ESCUELA SUPERIOR POLITECNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación.

CAPTURADOR DE PAQUETES IMPLEMENTADO EN HARDWARE

TESINA DE SEMINARIO

Previo a la obtención del título de:

INGENIERO EN ELECTRONICA Y TELECOMUNICACIONES

Presentado Por:

Simón Vinicio Castro González

Melany Carla Salvador Antón

GUAYAQUIL-ECUADOR

2011

AGRADECIMIENTO

Agradecemos a Dios por ayudarnos en el desarrollo de este trabajo a pesar de todas las adversidades, a nuestros padres y a todas las personas que de manera directa o indirecta contribuyeron.

DEDICATORIA

A Dios en primer lugar por ser el eje de mi vida.

A mis Padres Enrique y Betty, a mi familia en general que estuvo pendiente de este proceso; por apoyarme y guiarme hasta el final.

Melany Salvador Antón.

DEDICATORIA

Primeramente a Dios quien me ha guiado y supo darme fuerzas en los momentos más difíciles.

A mis Padres Simón y Rosana, hermanos, enamorada y Don Rigoberto Jarrín quienes me infundieron la ética y el rigor que guían mi transitar por la vida.

Simón Vinicio Castro González.

TRIBUNAL DE SUSTENTACION

_____

Ing. Ignacio Marín García. MSIS

PROFESOR DEL SEMINARIO DE GRADUACIÓN

_____

Ing. Daniel Ochoa Donoso.

PROFESOR DELEGADO DEL DECANO DE LA FIEC

DECLARACION EXPRESA

"La responsabilidad del contenido de este trabajo, nos corresponde exclusivamente; y el patrimonio intelectual del mismo a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL".

(Reglamento de exámenes y títulos profesionales de la ESPOL)

_____

Simón Castro

_____

Melany Salvador

RESUMEN

El trabajo presentado en este proyecto de seminario de graduación, fue el desarrollar un “Capturador de Paquetes en Hardware”, para lo cual utilizamos la Tecnología del Microcontrolador PIC18F4520, un modulo de red ENC28J60 y un módulo de Micro Memoria SD.

El primer capítulo, comprende una explicación teórica de la Seguridad en Redes, empezando por antecedentes y organizaciones de redes, definición del término seguridad en redes, principios fundamentales de una red segura y finalmente las funciones de capturar y analizar paquetes.

En el segundo capítulo constan los principios teóricos del Microcontrolador PIC18f4520, el modulo de memoria micro SD, el modulo de red ENC28J60; así como, la descripción y utilización de los mismos. También se detalla acerca del compilador MikroBasic y el software de elaboración de simulaciones electrónicas Proteus.

El tercer capítulo contiene los detalles del diseño y desarrollo de hardware y software para los módulos, la descripción general, diagramas de bloques y flujo.

En el cuarto capítulo se analiza el funcionamiento del prototipo, con los resultados obtenidos de las pruebas realizadas durante la captura; estas capturas fueron realizadas para paquetes UDP, TCP dirigidos y no dirigidos.

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

AGRADECIMIENTO II
DEDICATORIA III
DEDICATORIA IV
TRIBUNAL DE SUSTENTACION V
DECLARACION EXPRESA VI
RESUMEN VII
ÍNDICE GENERAL IX
ÍNDICE DE IMÁGENES XI
ÍNDICE DE TABLAS XII
TABLA DE ABREVIATURAS XIII
GLOSARIO XVI
INTRODUCCIÓN XVIII
CAPITULO 1. ANTECEDENTES Y ORGANIZACIÓN DE REDES 1
1.1 DEFINICION DEL TERMINO SEGURIDAD 1
1.2 PRINCIPIOS FUNDAMENTALES DE UNA RED SEGURA 2
1.3 ANALIZADOR DE PAQUETES 4
1.4 CAPTURADORES DE PAQUETES 4
CAPÍTULO 2. HERRAMIENTAS UTILIZADAS 7
2.1 MICROCONTROLADOR 7
2.2 MEMORIA Y MODULO SD. 8

2.3 MODULO DE RED 10
2.4 SOFTWARE 11
CAPITULO 3. DISEÑO Y DESARROLLO DE HARDWARE Y SOFTWARE 13
3.1 DESCRIPCIÓN GENERAL. 13
3.2 DISEÑO DE HARDWARE Y SOTFWARE 17
3.3 DESCRIPCION DE LOS BLOQUES FUNCIONALES 18
3.4 DESARROLLO DE SOFTWARE 23
3.5 DESARROLLO DE HARDWARE 31
CAPITULO 4. RESULTADOS EXPERIMENTALES 33
4.1 ANÁLISIS DE FUNCIONAMIENTO 33
4.2 RESULTADOS DE EFICIENCIA 35
CONCLUSIONES 44
RECOMENDACIONES 45
ANEXO A. PROTOCOLOS UTILIZADOS 46
ANEXO B. DETALLES DE PAQUETES UDP CAPTURADOS 55
ANEXO C. DETALLES DE PAQUETES TCP CAPTURADOS 59
ANEXO D. DETALLES DE PAQUETES NO DIRIGIDOS CAPTURADOS 63
ANEXO E. DETALLES DE PAQUETES ARP CAPTURADOS 67
ANEXO F. ARCHIVOS FUENTE DE LA PROGRAMACION DEL MICROCONTROLADOR 71
ANEXO G. ARCHIVOS FUENTE DE LA PROGRAMACION DEL MODULO 80
ANEXO H. EVALUACION ECONOMICA 83
ANEXO I. DISEÑO DE TARJETAS 88
BIBLIOGRAFÍA Y REFERENCIAS DE INTERNET 93

ÍNDICE DE IMÁGENES

FIGURA 2. 1 TARJETA DE MEMORIA MICRO SD Y ADAPTADOR 8
FIGURA 2. 2 MODULO SD 9
FIGURA 2. 3 DESCRIPCION DEL MODULO DE RED 10

FIGURA 3. 1 DIAGRAMA DE UNA RED 15
FIGURA 3. 2 INCLUSION DEL DISPOSITIVO EN LA RED 16
FIGURA 3. 3 DIAGRAMA DEL PROTOTIPO 17
FIGURA 3. 4 DIAGRAMA DE RED 18
FIGURA 3. 5 SINCRONIZACION DE SPI 19
FIGURA 3. 6 DIAGRAMA DE MEMORIA Y MODULO SD 21
FIGURA 3. 7 DIAGRAMA DE CAPTURA 23
FIGURA 3. 8 DIAGRAMA DE FLUJO DEL MODULO DE RED 25
FIGURA 3. 9 DIAGRAMA DE FLUJO DEL MODULO DE MEMORIA 27
FIGURA 3. 10 DIAGRAMA DE FLUJO DEL PROGRAMA PRINCIPAL 30
FIGURA 3. 11 DISEÑO DE TARJETA 31

FIGURA 4. 1 RESULTADOS DEL PROTOTIPO EN UDP 37
FIGURA 4. 2 EFICIENCIA DE LA CAPTURA CON PAQUETES UDP 37
FIGURA 4. 3 TIEMPO DE ESPERA PARA PAQUETES UDP 38
FIGURA 4. 4 RESULTADOS DEL PROTOTIPO EN TCP 40
FIGURA 4. 5 EFICIENCIA DE LA CAPTURA DE PAQUETES TCP 40
FIGURA 4. 6 TIEMPO DE ESPERA PARA PAQUETES TCP 41
FIGURA 4. 7 RESULTADOS DEL DISPOSITIVO EN PAQUETES NO DIRIGIDOS 42
FIGURA 4. 8 EFICIENCIA DE LA CAPTURA DE PAQUETES NO DIRIGIDOS 43

ÍNDICE DE TABLAS

TABLA 3.1. INSTRUCCIONES PARA EL MODULO DE RED 21

TABLA DE ABREVIATURAS

ARP: Protocolo de resolución de direcciones, (Address Resolution Protocol). Es un protocolo responsable de encontrar la dirección MAC.

CCP: captura/comparación/PWM. Es un modulo que tienen los microcontroladores el cual nos permite trabajar en diferente modo, sea captura, comparación o PWM.

CMOS: Semiconductor oxido-metalico complementario, (Complementary metal-oxide-semiconductor). Es una familia lógica cuyos integrados son de bajo consumo de potencia.

EEPROM: Memoria de lectura, programmable y borrable, (Electrically-Erasable Programmable Read-Only Memory).

FPGA: Campos de matriz de puertas programables, (Fiel Programmable Gate Array). Es un dispositivo semiconductor programable.

IEC: Comisión electrotécnica internacional, (International Electrotechnical Commission). Es una organización de normalización en los campos eléctrico, electrónico y tecnologías relacionadas.

IPV4: Protocolo de Internet versión 4 (Internet Protocol Version 4). Es un protocolo orientado a datos que se utiliza para comunicaciones entre redes.

ISO: Organización internacional de estándares, (International Organization for Standardization). Es una organización encargada de la estandarización de normas de productos y seguridad para las empresas u organizaciones a nivel internacional.

LAN: Red de área local, (local area network). Es una red de conexión en áreas pequeñas.

MAC: Control de Acceso al medio, (media Access control). Conjunto de protocolos de dispositivos que comparten un medio de comunicación común.

MBits/s : MegaBits por segundo. Velocidad de transmisión de datos.

MIPS: Millones de instrucciones por segundo. Es una forma de medir la potencia de los procesadores.

PIC: Controlador de interrupciones programable, (Programmable Interrupt Controller). Es una familia de microcontroladores.

PWM: Modulación de ancho de pulso,(pulse-width modulation). Es una técnica que modifica el ciclo de una señal periódica.

RAM: Memoria de acceso aleatorio,(random-access memory). Es una memoria de acceso del procesador donde almacena los resultados de sus procedimientos.

RISC: Conjunto de instrucciones de computadora reducidas, (reduced instruction set computer). Es una arquitectura computacional con instrucciones de tamaños fijos y presentados en un reducido número de formato.

SPI: Interfaz periférico serial, (Serial Peripheral Interface). Es un estándar de comunicaciones, usado principalmente para la transferencia de información entre circuitos integrados en equipos electrónicos.

TCP/IP: Protocolo de control de transmisión/ Protocolo de internet, (Transmission Control Protocol/Internet Protocol). Es un modelo que descripción protocolos de red.

TCP: Protocolo de control de transmisión, (Transmission Control Protocol). Protocolo de capa de transporte basado en la confirmación de entrega de paquetes.

UDP: Protocolo de datagramas de usuario, (User Datagram Protocol). Es un protocolo de la capa de transporte basado en el intercambio de paquetes de datos.

