10.792616	169.254.1.1	169.254.60.193	TCP
64	10.795616	169.254.1.1	169.254.60.193	TCP
65	10.795671	169.254.60.193	169.254.1.1	TCP
66	10.803616	169.254.1.1	169.254.60.193	TCP
67	10.804612	169.254.1.1	169.254.60.193	HTTP
68	10.804672	169.254.60.193	169.254.1.1	TCP
69	10.808425	169.254.60.193	169.254.1.1	TCP
70	10.810623	169.254.1.1	169.254.60.193	TCP
71	12.012686	0.0.0.0	255.255.255.255	DHCP
72	14.015801	0.0.0.0	255.255.255.255	DHCP
73	16.017916	0.0.0.0	255.255.255.255	DHCP
74	18.020030	0.0.0.0	255.255.255.255	DHCP

Figura 5.2 Paquetes del 38 al 74 capturados en la transmisión



No.	Time	Source	Destination	Protocol
80	18.819153	169.254.60.193	169.254.1.1	HTTP
81	18.856083	169.254.1.1	169.254.60.193	TCP
82	18.856136	169.254.60.193	169.254.1.1	HTTP
83	18.865075	169.254.1.1	169.254.60.193	TCP
84	18.867076	169.254.1.1	169.254.60.193	HTTP
85	18.867136	169.254.60.193	169.254.1.1	TCP
86	18.869948	169.254.60.193	169.254.1.1	TCP
87	18.870998	169.254.60.193	169.254.1.1	TCP
88	18.872075	169.254.1.1	169.254.60.193	TCP
89	18.873073	169.254.1.1	169.254.60.193	TCP
90	18.873117	169.254.60.193	169.254.1.1	TCP
91	18.873431	169.254.60.193	169.254.1.1	HTTP
92	18.876079	169.254.1.1	169.254.60.193	TCP
93	18.876115	169.254.60.193	169.254.1.1	HTTP
94	18.889078	169.254.1.1	169.254.60.193	TCP
95	18.889121	169.254.60.193	169.254.1.1	HTTP
96	18.892076	169.254.1.1	169.254.60.193	TCP
97	18.898074	169.254.1.1	169.254.60.193	TCP
98	18.900083	169.254.1.1	169.254.60.193	TCP
99	18.900129	169.254.60.193	169.254.1.1	TCP
100	18.906087	169.254.1.1	169.254.60.193	TCP
101	18.908079	169.254.1.1	169.254.60.193	TCP
102	18.908133	169.254.60.193	169.254.1.1	TCP
103	18.914084	169.254.1.1	169.254.60.193	TCP
104	18.916081	169.254.1.1	169.254.60.193	TCP
105	18.916137	169.254.60.193	169.254.1.1	TCP
106	18.922082	169.254.1.1	169.254.60.193	TCP
107	18.926088	169.254.1.1	169.254.60.193	TCP
108	18.926151	169.254.60.193	169.254.1.1	TCP
109	18.935082	169.254.1.1	169.254.60.193	TCP
110	18.939077	169.254.1.1	169.254.60.193	TCP
111	18.939117	169.254.60.193	169.254.1.1	TCP
112	18.945081	169.254.1.1	169.254.60.193	TCP
113	18.948083	169.254.1.1	169.254.60.193	TCP
114	18.948131	169.254.60.193	169.254.1.1	TCP
115	18.956087	169.254.1.1	169.254.60.193	TCP
116	18.957080	169.254.1.1	169.254.60.193	HTTP
117	18.957142	169.254.60.193	169.254.1.1	TCP
118	18.962259	169.254.60.193	169.254.1.1	TCP
119	18.964085	169.254.1.1	169.254.60.193	TCP
120	20.022145	0.0.0.0	255.255.255.255	DHCP
121	22.024260	0.0.0.0	255.255.255.255	DHCP

Figura 5.3 Paquetes del 80 al 121 capturados en la transmisión

Para la transmisión se hizo un muestreo en un tiempo de 22 segundos donde se capturaron un total de 121 paquetes. Como podemos observar , empezamos secuencialmente con los paquetes denotados en color celeste en la tabla que son de broadcast; es decir, se transmite un paquete multidestino hasta que obtengamos un resultado favorable de la tarjeta.

A continuación tenemos una trama de un paquete DHCP cuya longitud es de 342 bytes; la trama comienza con la dirección MAC de broadcast destino FF:FF:FF:FF:FF:FF y la dirección MAC de la fuente donde se muestra primeramente la dirección MAC de la tarjeta 00:04:a3:00:00:00.

Observamos también el tipo de trama que para nuestro caso es IP (0x800); la versión del protocolo de Internet que es 4; la longitud de la trama que es 52, 034 en hexadecimal; el tiempo de vida de 128, 64 en hexadecimal; el checksum de la cabecera que en nuestro caso es 4CBB, entre los parámetros más importantes de la trama.

DHCP

```
0000  ff ff ff ff ff ff 00 04  a3 00 00 00 08 00 45 00        ....... ......E.

0010  01 48 08 eb 00 00 64 11  4c bb 00 00 00 00 ff ff     .H....d. L.......

0020  ff ff 00 44 00 43 01 34  00 00 01 01 06 00 12 23     ...D.C.4 .......#

0030  34 56 00 00 80 00 00 00  00 00 00 00 00 00 00 00   4V...... ........
```

```
0040  00 00 00 00 00 00 00 04  a3 00 00 00 00 00 00 00   ........ ........
0050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0090  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00    ........ ........
0100  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0110  00 00 00 00 00 00 63 82  53 63 35 01 01 37 04 01    ...c. Sc5..7..
0120  03 06 0c ff 00 00 00 00  00 00 00 00 00 00 00 00     ........ ........
0130  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
```

```
0140  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........

0150  00 00 00 00 00 00
```

Así mismo podemos examinar una trama TCP cuya longitud es de 66 bytes la trama comienza con la dirección MAC de destino 00:04:A3:00:00:00 y la dirección MAC de la fuente donde se muestra primeramente la dirección MAC de la tarjeta 00:1E:90:8E:A4:6E.

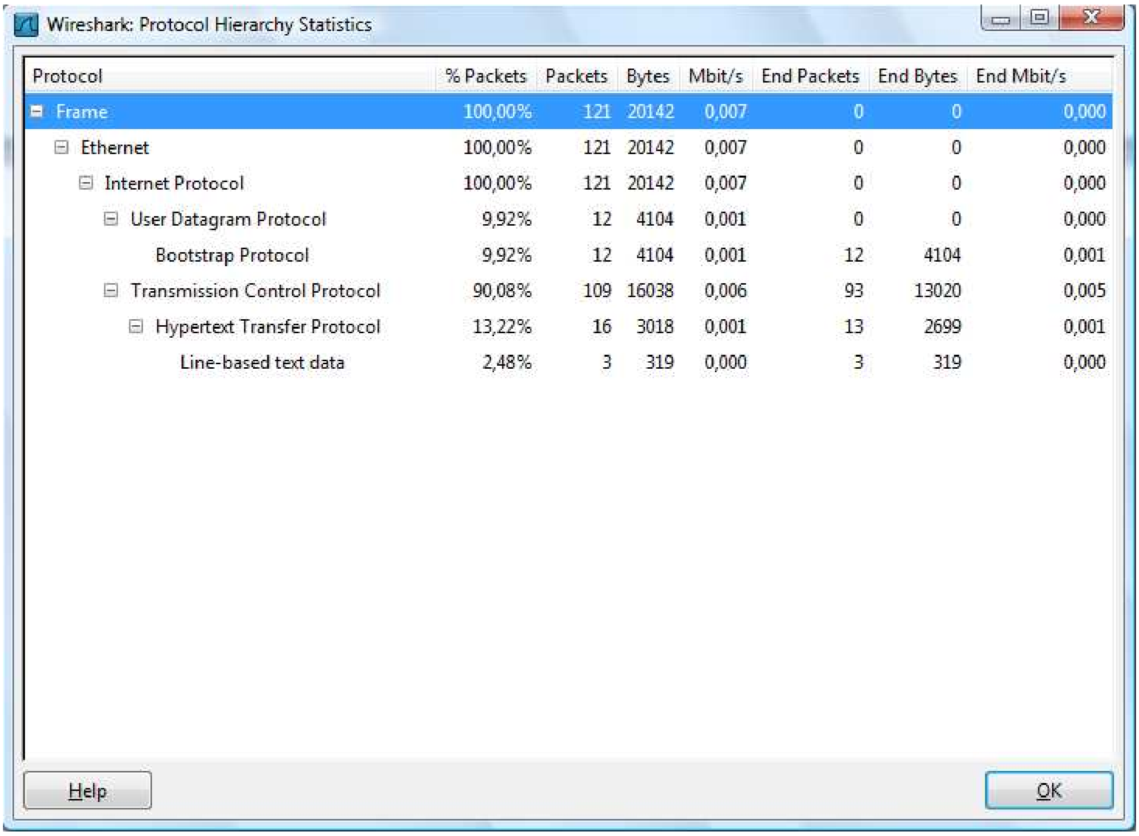
Observamos también, el tipo de trama que para nuestro caso es IP (0x800); la versión del protocolo de Internet que es 4, la longitud de la trama que es 328, 0148 en hexadecimal; el tiempo de vida de 100, 80 en hexadecimal; la dirección IP fuente 169.254.60.193, A9:FE:3C:C1 en hexadecimal y la dirección IP destino 169.254.1.1, A9:FE:01:01 en hexadecimal; el respectivo checksum para de detección de errores, 9D:31, entre los parámetros más importantes

TCP

```
0000  00 04 a3 00 00 00 00 1e  90 8e a4 6e 08 00 45 00   ........ ...n..E.

0010  00 34 02 29 40 00 80 06  00 00 a9 fe 3c c1 a9 fe   .4.)@... ....<...

0020  01 01 c2 eb 00 50 9e 88  be 61 00 00 00 00 80 02   .....P.. .a......

0030  20 00 9d 31 00 00 02 04  05 b4 01 03 03 02 01 01    ..1.... ........
```

0040 04 02 ..

En el siguiente gráfico observamos, la jerarquía de los protocolos utilizados en la transmisión de paquetes en los que, como las cifras lo indican, el protocolo Ethernet se encuentra en el 100% de los paquetes, el protocolo Internet también se encuentra en un 100%, el UDP en un 9.92%, el TCP en un 90.08% y el HTTP en un 13,22% entre los más importantes.