USART: Transmisor o receptor, síncrono o asíncrono, universal, (universal synchronous/asynchronous receiver/transmitter). Es un puerto de envío y recepción de datos en modo síncrono o asíncrono que poseen ciertos microcontroladores.

USB: Bus serial universal, (universal serial bus). Es un puerto utilizado para conectar periféricos a un ordenador. También es un protocolo de comunicación.

VDC: Voltaje de corriente continúa.

GLOSARIO

.txt : Extensión de archivos de texto plano.

10BASET: Es un estándar que define la conexión en un cable de par trenzado

Broadcast: Es una área lógica en una red de ordenadores en la que cualquier ordenador conectado a la red puede transmitir directamente a cualquier otro en el dominio.

Conmutador: Dispositivo digital utilizado para interconectar redes.

Estándar IEEE 802.3: Primera versión del estándar para Ethernet.

Firewall: Es dispositivo o software diseñado para filtrar el tráfico entrante y saliente de la red.

Flash: Es una tecnología de almacenamiento de datos a gran velocidad.

Full dúplex: Protocolo de envío de información bidireccional simultaneo.

Gateway: Puerta de enlace, Dispositivo que conmuta redes y protocolos.

Half dúplex: Semiduplex, significa que el método o protocolo de envió de información es bidireccional pero no simultaneo.

Host: Dispositivo utilizado por el usuario para acceder a la red.

Hub: Concentrador, un dispositivo para compartir una red de datos o de puertos de red.

Labview: Software de programación grafica para simulación.

Micro SD: Es un tipo de tarjeta flash de tamaño pequeño.

Ping: Es una utilidad diagnostica en redes de computadoras que comprueba el estado de conexión del host local con uno o varios equipos remotos de una red TCP/IP por medio del envió de paquetes ICMP de solicitud y respuesta.

Pull – up: Resistencia que se utiliza en electrónica de circuitos lógicos para asegurar que las aportaciones a los sistemas de lógica se ubicaren en los niveles que se espera si los dispositivos externos están desconectados o alta impedancia.

Wireshark: Software de análisis de trafico de red.

INTRODUCCIÓN

En la actualidad, la seguridad en redes es una parte integral en las redes informáticas, incluye protocolos, tecnologías, herramientas y técnicas que aseguran los datos y reducen las amenazas.

Nuestro proyecto consiste en un dispositivo capturador de paquetes de datos, el cual debe ser conectado al equipo de distribución de red. Una vez realizado ésto, el dispositivo se encuentra en estado de espera, en donde un administrador le dará la orden de que comience con la captura de paquetes.

El dispositivo está diseñado en tres etapas: de red, de captura y de archivo. En la etapa de red, es donde le brindamos al prototipo los campos necesarios para enlazarse a una red, siendo éstos la dirección IP, su puerta de enlace, máscara y DNS, los cuales serán otorgados o configurados al módulo de red. Una vez enlazados se procede a la segunda etapa de nuestro prototipo estando listos para realizar la captura de paquetes cuyo encargado es el microcontrolador. Finalmente en la última etapa participa el módulo de micro memoria SD quien será el encargado de guardar los paquetes capturados y procesados.

CAPITULO 1

ANTECEDENTES Y ORGANIZACIÓN DE REDES

La seguridad de las redes es ahora una parte integral de las redes informáticas. Incluye protocolos, tecnologías, dispositivos, herramientas y técnicas que aseguran los datos y reducen las amenazas.

1.1 DEFINICION DEL TERMINO SEGURIDAD

El término seguridad se lo puede definir como un estado de ánimo, una sensación, una cualidad intangible. También se lo puede definir como un objetivo y un fin que el hombre desea como una necesidad primaria. La Seguridad es un concepto que no se llega a alcanzar en un 100% debido a que si hay un camino de acceso se pueden encontrar formas de volverlo vulnerable [1].

1.2 PRINCIPIOS FUNDAMENTALES DE UNA RED SEGURA

Los **impulsores de la seguridad** son los profesionales encargados de la seguridad en redes. Ellos mantienen la seguridad de los datos de una organización y garantizan la integridad y confidencialidad de la información. Debe ser responsable de montar firewalls y sistemas de prevención de intrusos, así como también de asegurar el cifrado de los datos de la compañía.

Deben implementar esquemas de autenticación, mantener historiales detallados de actividad sospechosa en la red para usarla para reprender y procesar infractores, mantener una familiaridad con las organizaciones de seguridad en redes. Estas organizaciones generalmente tienen la información más actualizada sobre amenazas y vulnerabilidades.

Es vital que los profesionales de la seguridad en redes entiendan los motivos de la misma y se familiaricen con las organizaciones dedicadas a ésta. También es importante entender los varios **dominios de la seguridad en redes**. Los dominios proveen un marco organizado para facilitar el aprendizaje sobre la seguridad en redes.

Donde un dominio fundamental está en las Políticas de Seguridad, estando en ellas la declaración formal de las reglas a las cuales deberán atender las personas que tienen acceso a los bienes tecnológicos y de información de una empresa. La aplicación de nuestro proyecto puede servir para el posterior análisis de los datos capturados y conocer la utilización de la herramienta Internet.

La **política de seguridad** en redes se utiliza para asistir en el diseño de la red, transmitir principios de seguridad y facilitar el despliegue de la red. En nuestro caso podemos conocer el flujo de los paquetes en la red, al guardarlos en una memoria micro SD para su posterior análisis.

Para realizar un **análisis de la Seguridad Informática** se debe conocer las características de la Información. Conociendo esto podemos preservar la información crítica, valiosa y sensitiva de la empresa. Pudiendo así hacer análisis periódicos para conocer la privacidad, autenticidad de la información, manejando mecanismos de prevención y detección de intrusos a la red.

1.3 ANALIZADOR DE PAQUETES

Tener políticas de seguridad afirma que una red está bajo ciertas condiciones o reglamentos que deben cumplirse por parte de todos los usuarios que hagan uso de ella; por ello fue de suma importancia el contar con políticas de seguridad debido a la necesidad de realizar nuestro análisis de red bajo ciertos conceptos; es decir si no hubieran políticas de seguridad no sería necesario realizar un análisis de red porque no existiría un reglamento infringido por parte de los usuarios.

Partiendo de lo mencionando anteriormente podemos decir que un analizador de paquetes es una herramienta muy importante ya que debido a su análisis se pueden hacer restricciones de tráfico que fluye por la red. También un analizador decodifica datos en bruto, es decir que muestra los valores de varios campos en el paquete y se analiza su contenido de acuerdo a los correspondientes RFC (solicitud de comentario) u otras especificaciones.

1.4 CAPTURADORES DE PAQUETES

La captura de paquetes consiste en el acto de atrapar el tráfico que fluye en una red de ordenadores. La captura profunda de paquetes en cambio realiza la captura a una velocidad de red completa, es decir que

los paquetes de red son capturados en su cabecera y la carga útil (ó dato) en el cruce de tráfico por la red. Una vez capturada y almacenada, ya sea en una memoria a corto o a largo plazo, puede ser utilizada con las herramientas de software para una inspección profunda de paquetes (ó análisis de paquetes), pudiéndose realizar un análisis forense para descubrir la causa raíz de problemas de red, identificar amenazas de seguridad y garantizar las comunicaciones de datos y el uso de la red cumple con las políticas descritas.

También es importante mencionar que se puede realizar la captura parcial de paquetes, es decir la grabación de cabeceras sin grabar el contenido total de datagramas. Esto puede reducir los requisitos de almacenamiento y evitar problemas legales, pero aun no tenemos datos suficientes para revelar la información esencial necesaria para el diagnóstico del problema.

En la actualidad podemos encontrar dos tipos de capturadores de paquetes, estos son por software o por hardware, entre los que podemos citar: wireshark, packet viewer, packet analyzer profesional, keyloggings, sniffers, keydevil, keyghost, cuyas finalidades son conocer todo el tráfico de la red. Si bien es cierto, los keyloggings

capturan todo lo digitado por el usuario y no solo los datos de la red, se lo menciona por su función de capturar datos.

La mayoría de los capturadores citados requieren de actualizaciones de sus sistemas operativos, representado esto un costo para todo tipo de empresa; punto crítico en la toma de decisiones para la adquisición de nuevas tecnologías dentro de una empresa.

CAPÍTULO 2

HERRAMIENTAS UTILIZADAS

En este capítulo detallamos cada una de las herramientas utilizadas para la realización del prototipo, entre ellas tenemos el microcontrolador, una memoria y módulo SD, el módulo de red y el software de programación para el pic y simulación.

2.1 MICROCONTROLADOR

Nuestro proyecto utilizó el microcontrolador PIC18F4520, por tener un alto rendimiento de 10 millones de instrucciones por segundo (MIPS), es decir, posee gran velocidad de procesamiento de instrucciones, su arquitectura RISC que permite instrucciones de palabras de 16 bits, un bajo consumo de energía, la posibilidad de comunicarse con periféricos y protocolos USB, SPI y TCP/IP, y contener una memoria EEPROM, FLASH y RAM.

2.2 MEMORIA Y MODULO SD.

Para la realización del prototipo se utilizó una tarjeta de memoria micro SD con interface de comunicación de 8 pines. Esta es compatible con memorias SD con la utilización de un adaptador. La tecnología SD permite portabilidad, almacenamiento de datos, lo cual es apropiado para nuestro prototipo. La capacidad de almacenamiento escogida fue de 1GB.

FIGURA 2. 1 TARJETA DE MEMORIA MICRO SD Y ADAPTADOR

El módulo micro SD nos permite leer y/o escribir información dentro de una tarjeta de memoria micro SD, éste mismo tiene incorporado un

socket para micro SD, bus de datos para comunicación SPI y regulador interno de voltaje de 3.3 VDC (voltios corriente continua); puede ser alimentado con fuentes de VDD (5 – 18) VDC. Sus respectivas señales de comunicación SPI son: SEL, SDI, SCK, SDO; para habilitar y deshabilitar resistores pull-up para cada pin SPI de forma inmediata esta las señales EN y DS y finalmente tenemos la selección de voltaje de entrada.

Este módulo tiene que ser configurado por un microtrolador para que funcione correctamente, es decir todas las órdenes de escritura que se realizarán para este módulo vendrá desde el microcontrolador, por ejemplo si quiero grabar un dato, el microcontrolador preparará al módulo SD mediante señales de control para la respectiva grabación.

FIGURA 2. 2 MODULO SD

2.3 MODULO DE RED

También se utilizó un módulo de red, el ENC28J60 diseñado para servir como una interfaz de red Ethernet para cualquier controlador equipado con SPI; así mismo cumple con todas las especificaciones IEEE 802.3. El módulo incorpora una serie de filtros de paquetes para limitar los paquetes entrantes. También proporciona un módulo interno DMA para una rápida transferencia de datos. La comunicación con el controlador de host se implementa a través de un pin de interrupción (pin llamado INT) y el SPI, con velocidades de reloj de hasta 20 MHz.

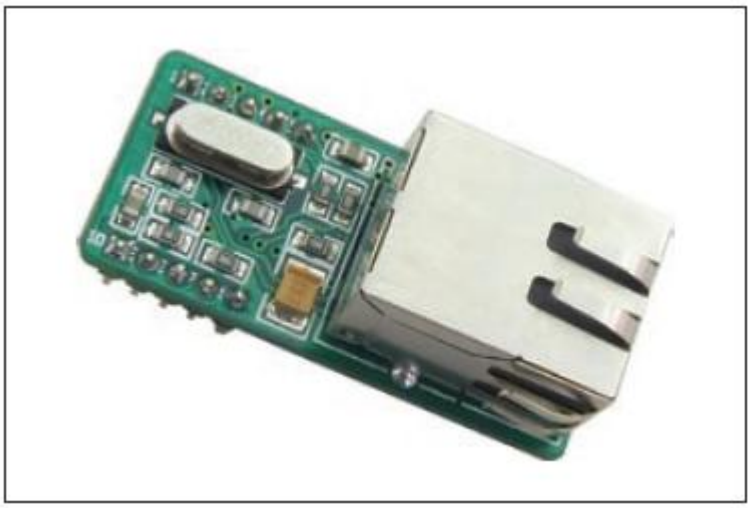


FIGURA 2. 3 DESCRIPCION DEL MODULO DE RED

El Módulo ENC28J60 nos permite desarrollar diferentes aplicaciones, entre la que podemos citar, en nuestro caso, el módulo Ethernet, es uno

de los componentes principales del Microcontrolador y de nuestro proyecto, el cual incluye implementaciones completas de módulos de Control de Acceso al Medio (MAC) y de la capa física (PHY). Lo único que se requeriría para la conexión a la red seria dos transformadores y un conjunto de elementos pasivos que se detallarán en el capítulo siguiente. Este módulo también reconoce todas las especificaciones para la conectividad, 10 Base T en una red de par trenzado. Para el proceso de datos, cálculos de verificación de direcciones IP y verificación de actividad del módulo Ethernet; incorpora dos señales de LEDS que actúan como salidas para demostrar su actividad y enlace respectivamente.

Al igual que el módulo micro SD, este módulo tiene que ser configurado por el microtrolador, partiendo desde la configuración de pines, de ciertos parámetros como dirección IP, mascara de red, DNS y la puerta de enlace.

2.4 SOFTWARE

Para la realización de la programación utilizamos **MikroBasic** como lenguaje de programación, el cual posee múltiples funciones de desarrollo para microcontroladores PIC y está diseñando para

proporcionar al cliente soluciones simples, teniendo una serie de librerías como “Multi Media Card Library, SPI Ethernet Library”, cada una de ellas con funciones dirigidas a aplicaciones, que ayudan en el desarrollo de sistemas, sin comprometer el rendimiento o el control.

Y para la elaboración de simulaciones electrónicas utilizamos **Proteus** como un paquete de software para el diseño de circuitos electrónicos que incluye captura de los esquemas, simulación analógica y digital combinada, además posee una aplicación llamada **ARES** que se utiliza para el diseño de circuitos impresos. Siendo esta herramienta quien nos permite simular nuestro proyecto antes de ensamblarlo y así evitar gastos innecesarios de hardware.

CAPITULO 3

DISEÑO Y DESARROLLO DE HARDWARE Y SOFTWARE

Para el diseño y desarrollo de hardware y software nos basamos en el concepto de la modularidad, la cual consistió en la división de diferentes etapas para la elaboración del prototipo, con la selección de un diagrama de red; siendo lo mencionado pilar fundamental para la elaboración del mismo.

3.1 DESCRIPCIÓN GENERAL.

Desarrollamos nuestro prototipo dividiendo en tres partes: el software y hardware de programación del microcontrolador, el módulo de red, y la tarjeta de memoria micro SD con su módulo micro SD. Con estos módulos conseguimos simplificar el trabajo, los cuales están explicados a lo largo del presente capítulo.

Una vez ya distribuidas las diferentes etapas o fases de cómo se fue desarrollando el prototipo; procedimos a la elaboración del software para el microcontrolador y para ello era indispensable saber con qué herramientas trabajaríamos. La programación que lleva el microcontrolador está distribuida en la configuración para el módulo de red, para el módulo micro SD y así mismo es este el encargado de la captura. Una vez desarrollado el código fuente, fue programado en al microcontrolador mediante la herramienta PICKIT Microcontroller Programmer (este es un módulo con conexión USB y adicional tiene los respectivos pines de programación para el microcontrolador).

El módulo Ethernet nos permitió comunicarnos con la red LAN, este realizó la comunicación Half y Full Duplex en base 10 (10BASE-T), siendo compatible con el estándar 802.3 de la IEEE. La tarjeta de memoria micro SD nos permitió el almacenamiento de los paquetes capturados y el modulo micro SD nos permitió la escritura de archivos con extensión .txt de forma autónoma.

Para demostrar el funcionamiento de comunicación del prototipo por la red LAN. Primeramente le otorgamos o configuramos al prototipo la dirección IP, mascara, puerta de enlace, DNS, y la Mac del mismo;

posteriormente lo conectamos al equipo de distribución de la red mediante un cable de red y finalmente desde un ordenador que se encontraba enlazado a la red se realizó una prueba de Ping la misma que tuvo un resultado exitoso, interpretándose que nuestro prototipo se encuentra enlazado con la red.

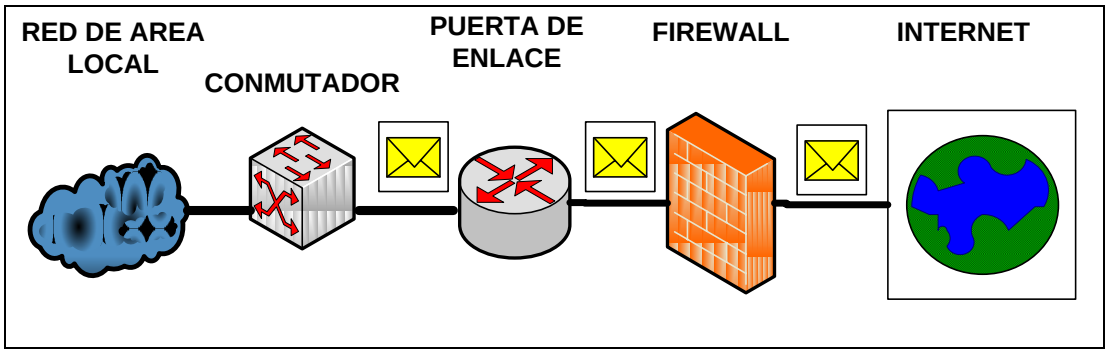


FIGURA 3. 1 DIAGRAMA DE UNA RED

En la figura 3.1 se muestra una configuración típica de una red LAN conectada a Internet. En toda red tenemos esencialmente dispositivos terminales que son los usuarios, seguido de ello tenemos un dispositivo conocido como conmutador que es el encargado de permitir que los usuarios se encuentren conectados a una red. En el caso de una empresa con pocos usuarios sería suficiente con un conmutador para que todos ellos se encuentren comunicados entre sí; el problema se fundamenta cuando se tiene un grupo grande de usuarios y la existencia de varios departamentos, donde como diseñadores de redes queremos dar a cada

departamento diferentes atributos y permisos dentro de una red; aquí nos encontramos con una gran limitante al querer usar solo un conmutador dentro de la red; siendo éste el escenario en donde ya interviene un enrutador que es un dispositivo capaz de enrutar paquetes a las diferentes direcciones.

Hemos hecho una breve descripción del diseño que proponemos en el diagrama de red indicado en la figura 3.1; en este diseño nos basamos para realizar el desarrollo e implementación de nuestro prototipo capturador de paquetes. En la siguiente figura (figura 3.2) mostramos en qué fase de la red va ubicado el prototipo.

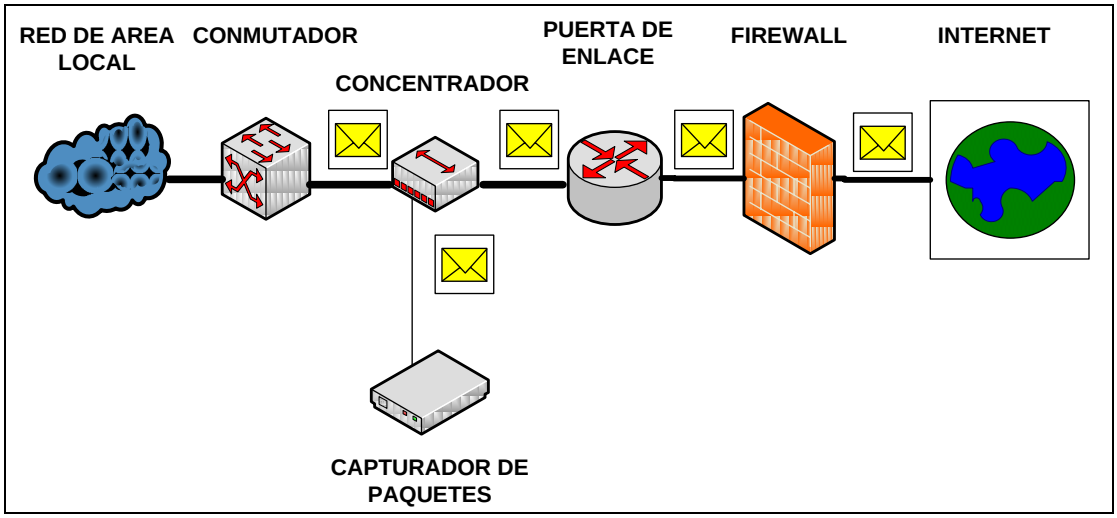


FIGURA 3. 2 INCLUSION DEL DISPOSITIVO EN LA RED

En la figura 3.2 observamos la utilización de un concentrador para conectar el prototipo a la red, esto con la finalidad duplicar o tener una copia del tráfico generado por la red. Esto significa que dicho dispositivo recibe una señal y repite esta señal por sus diferentes puertos; una característica que a nosotros nos interesa, ya que el tráfico que se está generando en la LAN llega a el Conmutador y mediante un enlace troncal pasa al enrutador (en nuestro caso al Gateway) ya sea para salir a Internet o para enrutarse internamente, nos referimos a enlace troncal a que todo el tráfico que se genera en la LAN pasará por ese enlace. Con esto confirmamos que un cien por ciento de la información generada en la LAN que viaje desde el Conmutador al Enrutador llegó a nuestro dispositivo.