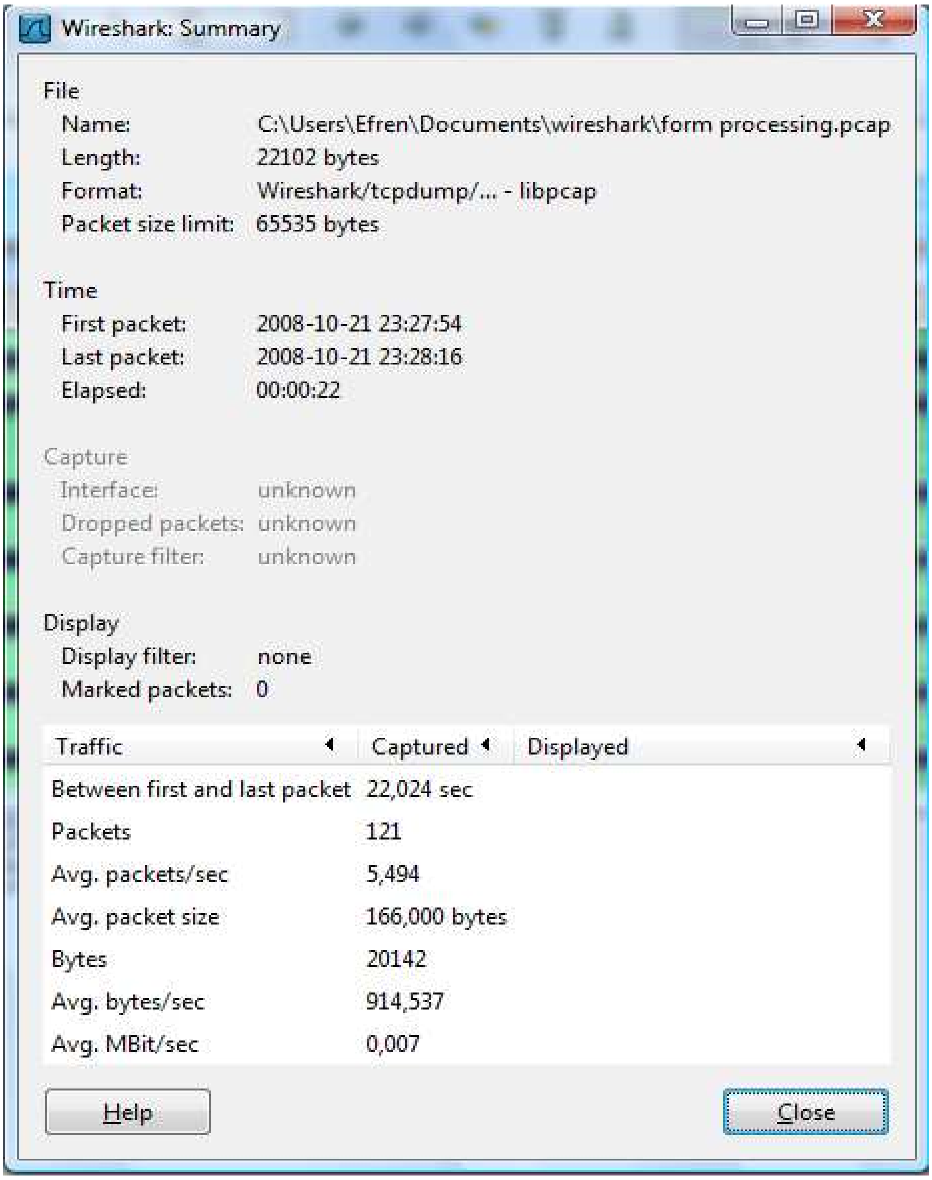
Protocol	% Packets	Packets	Bytes	Mbit/s	End Packets	End Bytes	End Mbit/s
Frame	100,00%	121	20142	0,007	0	0	0,000
Ethernet	100,00%	121	20142	0,007	0	0	0,000
Internet Protocol	100,00%	121	20142	0,007	0	0	0,000
User Datagram Protocol	9,92%	12	4104	0,001	0	0	0,000
Bootstrap Protocol	9,92%	12	4104	0,001	12	4104	0,001
Transmission Control Protocol	90,08%	109	16038	0,006	93	13020	0,005
Hypertext Transfer Protocol	13,22%	16	3018	0,001	13	2699	0,001
Line-based text data	2,48%	3	319	0,000	3	319	0,000

Figura 5.4 Jerarquía de Protocolos Utilizados en Transmisión

5.1.1 Resultados Finales

El software wireshark nos da la facilidad de observar una tabla en la que muestra ciertos parámetros generales de transmisión como el tiempo entre el primer y último paquete capturado que para nuestro caso es 22.02 segundos, el número de paquetes capturados, 121 y su respectivo promedio por segundo que es de 5.49 paquetes/seg.

Tenemos también el promedio de tamaño de los paquetes en base a todos los capturado que es de 166 bytes; el número de bytes total transmitidos 20142 y sus promedios en diferentes unidades, 914.53 bytes/seg y 0.007 MBit/seg.



File
Name: C:\Users\Efren\Documents\wireshark\form processing.pcap
Length: 22102 bytes
Format: Wireshark/tcpdump/... - libpcap
Packet size limit: 65535 bytes

Time
First packet: 2008-10-21 23:27:54
Last packet: 2008-10-21 23:28:16
Elapsed: 00:00:22

Capture
Interface: unknown
Dropped packets: unknown
Capture filter: unknown

Display
Display filter: none
Marked packets: 0

Traffic	Captured	Displayed
Between first and last packet	22,024 sec	
Packets	121	
Avg. packets/sec	5,494	
Avg. packet size	166,000 bytes	
Bytes	20142	
Avg. bytes/sec	914,537	
Avg. MBit/sec	0,007	

Figura 5.5 Parámetros Generales de Transmisión

Por último, destacamos el gráfico generado por el mismo software en el que podemos observar: en el eje **Y** el número de paquetes transmitidos y en el eje **X:** el tiempo en segundos durante el cual fueron transmitidos.

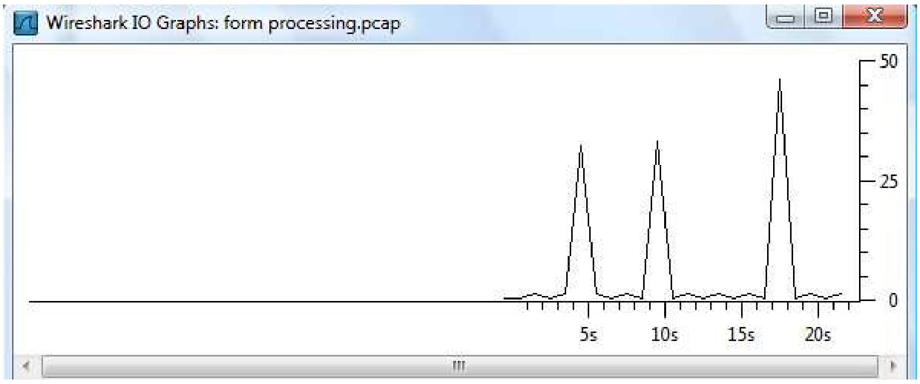
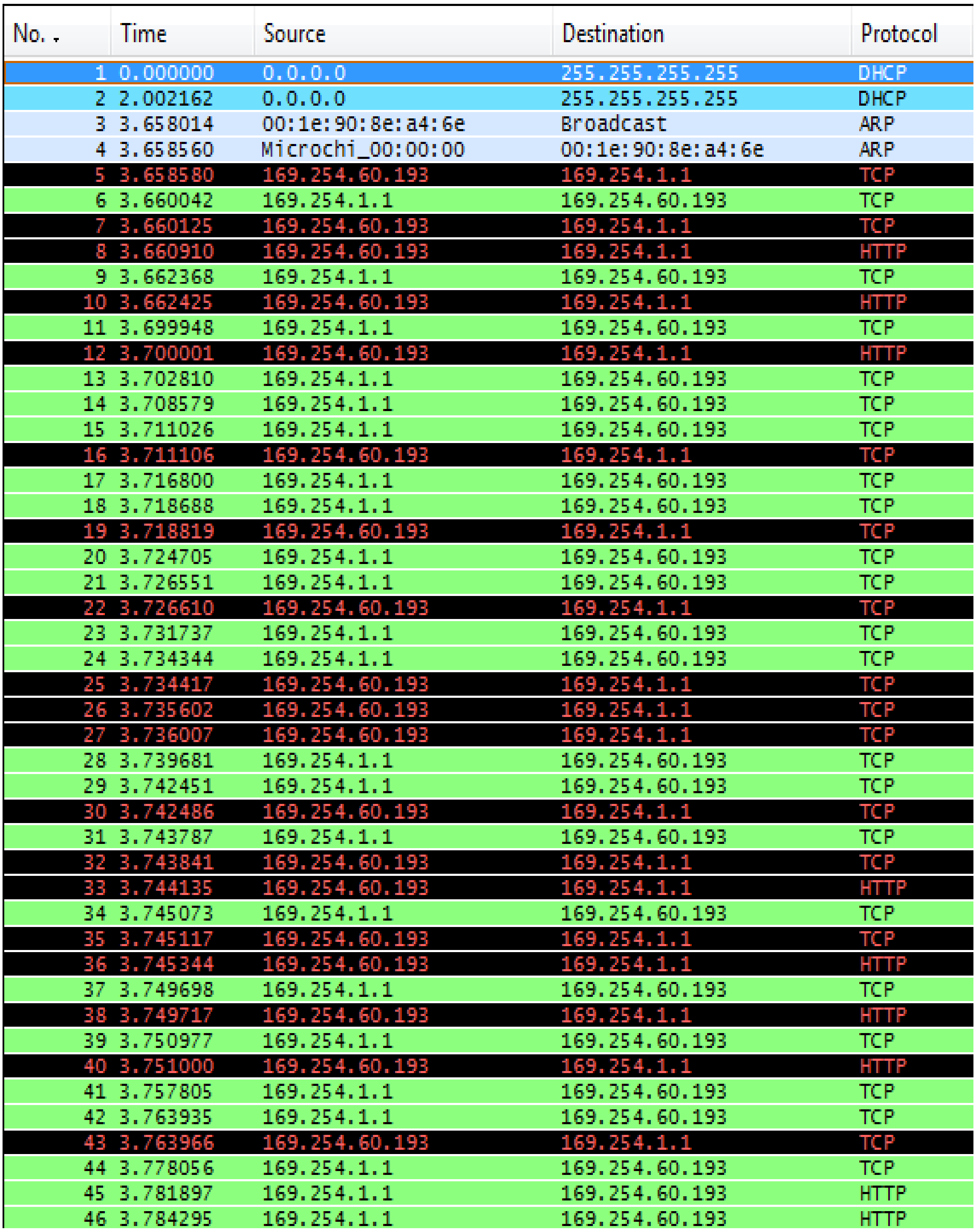


Figura 5.6 Gráfico de Paquetes vs. Tiempo de Transmisión

5.2 Recepción

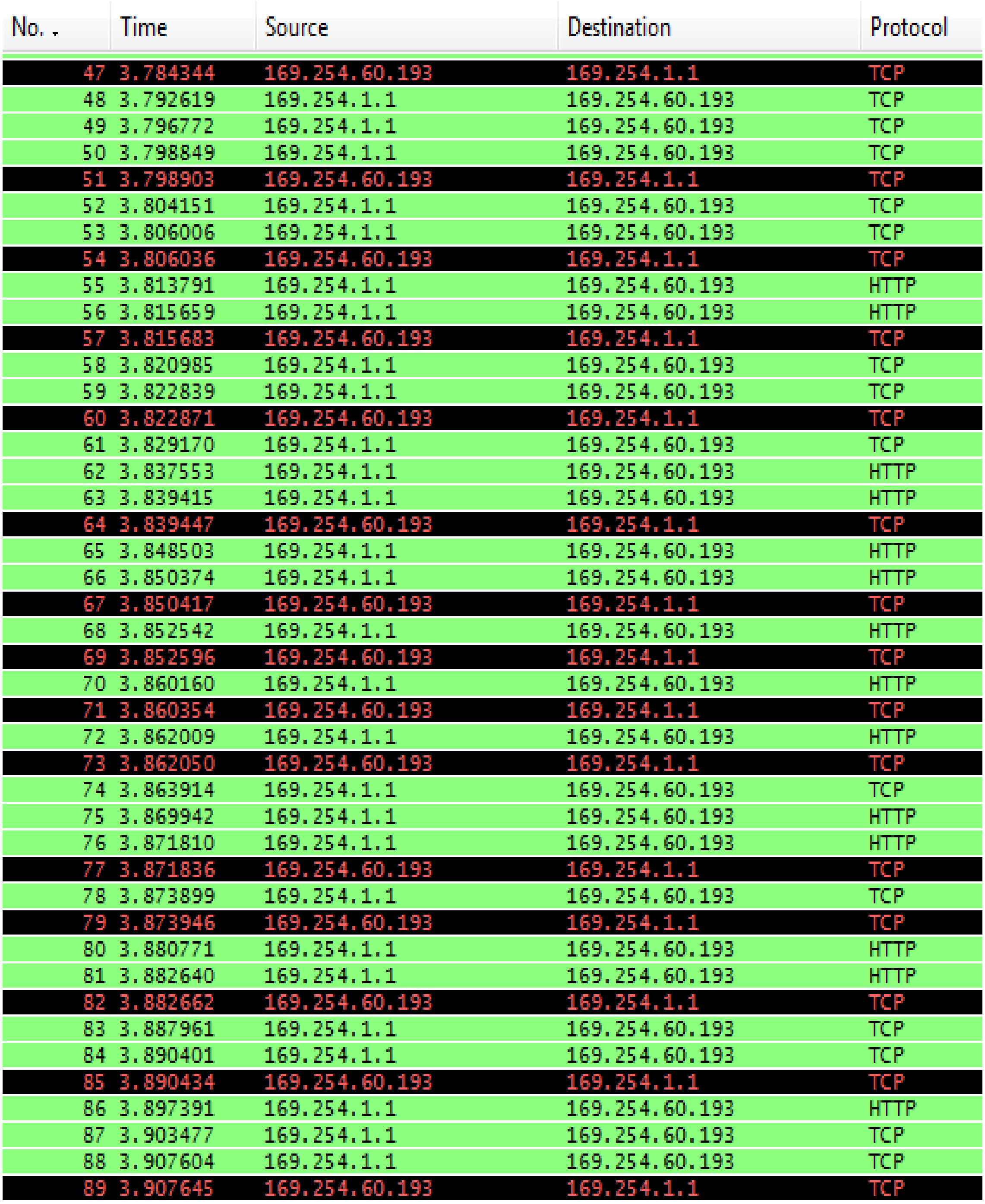
Al realizar la recepción de datos, pudimos obtener los siguientes resultados: en cuanto al conteo de paquetes que analizamos a continuación.

Primeramente, el software realizó la captura de paquetes que vemos en la siguiente figura:



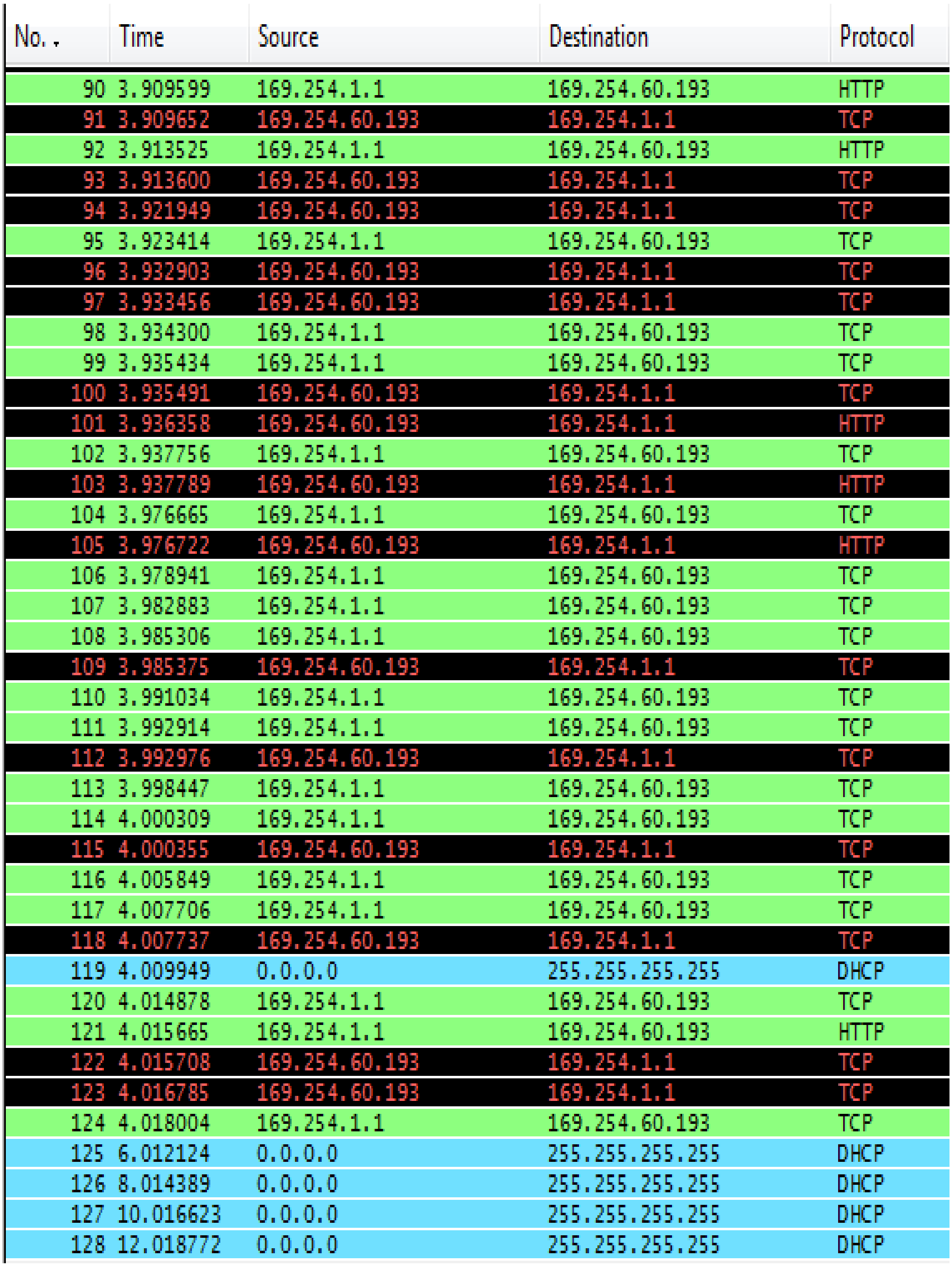
No.	Time	Source	Destination	Protocol
1	0.000000	0.0.0.0	255.255.255.255	DHCP
2	2.002162	0.0.0.0	255.255.255.255	DHCP
3	3.658014	00:1e:90:8e:a4:6e	Broadcast	ARP
4	3.658560	Microchi_00:00:00	00:1e:90:8e:a4:6e	ARP
5	3.658580	169.254.60.193	169.254.1.1	TCP
6	3.660042	169.254.1.1	169.254.60.193	TCP
7	3.660125	169.254.60.193	169.254.1.1	TCP
8	3.660910	169.254.60.193	169.254.1.1	HTTP
9	3.662368	169.254.1.1	169.254.60.193	TCP
10	3.662425	169.254.60.193	169.254.1.1	HTTP
11	3.699948	169.254.1.1	169.254.60.193	TCP
12	3.700001	169.254.60.193	169.254.1.1	HTTP
13	3.702810	169.254.1.1	169.254.60.193	TCP
14	3.708579	169.254.1.1	169.254.60.193	TCP
15	3.711026	169.254.1.1	169.254.60.193	TCP
16	3.711106	169.254.60.193	169.254.1.1	TCP
17	3.716800	169.254.1.1	169.254.60.193	TCP
18	3.718688	169.254.1.1	169.254.60.193	TCP
19	3.718819	169.254.60.193	169.254.1.1	TCP
20	3.724705	169.254.1.1	169.254.60.193	TCP
21	3.726551	169.254.1.1	169.254.60.193	TCP
22	3.726610	169.254.60.193	169.254.1.1	TCP
23	3.731737	169.254.1.1	169.254.60.193	TCP
24	3.734344	169.254.1.1	169.254.60.193	TCP
25	3.734417	169.254.60.193	169.254.1.1	TCP
26	3.735602	169.254.60.193	169.254.1.1	TCP
27	3.736007	169.254.60.193	169.254.1.1	TCP
28	3.739681	169.254.1.1	169.254.60.193	TCP
29	3.742451	169.254.1.1	169.254.60.193	TCP
30	3.742486	169.254.60.193	169.254.1.1	TCP
31	3.743787	169.254.1.1	169.254.60.193	TCP
32	3.743841	169.254.60.193	169.254.1.1	TCP
33	3.744135	169.254.60.193	169.254.1.1	HTTP
34	3.745073	169.254.1.1	169.254.60.193	TCP
35	3.745117	169.254.60.193	169.254.1.1	TCP
36	3.745344	169.254.60.193	169.254.1.1	HTTP
37	3.749698	169.254.1.1	169.254.60.193	TCP
38	3.749717	169.254.60.193	169.254.1.1	HTTP
39	3.750977	169.254.1.1	169.254.60.193	TCP
40	3.751000	169.254.60.193	169.254.1.1	HTTP
41	3.757805	169.254.1.1	169.254.60.193	TCP
42	3.763935	169.254.1.1	169.254.60.193	TCP
43	3.763966	169.254.60.193	169.254.1.1	TCP
44	3.778056	169.254.1.1	169.254.60.193	TCP
45	3.781897	169.254.1.1	169.254.60.193	HTTP
46	3.784295	169.254.1.1	169.254.60.193	HTTP