3.2 DISEÑO DE HARDWARE Y SOTFWARE

Con el objetivo de ir desarrollando etapa por etapa la elaboración del prototipo nos fundamentamos en el siguiente diagrama de bloques de la figura 3.3.

FIGURA 3. 3 DIAGRAMA DEL PROTOTIPO

Se observa que tenemos tres bloques, los cuales corresponden a la etapa de red, de captura y finalmente la de archivo. A continuación hablaremos detalladamente de cada uno de los bloques.

3.3 DESCRIPCION DE LOS BLOQUES FUNCIONALES

El bloque de red está comprendido por el módulo de red "ENC28J60". Este módulo está diseñado para interactuar directamente mediante una interfaz SPI.

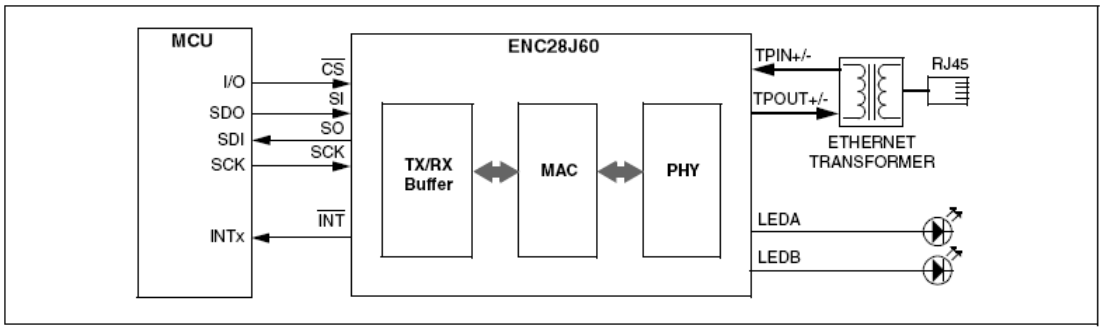


FIGURA 3. 4 DIAGRAMA DE RED

En la figura 3.4 se muestra detalladamente como está constituido el bloque de red. El MCU que observamos es la unidad de control, en nuestro caso el microcontrolador; los comandos y los datos se envían entre dispositivos (microcontrolador - ENC28J60) a través del pin SI,

se envían en el flanco de subida del SCK; mientras que los dados expulsados (ENC28J60 – microcontrolador) se envían en el flanco de bajada de SCK por el pin SO. El pin CS se debe mantener bajo mientras se realiza cualquier operación y regresaría a alto cuando se haya terminado de realizar operación alguna.

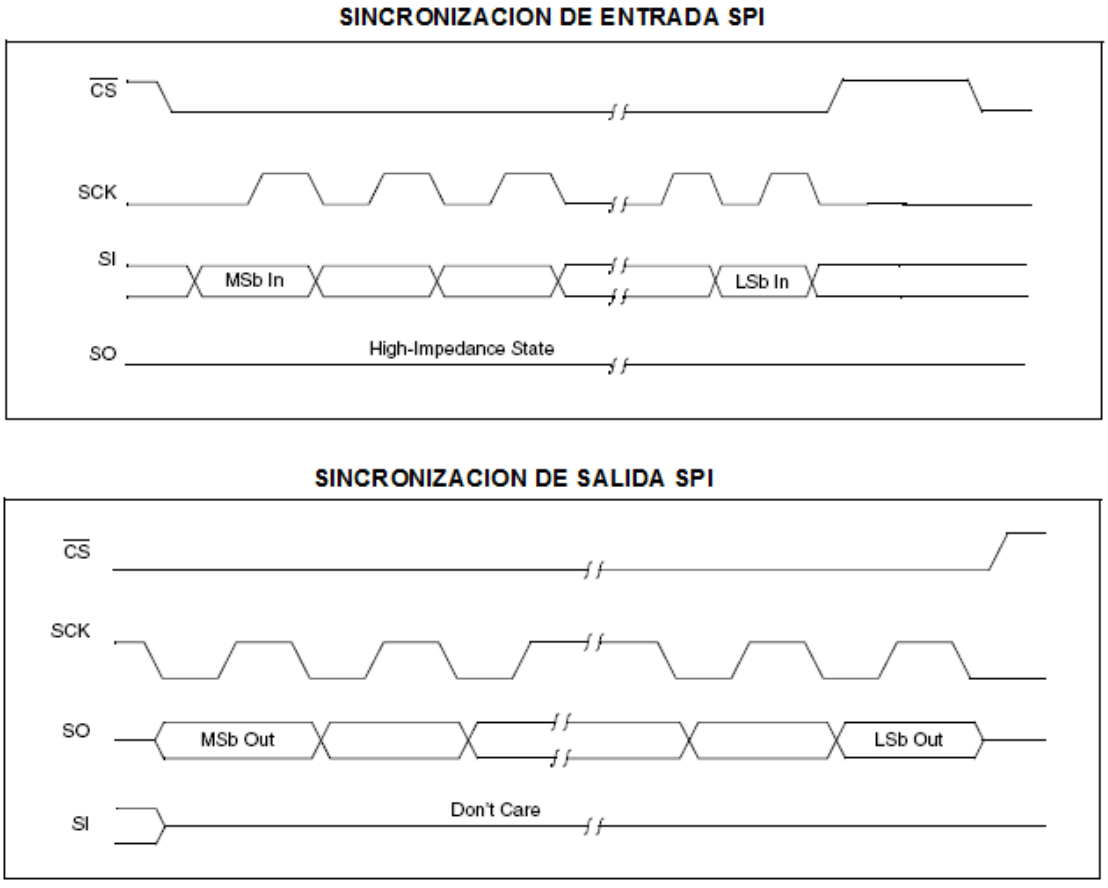


FIGURA 3. 5 SINCRONIZACION DE SPI

El funcionamiento del ENC28j60 depende enteramente de órdenes dadas por un controlador de host externo sobre la interfaz SPI. Estos

comandos tienen la forma de instrucciones, de uno o más bytes, que se utilizan para acceder a la memoria de control y espacios de amortiguación Ethernet. Las instrucciones consisten en un código de operación de 3 bis, seguido por uno de 5 bits que se especifica una dirección de registro o de datos constantes. Un total de siete instrucciones se aplican sobre el ENC28J60. En la tabla 3.1 se muestra los códigos de comandos para todas las operaciones.

Instruction Name and Mnemonic	Byte 0								Byte 1 and Following							
	Opcode			Argument					Data							
Read Control Register (RCR)	0	0	0	a	a	a	a	a	N/A							
Read Buffer Memory (RBM)	0	0	1	1	1	0	1	0	N/A							
Write Control Register (WCR)	0	1	0	a	a	a	a	a	d	d	d	d	d	d	d	d
Write Buffer Memory (WBM)	0	1	1	1	1	0	1	0	d	d	d	d	d	d	d	d
Bit Field Set (BFS)	1	0	0	a	a	a	a	a	d	d	d	d	d	d	d	d
Bit Field Clear (BFC)	1	0	1	a	a	a	a	a	d	d	d	d	d	d	d	d
System Reset Command (Soft Reset) (SRC)	1	1	1	1	1	1	1	1	N/A							

Legend: a = control register address, d = data payload.

TABLA 3.1. INSTRUCCIONES PARA EL MODULO DE RED

El módulo cuenta con un conector RJ45, para podernos conectar a una red mediante un cable de red; también cuenta con un búfer para recepción y transmisión de datos, un registro MAC que es compatible con paquetes multicast, unicast y broadcast, y brinda un filtrado de

paquetes. Finalmente cuenta con un registro de capa física brindándonos comunicación half y full dúplex, y sus leds A y B indican actividad de red.

El bloque de archivo está compuesto del módulo de SD y de la memoria micro SD.

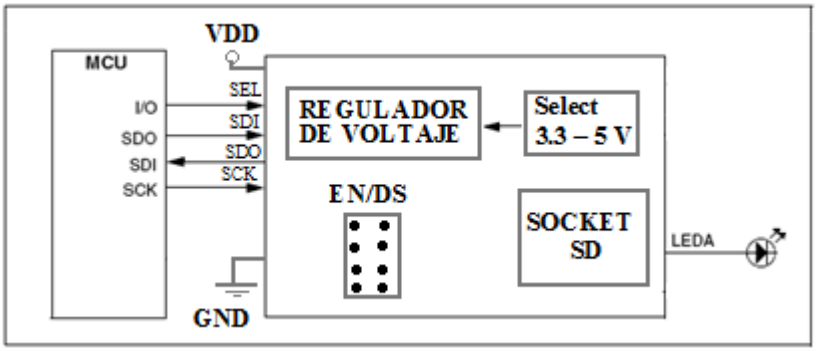


FIGURA 3. 6 DIAGRAMA DE MEMORIA Y MODULO SD

En la figura 3.6 observamos que al igual que el módulo de red interactúa mediante una interfaz SPI, que consta de su señal de reloj (SCK), señal de datos de entrada (SDI), señal de datos de salida (SDO) y una señal de selección (SEL). Consta de una alimentación de voltaje (VDD) que pasara por un regulador para brindarnos dos opciones de alimentación externa de 3.3 y 5 voltios; además tenemos los pines de EN/DS utilizados para habilitar o deshabilitar las resistencias pull-up

para cada pin SPI de forma independiente. También encontramos un socket dentro del diagrama de bloques en donde se ubicara la memoria micro para realizar la grabación de los diferentes paquetes de red.

El bloque de captura es considerado por nosotros un pilar fundamental para el desarrollo de nuestro proyecto; tanto el módulo de red y el módulo SD con su respectiva memoria son incapaces de trabajar autónomamente; como mencionamos anteriormente estos dos módulos se comunican mediante una interfaz SPI, esto quiere decir que los parámetros de configuración para el modulo de red serán inicializados en el microcontrolador, siendo este quien mediante una comunicación SPI le brindara al ENC28J60 sus atributos para tener una conexión exitosa. De la misma manera el módulo SD tiene que recibir las órdenes de configuración del microcontrolador mediante la interface SPI.

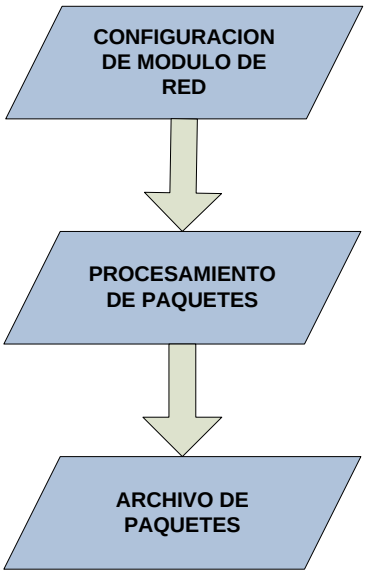


FIGURA 3. 7 DIAGRAMA DE CAPTURA

3.4 DESARROLLO DE SOFTWARE

Teniendo ya seleccionados los elementos a utilizar procedemos a la configuración para poder habilitarlos, esto se lo realizó con la ayuda de MICROBASIC®.

En la habilitación de módulo de red, se trabajo con la librería "eth_enc28j60", viene incluida en le paquete de librerías del software MICROBASIC; esta librería está compuesta por algunas subrutinas, pero entre las más importantes para nosotros son: SPI_Ethernet_Init es una rutina MAC quien inicializa al módulo ENC28J60, SPI_Ethernet_Enable es una rutina MAC que habilita el tráfico de red, SPI_Ethernet_confNetwork esta rutina configura los parámetros de red,

SPI_Ethernet_ doPacket es una rutina MAC que procesa el siguiente paquete recibido, SPI_Ethernet_ getByte es una rutina MAC que obtiene bytes de datos desde el búfer.

En la utilización de la rutinas mencionadas podemos inferir que una rutina clave para nuestro propósito de crear un capturador de paquetes fue la rutina SPI_Ethernet_doPacket que tiene diferentes opciones de operatividad sean ellas de procesar únicamente paquetes correctamente recibidos, de procesar paquetes con errores, recibir paquetes que no son IPV4 y recibir paquetes con campos desconocidos y la de más importancia para nosotros es la que nos permite trabajar en modo promiscuo, es decir que en este modo nuestro módulo de red es capaz de escuchar y capturar todo tipo de tráfico que fluye por la red.

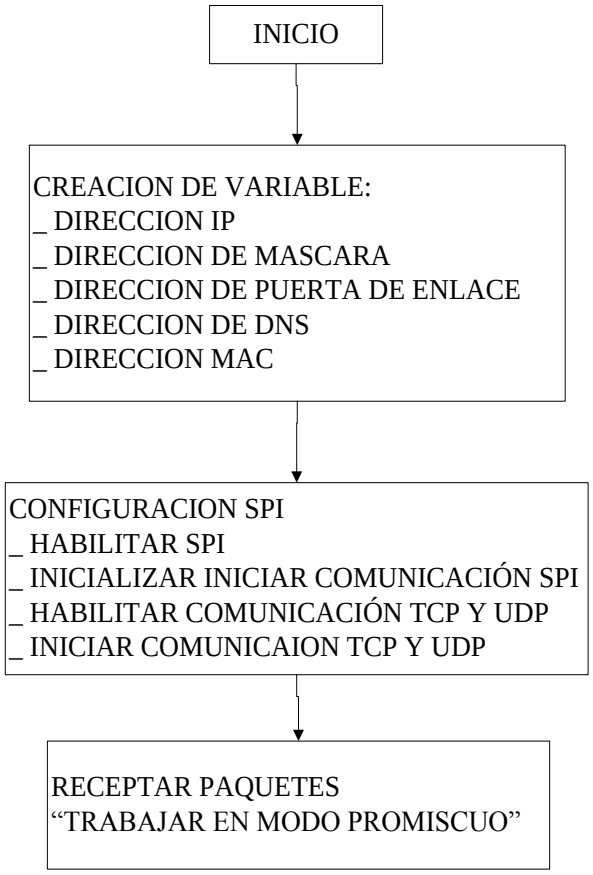


FIGURA 3. 8 DIAGRAMA DE FLUJO DEL MODULO DE RED

En la figura 3.8 se muestra un diagrama de la manera en que se configuró al módulo de red, en donde partimos de una creación de variables para dirección IP, dirección de mascara, dirección de puerta de enlace, dirección de DNS y una última variable para la creación del la dirección MAC del módulo. Ya creadas las variables se realizó la configuración de la comunicación SPI, comprendida en habilitar la señal CS para iniciar la comunicación, enviar parámetros de red que se

haya contenidos en cada una de la variables creadas y habilitar e iniciar la comunicación TCP y UDP.

En la habilitación del módulo micro SD, se trabajo con la librería "Multimedia Card Library", constituida por algunas rutina, entre las que nosotros utilizamos tenemos : Mmc_Fat_Init inicializa la librería con el hardware mediante la interfaz SPI, Mmc_Fat_Assign asigna archivo para operaciones de archivo (leer, escribir, borrar, etc), Mmc_Fat_Rewrite abre el archivo que se encuentra asignado para la escritura, Mmc_Fat_Append abre archivo que se encuentra asignado para anexar, Mmc_Fat_Write escribe el número solicitado de bytes en el fichero asignado actualmente abierto para escritura, Mmc_Fat_Set_File_Date establece la fecha y hora, Mmc_Fat_Reset procedimiento que re-establece el puntero de fichero asignado al inicio del mismo, Mmc_Fat_Read lee un byte del archivo abierto actualmente asignado a la lectura, Mmc_Fat_Delete elimina archivos asignados.

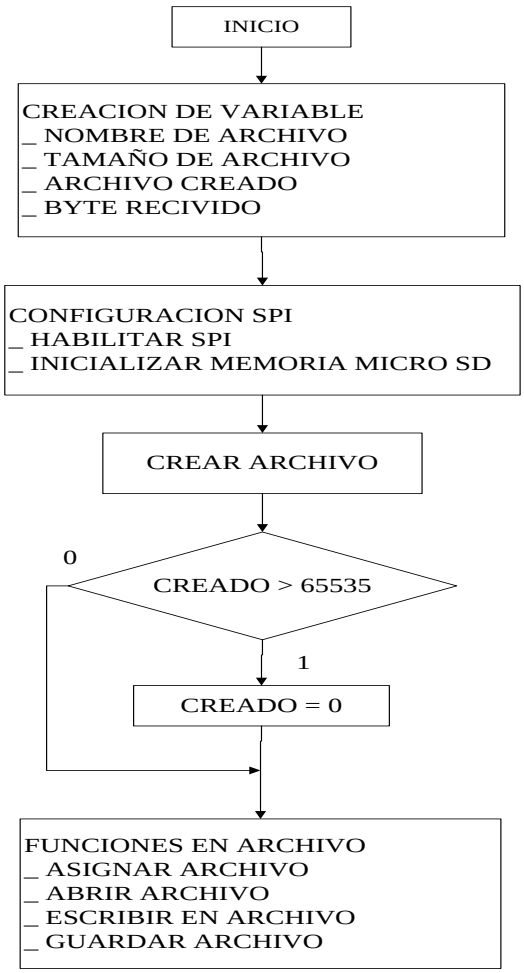


FIGURA 3. 9 DIAGRAMA DE FLUJO DEL MODULO DE MEMORIA

En la figura 3.9 se muestra un diagrama de configuración del módulo de memoria, aquí también partimos de la creación de variables para nombres de archivo, tamaño de archivo, que indique si un archivo está creado y una variable que indica si un byte fue recibido. Una vez creadas las variables se realizó la configuración de la comunicación SPI, comprendida esta en habilitar la señal CS para iniciar la

comunicación y habilitar la memoria micro SD, luego creamos un archivo y preguntamos si el archivo creado en mayor que 65535, esta cantidad se debe al límite de crear archivos por el software y este límite es la cantidad FFFF en hexadecimal correspondiente a 65535 en decimal, si archivo es mayor lo enceramos caso contrario no hacemos nada; seguido de la pregunta vienen las funciones de archivo y estas corresponde a aginar el archivo creado, abrir archivo asignado, escribir en el archivo y finalmente en la grabación del mismo.

Configurados estos dos módulos, entramos en la etapa de captura en donde al ya estar escuchando paquetes por parte del módulo de red, nuestro trabajo se fundamenta en transferir esos paquetes a nuestro repositorio o memoria.

La figura 3.10 explica el funcionamiento del programa principal en donde primeramente procedimos a la programación para configurar los módulos utilizados y puertos del microcontrolador. Posteriormente le enviamos la configuración inicial para que el dispositivo funcione correctamente, luego de los cual ingresa a un lazo infinito, en el que tenemos dos opciones: La primera opción es si los paquetes están dirigidos a la IP del prototipo y la segunda si están dirigidos a otros host. En el primer caso se captura los paquetes TCP y UDP tal cual

vienen y se procede a almacenarlos en la memoria SD. En el segundo caso se procede a capturar los datos de 255 bytes (la librería no nos permite obtener datos de mayor tamaño), debido a que una vez capturado el paquete debemos procesarlo para extraer la trama IP y desechar los bytes considerados no valido; luego de esto se lo pasará al módulo micro SD para realizar la grabación en la memoria micro SD.

En código fuente de programación en el microcontrolador para configurar a los módulos, captura de paquetes y archivo del mismo se encuentra en el Anexo F.