Figura 5.7 Paquetes del 1 al 46 capturados en la recepción



No.	Time	Source	Destination	Protocol
47	3.784344	169.254.60.193	169.254.1.1	TCP
48	3.792619	169.254.1.1	169.254.60.193	TCP
49	3.796772	169.254.1.1	169.254.60.193	TCP
50	3.798849	169.254.1.1	169.254.60.193	TCP
51	3.798903	169.254.60.193	169.254.1.1	TCP
52	3.804151	169.254.1.1	169.254.60.193	TCP
53	3.806006	169.254.1.1	169.254.60.193	TCP
54	3.806036	169.254.60.193	169.254.1.1	TCP
55	3.813791	169.254.1.1	169.254.60.193	HTTP
56	3.815659	169.254.1.1	169.254.60.193	HTTP
57	3.815683	169.254.60.193	169.254.1.1	TCP
58	3.820985	169.254.1.1	169.254.60.193	TCP
59	3.822839	169.254.1.1	169.254.60.193	TCP
60	3.822871	169.254.60.193	169.254.1.1	TCP
61	3.829170	169.254.1.1	169.254.60.193	TCP
62	3.837553	169.254.1.1	169.254.60.193	HTTP
63	3.839415	169.254.1.1	169.254.60.193	HTTP
64	3.839447	169.254.60.193	169.254.1.1	TCP
65	3.848503	169.254.1.1	169.254.60.193	HTTP
66	3.850374	169.254.1.1	169.254.60.193	HTTP
67	3.850417	169.254.60.193	169.254.1.1	TCP
68	3.852542	169.254.1.1	169.254.60.193	HTTP
69	3.852596	169.254.60.193	169.254.1.1	TCP
70	3.860160	169.254.1.1	169.254.60.193	HTTP
71	3.860354	169.254.60.193	169.254.1.1	TCP
72	3.862009	169.254.1.1	169.254.60.193	HTTP
73	3.862050	169.254.60.193	169.254.1.1	TCP
74	3.863914	169.254.1.1	169.254.60.193	TCP
75	3.869942	169.254.1.1	169.254.60.193	HTTP
76	3.871810	169.254.1.1	169.254.60.193	HTTP
77	3.871836	169.254.60.193	169.254.1.1	TCP
78	3.873899	169.254.1.1	169.254.60.193	TCP
79	3.873946	169.254.60.193	169.254.1.1	TCP
80	3.880771	169.254.1.1	169.254.60.193	HTTP
81	3.882640	169.254.1.1	169.254.60.193	HTTP
82	3.882662	169.254.60.193	169.254.1.1	TCP
83	3.887961	169.254.1.1	169.254.60.193	TCP
84	3.890401	169.254.1.1	169.254.60.193	TCP
85	3.890434	169.254.60.193	169.254.1.1	TCP
86	3.897391	169.254.1.1	169.254.60.193	HTTP
87	3.903477	169.254.1.1	169.254.60.193	TCP
88	3.907604	169.254.1.1	169.254.60.193	TCP
89	3.907645	169.254.60.193	169.254.1.1	TCP

Figura 5.8 Paquetes del 47 al 89 capturados en la recepción



No.	Time	Source	Destination	Protocol
90	3.909599	169.254.1.1	169.254.60.193	HTTP
91	3.909652	169.254.60.193	169.254.1.1	TCP
92	3.913525	169.254.1.1	169.254.60.193	HTTP
93	3.913600	169.254.60.193	169.254.1.1	TCP
94	3.921949	169.254.60.193	169.254.1.1	TCP
95	3.923414	169.254.1.1	169.254.60.193	TCP
96	3.932903	169.254.60.193	169.254.1.1	TCP
97	3.933456	169.254.60.193	169.254.1.1	TCP
98	3.934300	169.254.1.1	169.254.60.193	TCP
99	3.935434	169.254.1.1	169.254.60.193	TCP
100	3.935491	169.254.60.193	169.254.1.1	TCP
101	3.936358	169.254.60.193	169.254.1.1	HTTP
102	3.937756	169.254.1.1	169.254.60.193	TCP
103	3.937789	169.254.60.193	169.254.1.1	HTTP
104	3.976665	169.254.1.1	169.254.60.193	TCP
105	3.976722	169.254.60.193	169.254.1.1	HTTP
106	3.978941	169.254.1.1	169.254.60.193	TCP
107	3.982883	169.254.1.1	169.254.60.193	TCP
108	3.985306	169.254.1.1	169.254.60.193	TCP
109	3.985375	169.254.60.193	169.254.1.1	TCP
110	3.991034	169.254.1.1	169.254.60.193	TCP
111	3.992914	169.254.1.1	169.254.60.193	TCP
112	3.992976	169.254.60.193	169.254.1.1	TCP
113	3.998447	169.254.1.1	169.254.60.193	TCP
114	4.000309	169.254.1.1	169.254.60.193	TCP
115	4.000355	169.254.60.193	169.254.1.1	TCP
116	4.005849	169.254.1.1	169.254.60.193	TCP
117	4.007706	169.254.1.1	169.254.60.193	TCP
118	4.007737	169.254.60.193	169.254.1.1	TCP
119	4.009949	0.0.0.0	255.255.255.255	DHCP
120	4.014878	169.254.1.1	169.254.60.193	TCP
121	4.015665	169.254.1.1	169.254.60.193	HTTP
122	4.015708	169.254.60.193	169.254.1.1	TCP
123	4.016785	169.254.60.193	169.254.1.1	TCP
124	4.018004	169.254.1.1	169.254.60.193	TCP
125	6.012124	0.0.0.0	255.255.255.255	DHCP
126	8.014389	0.0.0.0	255.255.255.255	DHCP
127	10.016623	0.0.0.0	255.255.255.255	DHCP
128	12.018772	0.0.0.0	255.255.255.255	DHCP

Figura 5.9 Paquetes del 90 al 128 capturados en la recepción

Para la transmisión, se hizo un muestreo en un tiempo de 12 segundos donde se capturaron un total de 128 paquetes. Como podemos observar, empezamos secuencialmente con los paquetes denotados en color celeste en la tabla que son de broadcast, es decir,se transmite un paquete multidestino hasta que obtengamos un resultado favorable de la tarjeta.

Posteriormente encontramos un paquete ARP en el que tenemos la repuesta del dispositivo, enviándonos su dirección MAC en el siguiente paquete, 00:1E:90:8E:A4:6E y su dirección ip 169.254.1.1

A continuación, tenemos una trama de un paquete DHCP cuya longitud es de 342 bytes; la trama comienza con la dirección MAC de broadcast destino FF:FF:FF:FF:FF:FF y la dirección MAC de la fuente donde se muestra primeramente la dirección MAC de la tarjeta 00:04:a3:00:00:00.

Observamos también, el tipo de trama que para nuestro caso es IP (0x800); la versión del protocolo de Internet que es 4, la longitud de la trama que es 20, 045 en hexadecimal, el tiempo de vida de 100, 64 en hexadecimal; y el checksum de la cabecera que en nuestro caso es 4DD7, entre los parámetros más importantes de la trama.

DHCP

```
0000  ff ff ff ff ff ff 00 04  a3 00 00 00 08 00 45 00   ........ ......E.

0010  01 48 07 cf 00 00 64 11  4d d7 00 00 00 00 ff ff   .H....d. M.......

0020  ff ff 00 44 00 43 01 34  00 00 01 01 06 00 12 23   ...D.C.4 .......#

0030  34 56 00 00 80 00 00 00  00 00 00 00 00 00 00 00   4V...... ........

0040  00 00 00 00 00 00 00 04  a3 00 00 00 00 00 00 00   ........ ........

0050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........

0060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........

0070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........

0080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........

0090  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........

00a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........

00b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........

00c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........

00d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........

00e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........

00f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........

0100  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........

0110  00 00 00 00 00 00 63 82  53 63 35 01 01 37 04 01   ......c. Sc5..7..
```

```
0120  03 06 0c ff 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........

0130  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........

0140  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........

0150  00 00 00 00 00 00                                  ......
```

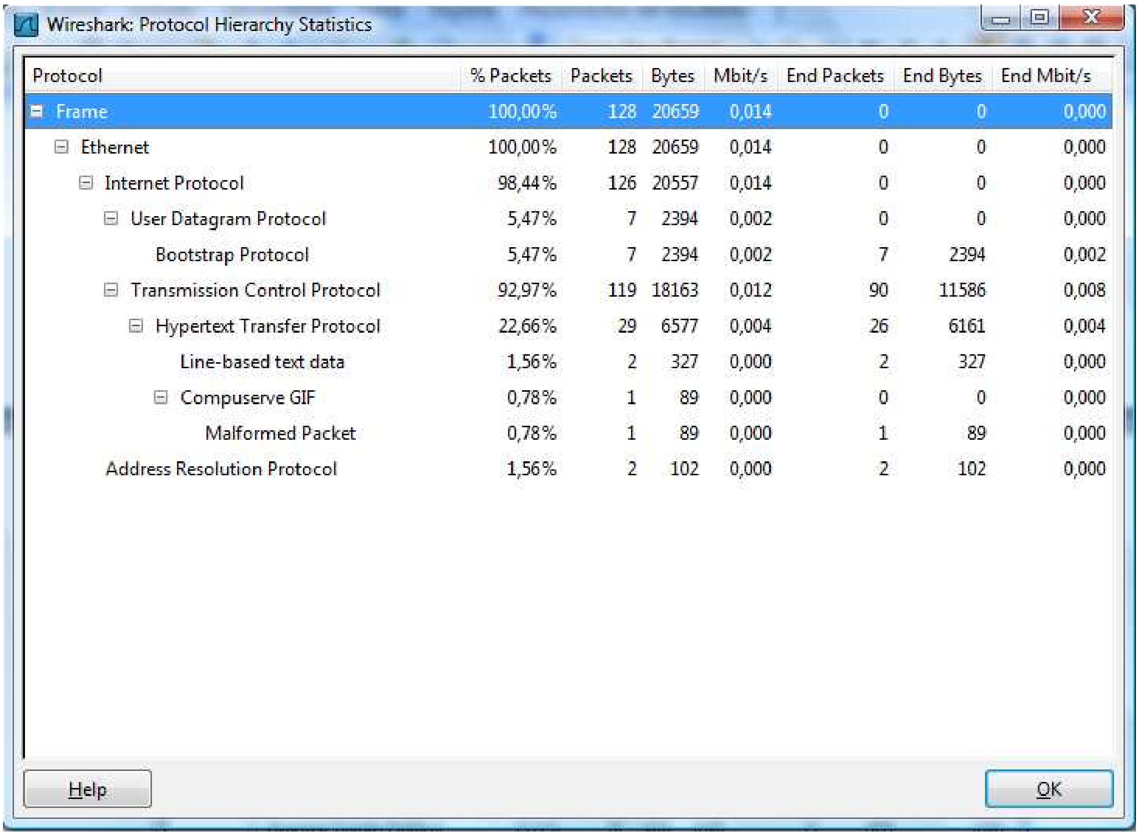
Así mismo, podemos examinar una trama TCP cuya longitud es de 66 bytes la trama comienza con la dirección MAC de destino 00:04:A3:00:00:00 y la dirección MAC de la fuente donde se muestra primeramente la dirección MAC de la tarjeta 00:1E:90:8E:A4:6E.

Así mismo se puede apreciar, el tipo de trama que para nuestro caso es IP (0x800); la versión del protocolo de Internet que es 4; la longitud de la trama que es 328, 0148 en hexadecimal; el tiempo de vida de 100, 80 en hexadecimal; la dirección IP fuente 169.254.1.1 la dirección IP destino 169.254.60.193; el respectivo checksum para detección de errores, BD3C, entre los parámetros más importantes

TCP

```
0000  00 1e 90 8e a4 6e 00 04  a3 00 00 00 08 00 45 00   .....n.. ......E.

0010  00 2c 07 d1 00 00 64 06  bd 3c a9 fe 01 01 a9 fe   .,....d. .<......

0020  3c c1 00 50 c2 ac 4b e1  78 60 b2 b7 93 22 60 12   <..P..K. x`..."`.

0030  00 05 3c da 00 00 02 04  02 14 00 00               ..<..... ....
```

En el siguiente gráfico se puede apreciar la jerarquía de los protocolos utilizados en la transmisión de paquetes en los que, como las cifras lo indican, el protocolo Ethernet se encuentra en el 100% de los paquetes, el protocolo Internet se encuentra en un 98.44%, el UDP en un 5.47%, el TCP en un 92.97% y el HTTP en un 22.66%. Además encontramos el protocolo ARP en un 1.56% entre los más importantes.



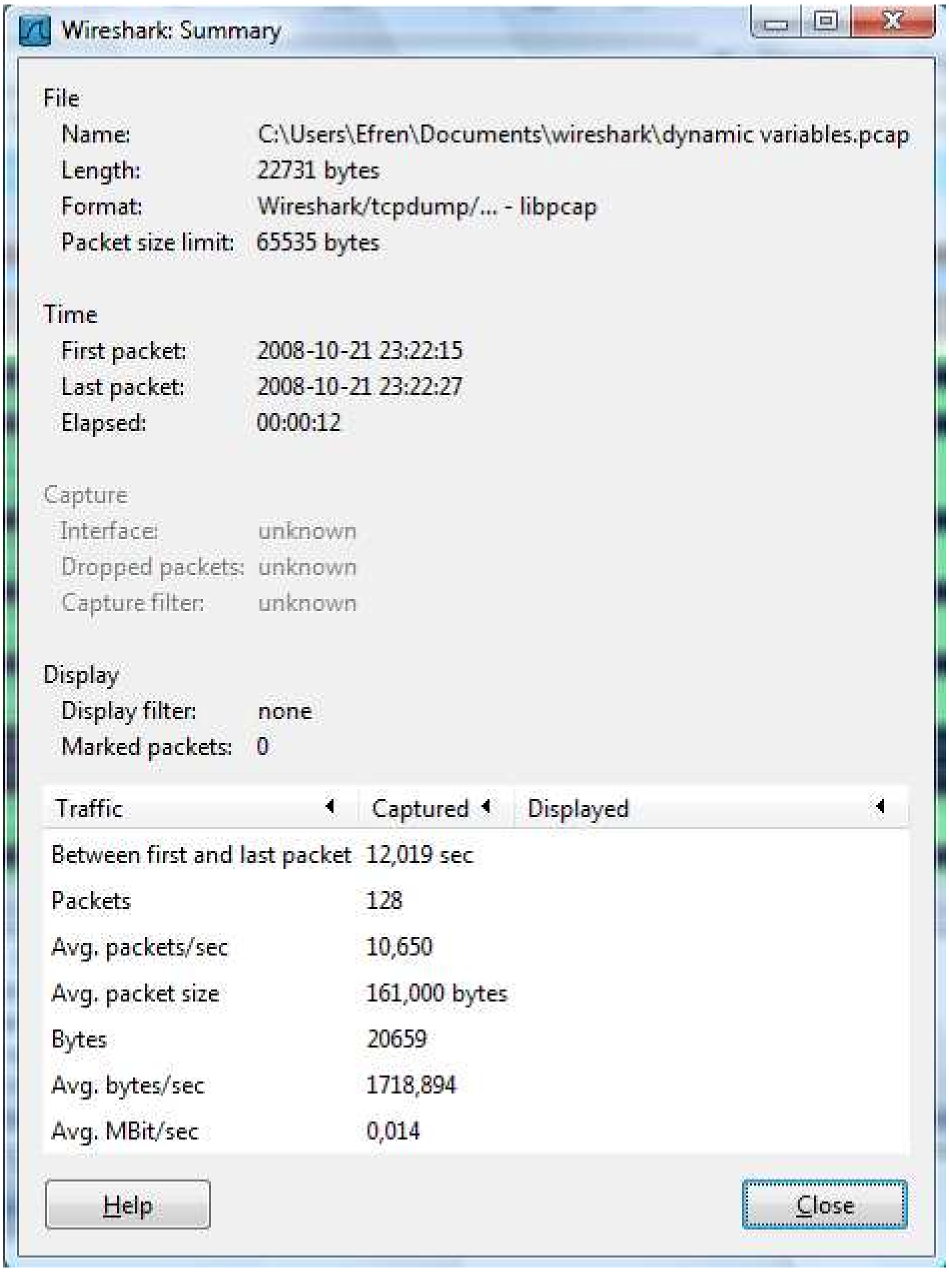
Protocol	% Packets	Packets	Bytes	Mbit/s	End Packets	End Bytes	End Mbit/s
⊟ Frame	100,00%	128	20659	0,014	0	0	0,000
⊟ Ethernet	100,00%	128	20659	0,014	0	0	0,000
⊟ Internet Protocol	98,44%	126	20557	0,014	0	0	0,000
⊟ User Datagram Protocol	5,47%	7	2394	0,002	0	0	0,000
Bootstrap Protocol	5,47%	7	2394	0,002	7	2394	0,002
⊟ Transmission Control Protocol	92,97%	119	18163	0,012	90	11586	0,008
⊟ Hypertext Transfer Protocol	22,66%	29	6577	0,004	26	6161	0,004
Line-based text data	1,56%	2	327	0,000	2	327	0,000
⊟ Compuserve GIF	0,78%	1	89	0,000	0	0	0,000
Malformed Packet	0,78%	1	89	0,000	1	89	0,000
Address Resolution Protocol	1,56%	2	102	0,000	2	102	0,000

Figura 5.10 Jerarquía de Protocolos utilizados en Recepción

5.3 Resultados Finales

El software wireshark, nos da la facilidad de observar una tabla en la que muestra ciertos parámetros generales de recepción como el tiempo entre el primer y último paquete capturado que para nuestro caso es 12,019 segundos, el número de paquetes capturados, 128 y su respectivo promedio por segundo que es de 10,650 paquetes/seg.

Tenemos también el promedio de tamaño de los paquetes en base a todos los capturado que es de 161 bytes; el número de bytes total transmitidos, 20659 y sus promedios en diferentes unidades, 1718.894 bytes/seg y 0.014 MBit/seg.



File
Name: C:\Users\Efren\Documents\wireshark\dynamic variables.pcap
Length: 22731 bytes
Format: Wireshark/tcpdump/... - libpcap
Packet size limit: 65535 bytes

Time
First packet: 2008-10-21 23:22:15
Last packet: 2008-10-21 23:22:27
Elapsed: 00:00:12

Capture
Interface: unknown
Dropped packets: unknown
Capture filter: unknown

Display
Display filter: none
Marked packets: 0

Traffic	Captured	Displayed
Between first and last packet	12,019 sec	
Packets	128	
Avg. packets/sec	10,650	
Avg. packet size	161,000 bytes	
Bytes	20659	
Avg. bytes/sec	1718,894	
Avg. MBit/sec	0,014	

Figura 5.11 Parámetros Generales de Recepción

Por último, destacamos el gráfico generado por el mismo software en el que podemos observar lo siguiente : en el eje **Y** el número de paquetes transmitidos y en el eje **x** el tiempo en segundos durante el cual fueron transmitidos.