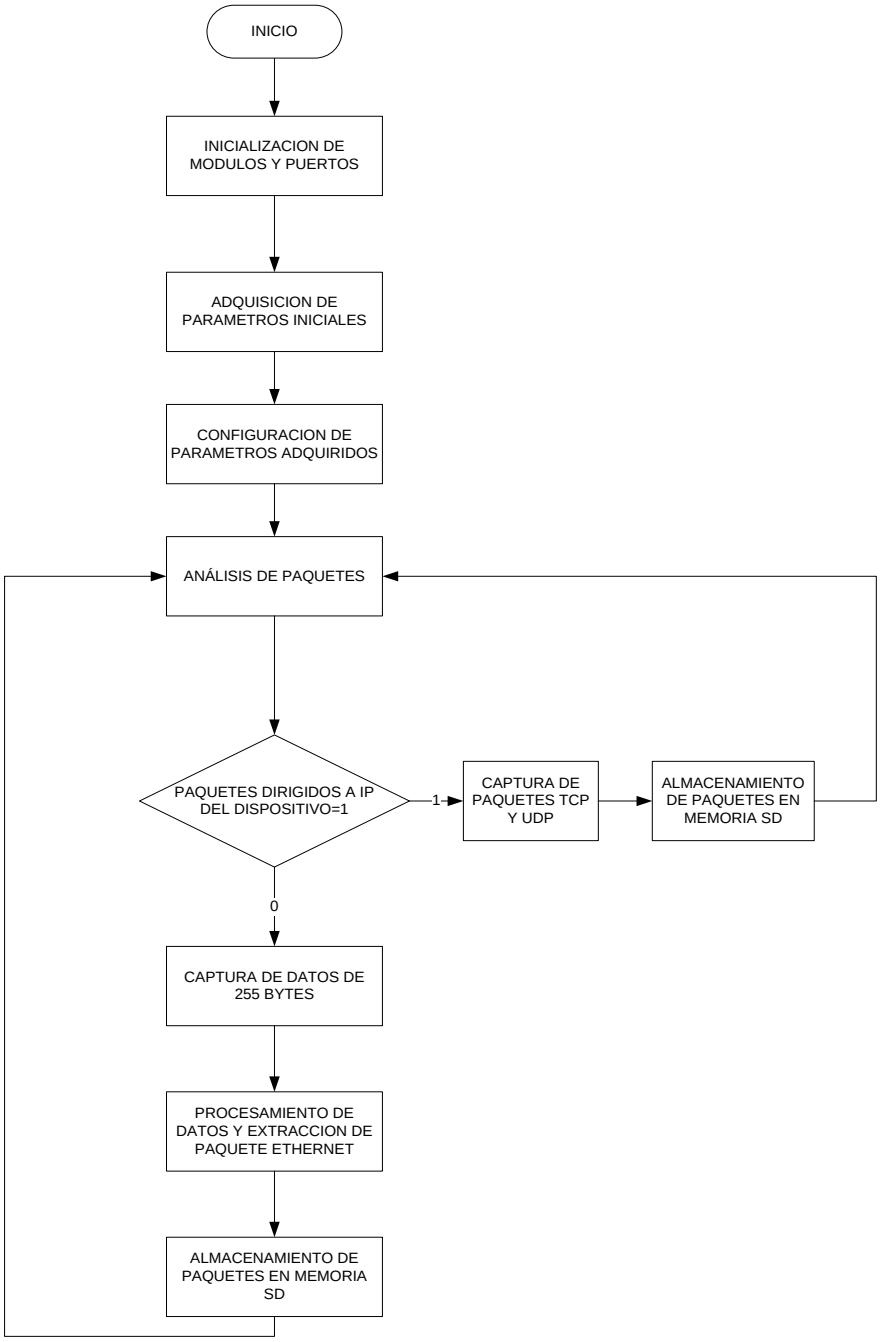


FIGURA 3. 10 DIAGRAMA DE FLUJO DEL PROGRAMA PRINCIPAL

3.5 DESARROLLO DE HARDWARE

Nuestro desarrollo de hardware se fundamenta únicamente en la elaboración de una tarjeta controladora, en la cual van sujetos los módulos, el microcontrolador, así mismo todo los elementos necesarios para su funcionamiento, como reguladores de voltaje, cristal de oscilación, leds que indican cuando se esta guardando un paquete, conmutador de encendido y apagado del prototipo.

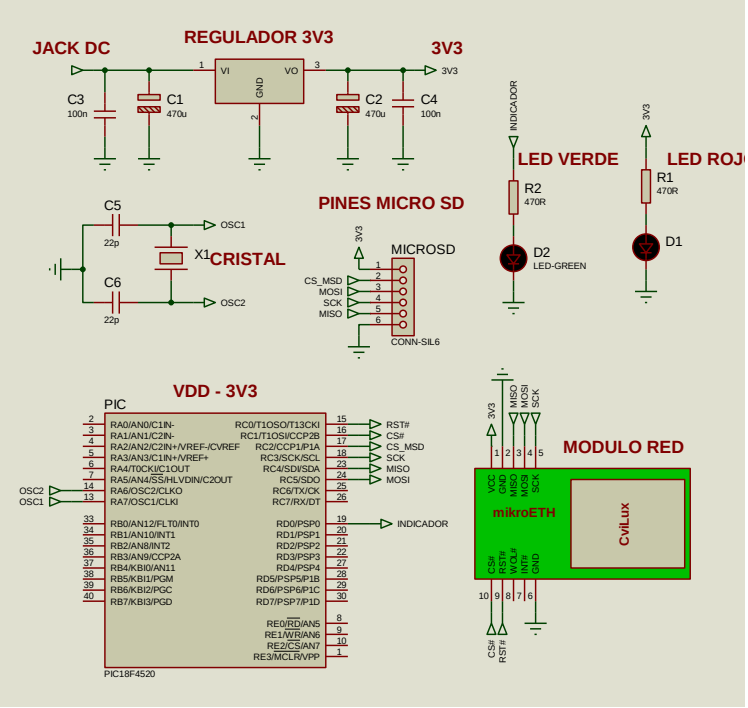


FIGURA 3. 11 DISEÑO DE TARJETA

En la figura 3.11 se observa que la tarjeta controladora está comprendida de un microcontrolador de 40 pines (PIC 18F4520), de una etapa de regulación de voltaje conjunto con un interruptor, esto con el objetivo de proteger eléctrica y electrónicamente a cada uno de los componentes de la misma; cuenta con dos leds el rojo indica que la tarjeta esta energizada y el verde indica que se está transfiriendo un dato para ser grabado en la memoria micro SD; otra etapa de la tarjeta es la configuración del cristal de 10MHz para trabajar con el microcontrolador, esta configuración ya viene dada por parte del manual del microcontrolador. La tarjeta también cuenta con pines disponible para realizar las conexiones de los módulos. (Anexo I)

CAPITULO 4

RESULTADOS EXPERIMENTALES

Se realizaron diferentes resultados experimentales con la finalidad de ver el porcentaje de eficiencia del prototipo, se comparo con un software (wireshark) que se instaló en una computadora.

4.1 ANÁLISIS DE FUNCIONAMIENTO

Utilizamos el software Wireshark para contar, capturar paquetes Ethernet y realizar gráficos estadísticos en base al número de paquetes y la clase de los mismos. Adicionalmente se utilizó el software Labview para realizar en envío de paquetes hacia el prototipo.

Antes de comenzar con el análisis de funcionamiento tenemos que definir lo que son paquetes dirigidos y no dirigidos, cuando hablemos de paquetes dirigidos serán todos los paquetes que tenga como IP

destino la IP del prototipo, y para el caso de paquetes no dirigidos serán aquellos paquetes que su IP destino difiere de la del prototipo.

Al realizar la **captura de paquetes** obtuvimos los siguientes escenarios: Primeramente enviamos **paquetes UDP dirigidos** para nuestro dispositivo, lo hicimos con la ayuda de un software creado en Labview, para ello necesitábamos tener al dispositivo conectado a la red, pos consiguiente este tendría una dirección IP, la cual necesitábamos para proporcionarle al software de labview para enviar los paquete al prototipo.

Para la captura de paquetes UDP se hizo un muestreo en un tiempo de 2 minutos con 55 segundos donde se obtuvieron un total de 64 paquetes capturados por el software wireshark de 32 paquetes generados por el software de labview.(Anexo B).

Una vez realizado el envió de paquetes UDP precedimos al siguiente escenario, en donde enviamos **paquetes TCP dirigidos** para nuestro dispositivo, lo hicimos con la ayuda de un software creado en Labview que al igual que en UDP únicamente necesitábamos la dirección IP del prototipo.

La captura de paquetes TCP que se hizo un muestreo en un tiempo de 2 minutos con 6 segundos donde se obtuvieron un total de 110 paquetes capturados por el software wireshark. (Anexo C)

Y finalmente nuestro último escenario fue al realizar la **captura de paquetes NO dirigidos** obtuvimos los siguientes escenarios:
Primeramente, desde un ordenador navegamos en internet para obtener un flujo de red capturándose este mediante el software wireshark.

Para la captura de paquetes no dirigidos se hizo un muestreo en un tiempo de 3 minutos con 10 segundos donde se obtuvieron un total de 63 paquetes capturados por wireshark.(Anexo D).

4.2 RESULTADOS DE EFICIENCIA

En base a **Paquetes UDP dirigidos,** Wireshark capturo un total de 64 paquetes. Lo cual equivale a un 100% de eficiencia en la captura de paquetes UDP dirigido.

En el escenario observamos que hay un tipo de incoherencia, al haber 32 paquetes enviados y al tener 64 paquetes capturados por el software; el motivo de este resultado fue que el software capturó los datos que son enviados a nuestro dispositivo y datos que llegan en modo promiscuo al ordenador donde está corriendo Wireshark. Estos resultados los tomamos como referencia para compararlo con nuestro prototipo.

En el escenario de Paquetes UDP Dirigidos se realizaron cuatro diferentes capturas. Cada una de ellas está distribuida en paquetes enviados por el software de labview, paquetes recibidos por nuestro prototipo y en los paquetes perdidos, en la siguiente figura (figura 4.1) se detalla cata una de las capturas.

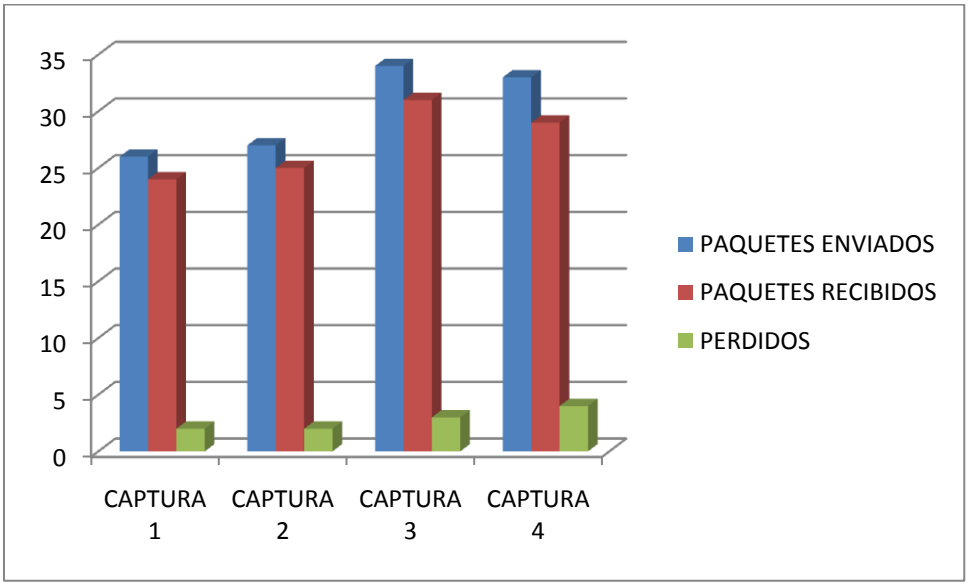


FIGURA 4. 1 RESULTADOS DEL PROTOTIPO EN UDP

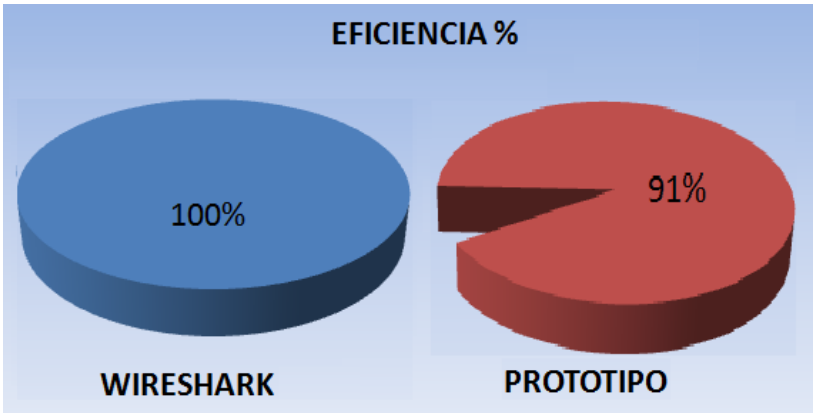


FIGURA 4. 2 EFICIENCIA DE LA CAPTURA CON PAQUETES UDP

En la figura 4.2 observamos la eficiencia del prototipo, tenemos un 91%, una de las causas de no obtener un 100% es tiempo de espera que debe existir entre paquete y paquete para el dispositivo los procese correctamente y como consecuencia tenemos una conectividad de 10Mbits.