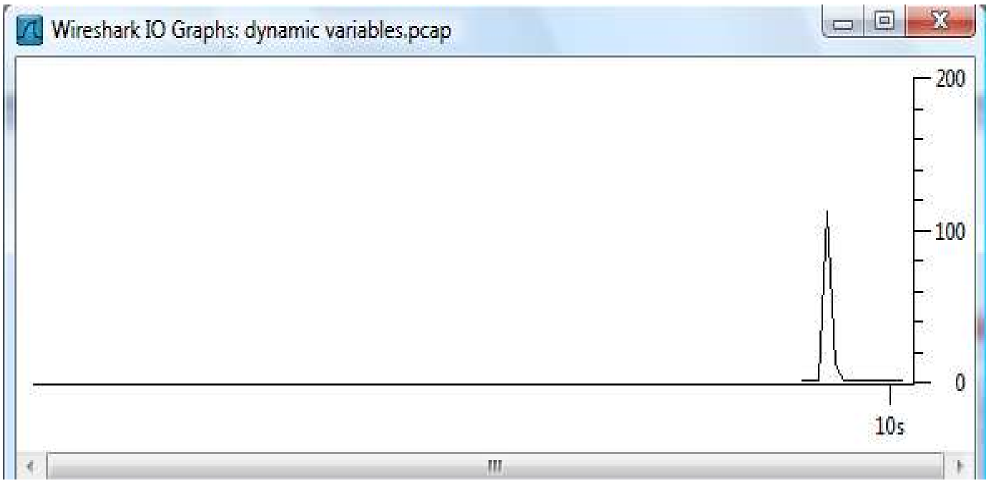
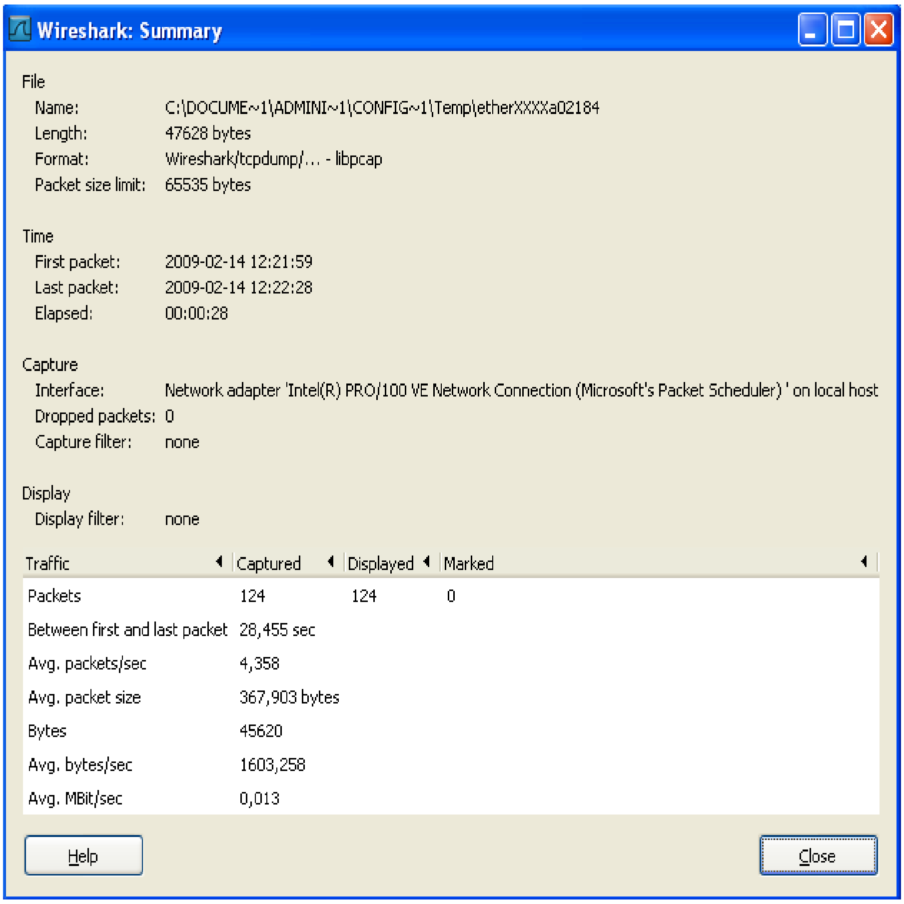


Figura 5.12 Gráfico de Paquetes vs. Tiempo de Recepción

Si realizamos las mismas mediciones con el mismo software en una tarjeta de red Intel® PRO/100 VE Network Connection, obtenemos la siguiente tabla de resultados:



```
File
  Name:             C:\DOCUME~1\ADMINI~1\CONFIG~1\Temp\etherXXXXa02184
  Length:           47628 bytes
  Format:           Wireshark/tcpdump/... - libpcap
  Packet size limit: 65535 bytes

Time
  First packet:     2009-02-14 12:21:59
  Last packet:      2009-02-14 12:22:28
  Elapsed:          00:00:28

Capture
  Interface:        Network adapter 'Intel(R) PRO/100 VE Network Connection (Microsoft's Packet Scheduler) ' on local host
  Dropped packets:  0
  Capture filter:   none

Display
  Display filter:   none

Traffic                      Captured       Displayed   Marked
Packets                      124            124         0
Between first and last packet 28,455 sec
Avg. packets/sec             4,358
Avg. packet size             367,903 bytes
Bytes                        45620
Avg. bytes/sec               1603,258
Avg. MBit/sec                0,013
```

Figura 5.13 Parámetros de Transmisión en Tarjeta de Red Intel(R) PRO/100 VE Network Connection

Como podemos observar tenemos un promedio de transmisión de 4,358 paquetes/segundo con paquetes con un promedio de 367,903 bytes; lo que finalmente nos da un promedio de 1603,258 bytes/segundo y/o 0,013 Mbit/segundo; resultados que coinciden con los parámetros de transmisión y recepción de nuestra tarjeta controladora, teniendo en cuenta pequeñas variaciones que dependen del tamaño de página web utilizado en nuestro proyecto con el tamaño de las páginas que encontramos actualmente en la web; además debemos considerar que el microcontrolador utilizado posee una tasa de transferencia de 10 Mbit/seg. y la tarjeta de red utilizada en la computadora es de 10/100 Mbit/seg.

CAPÍTULO 6

6. ANÁLISIS DE COSTOS

Para este capítulo, hemos hecho un análisis detallado de los costos de nuestras tarjetas controladora y de potencia respectivamente, que al final nos permitirá dar una conclusión en cuanto al mismo.

6.1 Cuadro de Precios

Los siguientes cuadros de Precios detallan los elementos utilizados, las cantidades de los mismos, el precio por unidad y el precio total tomando en cuenta la cantidad de elementos utilizados.

6.1.1 Elementos de la Tarjeta de Control

Descripción	Cantidad	Precio Unitario	Precio Total
PIC18F97J60	1	$15.50	$15.50
PROGRAMADOR SERIAL MPLAB ICD2	1	$367.75	$367.75
LCD LCMSO1602DTR/M	1	$16.00	$16.00
CONECTOR MODULAR RJ45	1	$5.50	$5.50
MEMORIA EEPROM 25LC256	1	$9.80	$9.80
MAX3232	1	$4.00	$4.00
REGULADOR LM2940	1	$2.00	$2.00
REGULADOR 3.3 V.	1	$8.50	$8.50
CRISTAL DE 25 MHZ	1	$1.00	$1.00
PULSADORES	4	$0.30	$1.20
CONECTOR DB9	1	$0.50	$0.50
INDUCTOR	1	$2.50	$2.50
CONECTOR ICD	1	$0.50	$0.50
CAPACITORES 0.1uF	27	$0.25	$6.75
CAPACITORES 33pF	2	$0.25	$0.50
CAPACITORES 1000 uF	2	$0.25	$0.50
CAPACITORES 470Uf	2	$0.25	$0.50
RESISTENCIAS	20	$0.05	$1.00
POTENCIOMETRO 5K	1	$0.30	$0.30
DIODO LED	1	$0.25	$0.25
DIODO 1N4006	1	$0.30	$0.30
TRANSISTOR Q2N3904	1	$0.30	$0.30
JUMPERS DE PASO	9	$0.20	$1.80
PULSADORES	2	$0.30	$0.60
CONECTORES DE 16 PINES	3	$0.30	$0.90
CABLE DE UTP CAT 5	1 metro	$0.30	$0.30
CONECTORES RJ 45	2	$0.30	$0.60
CONECTOR DE ENERGÍA	1	$0.25	$0.25
ADAPTADOR DE 9v. dc	1	$5.00	$5.00
		TOTAL	$453.70

Tabla 6.1 Cuadro de Precios de Elementos de la Tarjeta de Control

Como observamos, se ha gastado un total de $85.95 en los elementos de la tarjeta controladora, valor que consideramos aceptable para la realización de este proyecto.

6.1.2 Elementos del Adaptador

Descripción	**Cantidad**	**Precio Unitario**	**Precio Total**
TRANSFORMADOR DE 12 V. DC	1	$5.00	$5.00
CAPACITOR	1	$0.30	$0.30
CAPACITOR	1	$0.30	$0.30
PUENTE RECTIFICADOR	1	$0.80	$0.80
FUSIBLE	1	$0.25	$0.25
DIODO	1	$0.35	$0.35
LN2803	1	$3.00	$3.00
RESISTENCIAS	4	$0.05	$0.20
RELÉS	4	$1.00	$4.00
DIODOS LEDS	4	$0.25	$1.00
BORNERAS	6	$0.50	$3.00
CONECTOR 16 PINES	1	$0.30	$0.90
		TOTAL	$19.10

Tabla 6.2 Cuadro de Precios de Elementos del Adaptador

En los elementos del adaptador o tarjeta de potencia se gastó un total de $19.10, todos los elementos fueron conseguidos en el mercado local sin ninguna dificultad.

6.1.3 Diseño y Montaje

Tarjeta Controladora

Descripción	Cantidad	Precio Total
Diseño y Elaboración de PCB de la Tarjeta en Altium Designer	1	$58.05
Montaje y Soldada de Elementos	1	$8
	TOTAL	$66.05

Tabla 6.3 Cuadro de Precios de Diseño y Montaje de Tarjeta Controladora

Tarjeta de Potencia

Descripción	Cantidad	Precio Total
Diseño y Elaboración de PCB de la Tarjeta en Altium Designer	1	$30.00
Montaje y Soldada de Elementos	1	$7.50
	TOTAL	$37.50

Tabla 6.4 Cuadro de Precios de Diseño y Montaje de Tarjeta de Potencia

6.2 Resumen de Costos

TARJETA CONTROLADORA	$519.75
ADAPTADOR	$56.60
TOTAL	$576.35

Tabla 6.5 Resumen Total de Costos

En esta sección, vemos el valor económico total utilizado en la elaboración de nuestro proyecto, un valor de $576.35; cabe recalcar, que este valor no incluye ciertos elementos que se adquirieron para realizar las pruebas necesarias para su buen funcionamiento.

ANEXOS

A. DESCRIPCIÓN DE LOS COMPONENTES ELECTRÓNICOS

25LC256

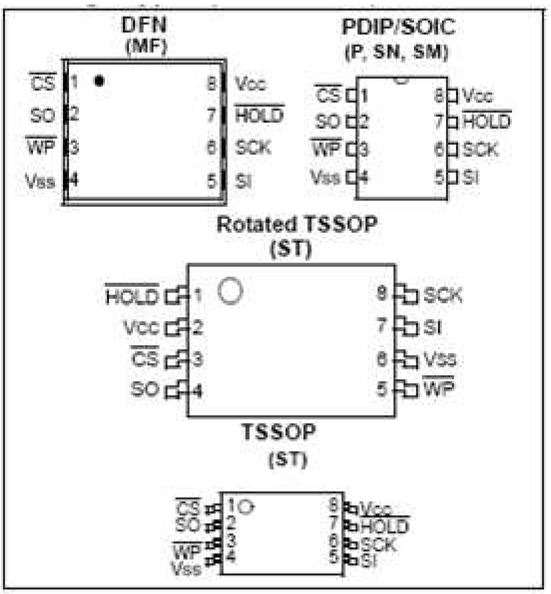
La Tecnología Microchip 25LC256 es una memoria serial eléctricamente borrable (EEPROM) de 256 Kbits

Accedemos a la memoria por una interfase serial compatible con un bus serial. Las señales del bus de datos requeridas son: una señal de reloj (SCK) más datos separados en líneas de entradas y salidas SI y SO. Además accedemos al dispositivo a través de la señal de entrada de seleccionar circuito (CS).

La comunicación hacia el dispositivo puede ser pausada mediante el pin específico (HOLD).

Mientras el dispositivo es pausado, las transiciones en sus entradas serán ignoradas, con excepción de la entrada de selección del dispositivo (CS), permitiendo dar interrupciones de mayor prioridad.

A continuación observamos los diferentes empaquetados del dispositivo con sus respectivas señales de entrada y salida y su respectiva descripción.



Nombre	Función
CS	Entrada de selección de integrado
SO	Salida Serial de datos
WP	Protección de Escritura
VSS	Tierra
SI	Entrada Serial de Datos
SCK	Entrada Serial de Reloj
HOLD	Salida de Mantenimiento
VCC	Voltaje de Alimentación

CARACTERISTICAS ELECTRICAS

Vemos continuación los máximos rangos de operación de la EEPROM 25LC256.

Vcc 6.5V.

Vss para todas las entradas -0.6V. a Vcc + 1.0V.

Temperaturas de Almacenamiento 65ºC a 150ºC

Temperatura ambiente bajo BIAS 40ºC a 125ºC

Protección ESD en todos los pines4Kv.

CARACTERISTICAS DC

			Industrial (I) TA=-40ºC a 85 ºC Vcc=1.8 V. a 5.5 V. Automotriz (E) TA=-40ºC a 125ºC Vcc=2.5 V. a 5.5 V.				
Parámetro No.	**Símbolo**	**Característica**	**Min.**	**Tipo**	**Max.**	**Unidades**	**Condiciones de Prueba**
D001	VIH	Voltaje de entrada de nivel alto	0.7 VCC	-	VCC + 1	V.	
D002	VIL	Voltaje de entrada de nivel bajo	-0.3	-	0.3 VCC	V.	VCC > 2.5 V.
D003	VIL	Voltaje de entrada de nivel bajo	-0.3	-	0.2 VCC	V.	VCC < 2.5 V.
D004	VOL	Voltaje de salidad de nivel bajo	-	-	0.4	V.	IOL = 2.1 mA, VCC = 4.5V.
D005	VOL	Voltaje de salidad de nivel bajo	-	-	0.2	V.	IOL = 1 mA, VCC = 2.5 V.
D006	VOH	Voltaje de Salida de Nivel Alto	Vcc – 0.5	-	-	V.	IOH -400uA
D007	ILI	Corriente de Fuga de Entrada	-	-	+/- 1	uA	CS = VCC, VIN=VSS o VCC
D008	ILO	Corriente de Fuga de Salida	-	-	+/- 1	uA	CS = VCC, VOUT=VSS o VCC

D009	CINT	Capacitancia Interna	-	-	7	pF	TA=25ºC, FCLK=1Mhz. Vcc=5V.
D010	ICC Read	Corriente de Operación	-	2.5 0.5	6 2.5	mA mA	VCC=5.5V.; FCLK=10Mhz; SO=Abierto VCC=2.5V.; FCLK=5Mhz; SO=Abierto
D011	ICC Write	Corriente de Operación	-	0.6 0.15	5 3	mA mA	VCC=5.5 V. VCC=2.5 V.
D012	ICCS	Corriente de Emergencia	-	0.1 -	5 1	uA uA	CS=VCC=5.5V., entradas iguales a VCC o VSS, 125ºC CS=VCC=5.5V., entradas iguales a VCC o VSS, 85ºC

CARACTERISTICAS AC

			Industrial (I) TA=-40ºC a 85 ºC Vcc=1.8 V. a 5.5 V. Automotriz (E) TA=-40ºC a 125ºC Vcc=2.5 V. a 5.5 V.			
Parámetro No.	**Símbolo**	**Característica**	**Min.**	**Max.**	**Unidades**	**Condiciones de Prueba**
1	FCLK	Frecuencia de Reloj	- - -	10 5 3	Mhz Mhz Mhz	4.5V<VCC<5.5V 2.5V<VCC<4.5V 1.8V<VCC<2.5V
2	TCSS	Tiempo de Fijación de CS	50 100 150	- - -	ns. ns. ns.	4.5V<VCC<5.5V 2.5V<VCC<4.5V 1.8V<VCC<2.5V
3	TCSH	Tiempo de mantenimiento de CS	100 200 250	- - -	ns. ns. ns.	4.5V<VCC<5.5V 2.5V<VCC<4.5V 1.8V<VCC<2.5V
4	TCSD	Tiempo de Deshabilitación de CS	50	-	ns	-
5	TSU	Tiempo de Fijación de Datos	10 20 30	- - -	ns. ns. ns.	4.5V<VCC<5.5V 2.5V<VCC<4.5V 1.8V<VCC<2.5V
6	THD	Tiempo de mantenimiento de datos	20 40 50	- - -	ns. ns. ns.	4.5V<VCC<5.5V 2.5V<VCC<4.5V 1.8V<VCC<2.5V
7	TR	Tiempo de Subida de Reloj	-	100	ns	-
8	TF	Tiempo de Bajada de Reloj	-	100	ns	-
9	THI	Tiempo de Nivel Alto de Reloj	50 100 150	- - -	ns. ns. ns.	4.5V<VCC<5.5V 2.5V<VCC<4.5V 1.8V<VCC<2.5V
10	TLO	Tiempo de Nivel Bajo de Reloj	50 100 150	- - -	ns. ns. ns.	4.5V<VCC<5.5V 2.5V<VCC<4.5V 1.8V<VCC<2.5V
11	TCLD	Tiempo de Retardo de Reloj	50	-	ns	-
12	TCLE	Tiempo de Habilitación de Reloj	50	-	ns	-
13	TV	Salida Válida para nivel bajo de Reloj	- - -	50 100 160	ns. ns. ns.	4.5V<VCC<5.5V 2.5V<VCC<4.5V 1.8V<VCC<2.5V

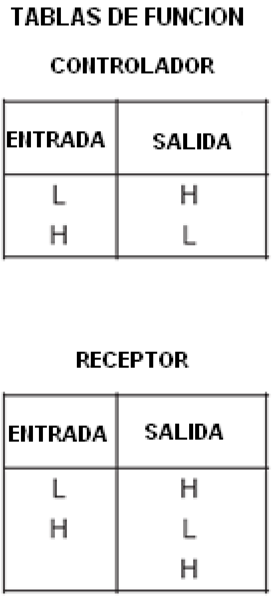
14	THO	Tiempo de mantenimiento de salida	0	-	ns	-
15	TDIS	Tiempo de Des habilitación de salida	- - -	40 80 160	ns. ns. ns.	4.5V<VCC<5.5V 2.5V<VCC<4.5V 1.8V<VCC<2.5V
16	THS	Tiempo de Fijación de Mantenimiento	20 40 80	- - -	ns. ns. ns.	4.5V<VCC<5.5V 2.5V<VCC<4.5V 1.8V<VCC<2.5V
17	THH	Tiempo de Mantenimiento	20 40 80	- - -	ns. ns. ns.	4.5V<VCC<5.5V 2.5V<VCC<4.5V 1.8V<VCC<2.5V
18	THZ	Nivel Bajo de Mantenimiento para salida de alta impedancia	30 60 160	- - -	ns. ns. ns.	4.5V<VCC<5.5V 2.5V<VCC<4.5V 1.8V<VCC<2.5V
19	THV	Nivel Alto de Mantenimiento para salida válida	30 60 160	- - -	ns. ns. ns.	4.5V<VCC<5.5V 2.5V<VCC<4.5V 1.8V<VCC<2.5V
20	TWC	Tiempo de Ciclo Interno de Escritura	-	5	ns	-
21	-	Resistencia	1 M	-	Ciclos	-

MAX 3232

El elemento MAX3232 consiste en dos adaptadores de dos líneas, dos receptores de dos líneas y un circuito de carga dual de +/- 15kV. El dispositivo cumple los requerimientos de la especificación TIA/EIA-232-F y provee la interfaz eléctrica entre el controlador de una comunicación asincrónica y el conector de puerto serial

La carga y cuatro capacitores externos permiten la operación de una fuente simple de 3 a 5.5 v.