La figura 4.3 nos muestra que el tiempo de 261 milisegundos, es el tiempo promedio entre paquete y paquete para que sean capturados correctamente por el prototipo.

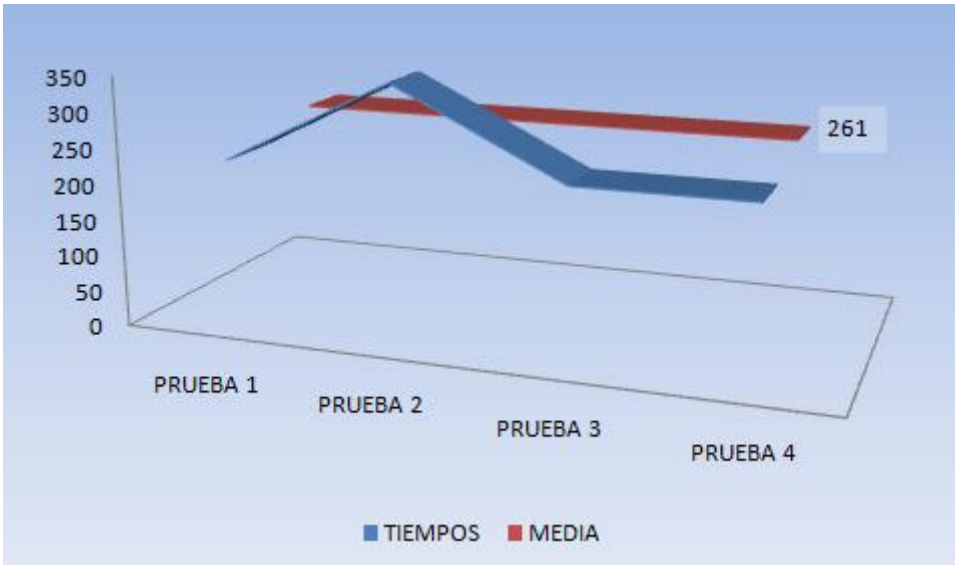


FIGURA 4. 3 TIEMPO DE ESPERA PARA PAQUETES UDP

En base a Paquetes **TCP dirigidos,** Wireshark capturo un total de 110 paquetes. Lo cual equivale a un 100% de eficiencia en la captura de paquetes TCP dirigido.

En el escenario, observamos que hay un tipo de incoherencia igual que el caso anterior, al haber 12 paquetes enviados y al tener 110 paquetes capturados por el software, el motivo de este resultado es que el

software captura los datos que son enviados a nuestro dispositivo y datos que llegan en modo promiscuo al ordenador donde está corriendo Wireshark, entre estos paquetes debemos tener en cuenta los paquetes de confirmación de recepción. Estos resultados los tomaremos como referencia para compararlo con nuestro dispositivo.

En el escenario de Paquetes TCP Dirigidos se realizaron cuatro diferentes capturas. Cada una de ellas está distribuida en paquetes enviados por el software de labview, paquetes recibidos por nuestro prototipo y en los paquetes perdidos, en la siguiente figura (figura 4.4) se detalla cata una de las capturas.

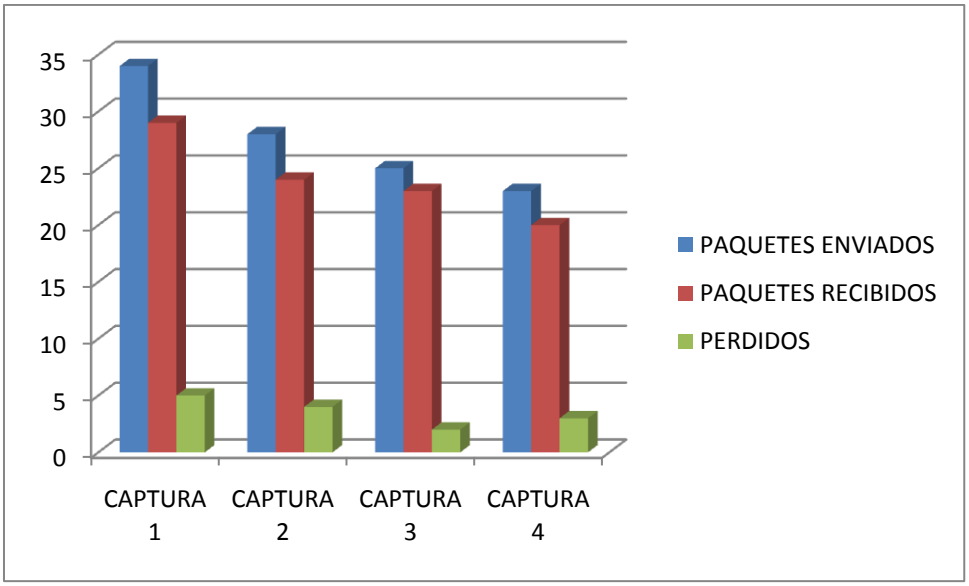


FIGURA 4. 4 RESULTADOS DEL PROTOTIPO EN TCP

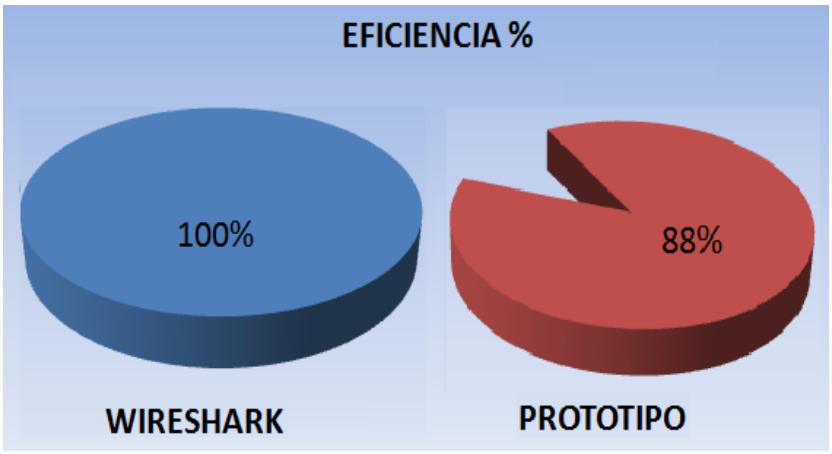


FIGURA 4. 5 EFICIENCIA DE LA CAPTURA DE PAQUETES TCP

En la figura 4.5 muestra la eficiencia del mismo siendo esta de 88% de y al igual que el caso anterior por los tiempos de espera, no logramos llegar a un 100%. La figura 4.6 nos muestra que el tiempo de 1272,75 milisegundos, es el tiempo promedio entre paquete y paquete para que sean capturados correctamente por el prototipo.

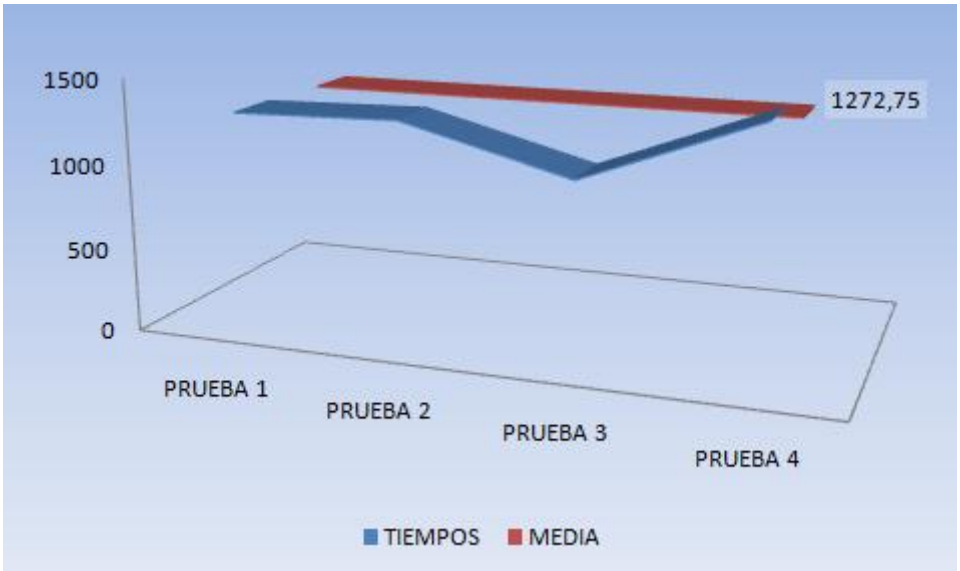


FIGURA 4. 6 TIEMPO DE ESPERA PARA PAQUETES TCP

Al observar este tiempo promedio de espera, podemos notar que es un tiempo muy elevado con respecto al obtenido en la captura de paquetes UDP; como sabemos el protocolo TCP está dedicado a conexión, utilizando algunos campos como sincronización, acuse de recibo, los cuales determinan que un paquete fue entregado correctamente y como consecuencia tenemos el procesamiento en nuestro dispositivo con tiempos de espera más elevados.