La operación de transferencia de datos se realiza en tasas de hasta 250 kbits/s



TABLAS DE FUNCION

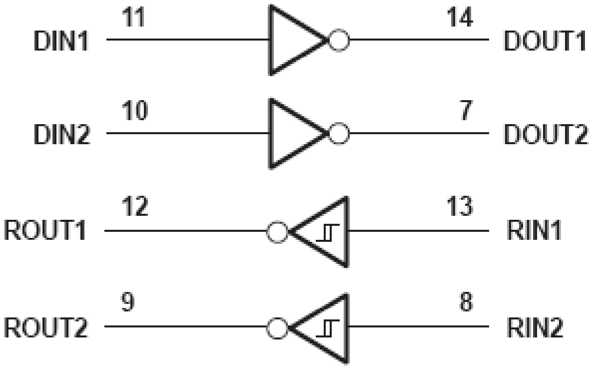
CONTROLADOR

ENTRADA	SALIDA
L	H
H	L

RECEPTOR

ENTRADA	SALIDA
L	H
H	L
	H

Diagrama Lógico (Lógica positiva)

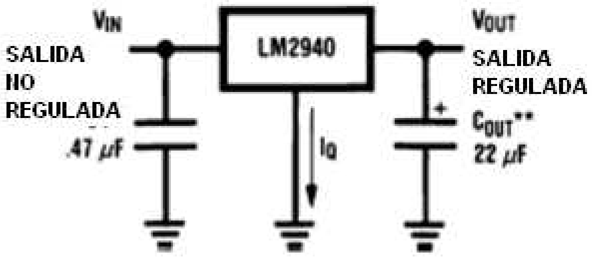


LM2940/LM2940C

El regulador de voltaje positivo LM2940/LM2940C permite una alimentación eléctrica con una corriente máxima de 1A, trabajan regularmente con un mínimo diferencial de voltaje entre la entrada y la salida. En el caso especifico, entre 0.5 y 1V. Además, incluye un circuito que reduce la corriente de tierra cuando el voltaje diferencial entre el voltaje de entrada y el voltaje de salida es excede aproximadamente 3 voltios.

Previsto también para aplicaciones automotrices, el IC y los dispositivos que alimenta están protegidos contra en inversión de la batería, la duplicación de voltaje de la misma

hasta 26V (como ocurre cuando se intente arrancar el vehículo con fuerza bruta), los transitorios de línea y hasta la conexión temporánea invertida (mirror image) del regulador. La desconexión automática se realiza en caso de sobrecarga térmica o corto circuito.



Características Principales:

Voltaje de salida: 5V ±0.15V máx.

Amperaje de salida: 1A

Regulación de línea: 20mV

Regulación de Carga: 35mV

Corriente en reposo: 10mA máx.

Ruido en la salida: 150μV rms

Rechazo de ripple: 72dB min.

Estabilidad (1000h): 20mV

Diferencial de Voltaje: 0.5V min.

Voltaje de Entrada máx.: 26V

Rango de temperatura: 0ºC ~ 125ºC

TC1262

El TC1262 es un regulador mejorado, de alta precisión, con muy bajo voltaje diferencial entre el voltaje de entrada y voltaje de salida; diseñado específicamente para sistemas operados con baterías. Su total alimentación de corriente es de 80 uA en carga completa (de 20 a 60 veces más bajo que en reguladores bipolares).

Entre sus características principales tenemos las siguientes: incluye una operación de muy bajo nivel de ruido, un voltaje diferencial entre la entrada y salida bajo (típicamente 350 mV en carga completa) y una rápida respuesta a cambios en la carga. Además, incluye protección ante niveles elevados de corriente y temperatura. El regulador TC262 es estable con un capacitor externo de 1uF y tiene una corriente de salida de 500 mA.

Características Eléctricas

Voltaje de entrada.	6.5V
Voltaje de Salida	(VSS – 0.3V) to (VIN + 0.3V)
Disipación de Potencia	Internally Limited
Voltaje máximo de cualquier pin	VIN +0.3V to -0.3V
Rango de Temperatura de Operación	-40°C < TJ < 125°C

B. CONCLUSIONES Y RECOMENDACIONES

Una vez desarrollado nuestro proyecto, hemos alcanzado los objetivos propuestos y a la vez podemos dar recomendaciones que permitan facilitar la implementación de futuras aplicaciones.

Podemos entonces concluir y recomendar lo siguiente:

Conclusiones

Si bien es cierto, la aplicación de nuestro proyecto es la de encender y apagar luces, pero es importante recalcar que la tarjeta fue diseñada para que la misma sea escalable. Se la ha dado escalabilidad al colocar varios puertos de salida, dejando asentada así las bases para proyectos futuros y de mayor complejidad con la misma.

La utilización de los protocolos TCP/IP dieron los resultados esperados y fueron la base en la elaboración del código fuente, especialmente en la etapa de conexión a Internet del módulo embebido Ethernet que posee el microcontrolador que utilizamos; además, se pudo observar el trabajo realizado a través de cada una de las cinco capas del modelo TCP/IP como lo demuestra el código.

Al terminar nuestro proyecto, podemos concluir que el PIC18F97J60 es el indicado para aplicaciones de domótica, ya que su costo no es muy elevado como lo podemos observar en el último capítulo y por medio de una simple lógica de programación nos da la facilidad de desarrollar diferente tipos de automatizaciones en el hogar o empresa que permitan mayor control de la misma.

El programa MPLAB IDE junto a su compilador C18 y el programador MPLAB ICD2 fueron en conjunto la herramienta funcional principal de nuestro proyecto de tesis, complementándose entre sí en las diferentes etapas del trabajo y ayudando a hacer las modificaciones correspondientes a través de sus cuadros de diálogo, con el fin de proporcionarnos la facilidad y seguridad en el desarrollo de nuestra aplicación.

A pesar de que usamos el programa Dreamweaver versión 8.0, es imprescindible indicar que puede usarse incluso un editor de textos tan simple como el notepad de Windows, ya que una página web está construida exclusivamente de texto plano que luego es transmitido vía http utilizando la interface Ethernet del microcontrolador.

Normalmente la interface de fuerza, que es la que maneja los voltajes y corrientes de niveles mucho mayores a los que maneja el microcontrolador suele ser bastante tediosa de resolver debido a que hay que hacer un acoplamiento de niveles. Esto lo hemos manejado con el circuito integrado ULN2803AG provisto por TOSHIBA, con lo que no solo superamos lo engorroso del asunto sino que logramos que la etapa de fuerza sea muy pequeña en comparación con los esquemas tradicionales de manejo de relés hechos a base de transistores y adicional a eso permite disminuir el consumo de potencia.

Cuando hicimos la comparación de velocidades entre la tarjeta que diseñamos con el microcontrolador PIC18F97J60 y la tarjeta de red 10/100 Intel® PRO/100 VE Network Connection, pudimos verificar que esa característica era muy similar en ambas, lo cual nos deja muy satisfechos, ya que nuestro objetivo cuando empezamos este proyecto fue lograr un producto que sirva para el control remoto de actividades en el hogar con características que puedan competir en el mercado.

El costo total de la elaboración del proyecto fue de $576.35, valor aceptable en vista de todas las aplicaciones que se pueden desarrollar con las respectivas modificaciones en el código de programación.

Recomendaciones

Se recomienda el uso de MPLAB IDE, como una potente herramienta de programación de todo tipo de microcontroladores proveídos por Microchip, ya que brinda toda la ayuda y facilidades necesarias al momento de compilar y programar los diferentes códigos fuentes.

Es necesario el conocimiento básico de lenguaje C, ya que es una manera mucho más reducida y amigable al momento de la programación de los microcontroladores.

Existen diferentes modelos de programación proveídos por Microchip en su página web que debemos tomar en cuenta, ya que nos permiten entender las estructuras de los diferentes tipos de programas, haciendo así mucho más sencillo el desarrollo de los mismos.

Recomendamos la utilización de conectores modulares RJ 45 para la conexión a la red de nuestra tarjeta, ya que ellos nos dan la seguridad de no tener interferencias electromagnéticas de ningún tipo.

Por efectos de alimentación, es necesario utilizar fuentes DC diferentes en ambas etapas del proyecto teniendo en cuenta su respectiva polaridad y amperaje.

Se recomienda, modularizar físicamente la aplicación, es decir, que ésta tenga la capacidad de ser conectada y desconectada entre etapas para su mejor control y reparación en caso de daños o mal funcionamiento

BIBLIOGRAFÍA

1. Sistemas de Control para Viviendas y Edificios: Domótica, José Mª Quinterito González, Javier Lamas Graziani y Juan D. Sandoval González.
2. Instalaciones automatizadas en viviendas y edificios, Leopoldo Mc Grawn Hill.
3. Dreamweaver 8, Primeros pasos con Dreamweaver.
4. Manual de Usuario de MPLAB IDE.
5. Manual de usuario del Compilador C, MPLAB C18 (v3.00).
6. Manual de usuario de Programador ICD 2.
7. Hoja de datos del PIC 18F97J60.

REFERENCIAS DE INTERNET

- ✓ http://es.wikipedia.org
- ✓ http://www.casadomo.com
- ✓ http://www.cebus.com
- ✓ http://www.intellon.com
- ✓ http://www.domosys.com
- ✓ http://www.echelon.com
- ✓ http://www.lonmark.com
- ✓ http://www.microsoft.com
- ✓ http://www.bacnet.org
- ✓ http://www.havi.org
- ✓ http://www.jini.org
- ✓ http://www.zigbee.org
- ✓ http://www.konnex.org

- ✓ http://www.microchip.com