En base a Paquetes **NO dirigidos,** Wireshark capturo un total de 110 paquetes. Lo cual equivale a un 100% de eficiencia en la captura de

paquetes NO dirigido, estos resultados los tomaremos como referencia para compararlo con nuestro dispositivo.

En el escenario de Paquetes NO Dirigidos se realizaron cuatro diferentes capturas. Cada una de ellas está distribuida en paquetes generados por un usuario que navego por la red, paquetes recibidos por nuestro prototipo y en los paquetes perdido, en la siguiente figura (figura 4.7) se detalla cata una de las capturas.

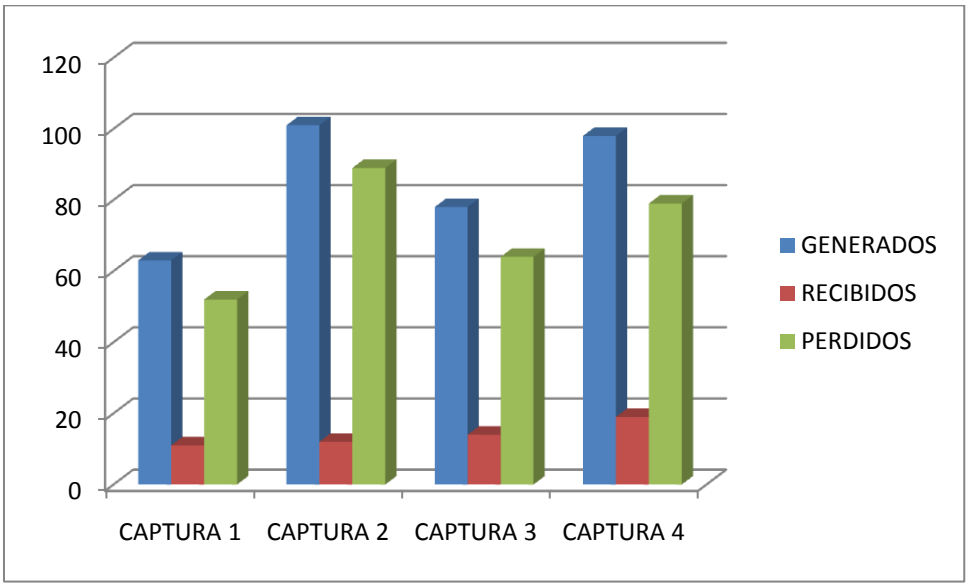


FIGURA 4. 7 RESULTADOS DEL DISPOSITIVO EN PAQUETES NO DIRIGIDOS

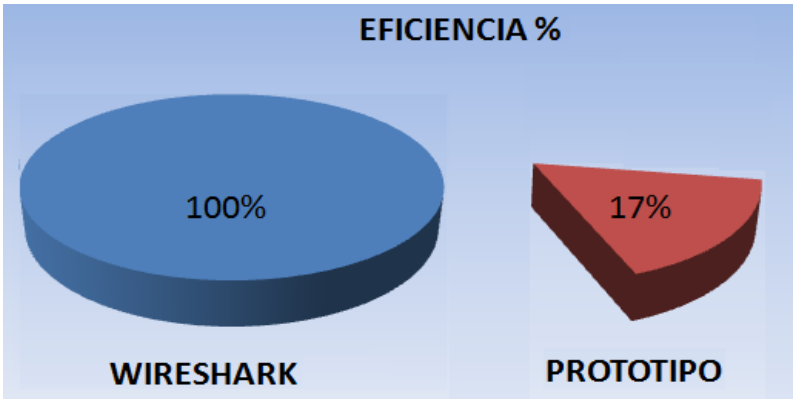


FIGURA 4. 8 EFICIENCIA DE LA CAPTURA DE PAQUETES NO DIRIGIDOS

Se realizaron varias pruebas en la cuales se generó tráfico de red navegando por algunos sitios web, estos paquetes fueron recibidos por el prototipo, estos datos están resumidos en la figura 4.7, donde podemos evaluar la eficiencia, siendo esta de 17% figura 4.8, igual que en los casos ya revisados los tiempos de espera son una limitante para obtener 100%.

CONCLUSIONES

1. El conocer las direcciones IP visitadas por empleados, lo podemos realizar con la captura de paquetes de datos de la red previamente configurada en nuestro prototipo para su posterior análisis.

2. Debido a la construcción del prototipo con componentes de electrónica básica puede ser considerado de bajo costo y fácil ensamblaje, si se lo fabricase a gran escala se reducirían los costos un cincuenta por ciento (50%) , convirtiéndolo de bajo costo.

3. En el transcurso de desarrollo y elaboración del proyecto nos encontramos con dos inconvenientes, siendo éstos la elección de un módulo de red con conectividad de 10 Mbits y el otro inconveniente era que tanto el módulo de red y el módulo de micro SD trabajaban con comunicación SPI, causando esto conflictos para nuestro microcontrolador ya que él solo posee un modulo SPI, esto quiere decir que el no puede manipular dos módulos en el mismo instante. Entonces se concluye que cuando él está escuchando la red, él no puede grabar datos y viceversa, causando que por ciertos instantes el prototipo no está escuchando a la red, perdiendo así paquetes que se reflejan en la eficiencia del prototipo.

RECOMENDACIONES

1. Se recomienda utilizar módulos o tarjetas de red de mayor velocidad (, como por ejemplo una tarjeta FPGA), debido a la limitante en cuánto a velocidad del prototipo la cuál es de 10 Mbits/s.

2. Emplear un microcontrolador que pueda ofrecer dos interfaces SPI para estar enlazados tanto al módulo de red y al módulo de la memoria micro SD para poder realizar las capturas y almacenamiento de datos simultáneamente.

3. Se recomienda utilizar módulos con mayor capacidad de memoria de grabación para evitar tener que reemplazarla al encontrarse llena, teniendo un promedio de paquetes de 64 bytes podemos grabar un total de 2 millones de paquetes en una tarjeta de 1 Gigabit.

ANEXO A

PROTOCOLOS UTILIZADOS

Protocolo de Resolución de Direcciones (ARP)

Es un protocolo de nivel de enlace responsable de encontrar la dirección hardware (Ethernet MAC) que corresponde a una determinada dirección IP. Para ello se envía un paquete (ARP request) a la dirección de difusión de la red (broadcast (MAC = FF FF FF FF FF FF)) que contiene la dirección IP por la que se pregunta, y se espera a que esa máquina (u otra) responda (ARP reply) con la dirección Ethernet que le corresponde. Cada máquina mantiene una caché con las direcciones traducidas para reducir el retardo y la carga. ARP permite a la dirección de Internet ser independiente de la dirección Ethernet, pero esto sólo funciona si todas las máquinas lo soportan.

En Ethernet, la capa de enlace trabaja con direcciones físicas. El protocolo ARP se encarga de traducir las direcciones IP a direcciones MAC (direcciones físicas).Para realizar ésta conversión, el nivel de enlace utiliza las tablas ARP, cada interfaz tiene tanto una dirección IP como una dirección física MAC.

ARP se utiliza en 4 casos referentes a la comunicación entre 2 hosts:

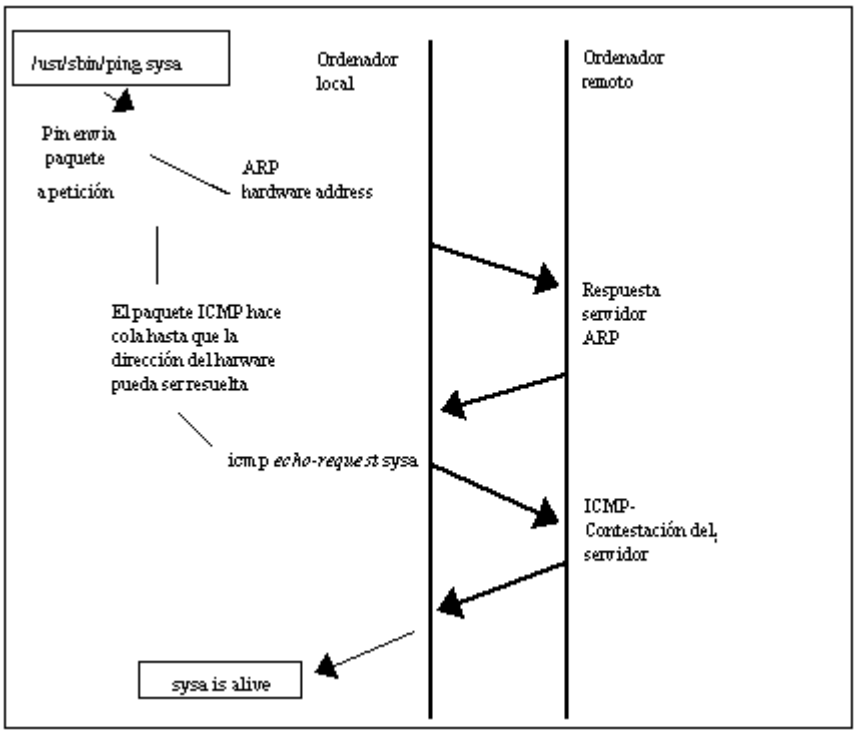
1. Cuando 2 hosts están en la misma red y uno quiere enviar un paquete a otro.
2. Cuando 2 host están sobre redes diferentes y deben usar un gateway/enrutador para alcanzar otro host.
3. Cuando un enrutador necesita enviar un paquete a un host a través de otro enrutador.

4. Cuando un enrutador necesita enviar un paquete a un host de la misma red.

Funcionamiento

Si A quiere enviar una trama a la dirección IP de B (misma red), mirará su tabla ARP para poner en la trama la dirección destino física correspondiente a la IP de B. De esta forma, cuando les llegue a todos la trama, no tendrán que deshacerla para comprobar si el mensaje es para ellos, sino que se hace con la dirección física.

Si A quiere enviar un mensaje a C (un nodo que no esté en la misma red), el mensaje deberá salir de la red. Así, A envía la trama a la dirección física de salida del enrutador. Esta dirección física la obtendrá a partir de la IP del enrutador, utilizando la tabla ARP. Si esta entrada no está en la tabla, mandará un mensaje ARP a esa IP (llegará a todos), para que le conteste indicándole su dirección física.



Protocolo de Internet (IP)

Es uno de los protocolos de Internet más importantes ya que permite el desarrollo y transporte de datagramas de IP (paquetes de datos), aunque sin garantizar su "entrega". El protocolo IP determina el destinatario del mensaje mediante 3 campos:

- el campo de dirección IP: Dirección del equipo;
- el campo de máscara de subred: una máscara de subred le permite al protocolo IP establecer la parte de la dirección IP que se relaciona con la red;

- el campo de pasarela predeterminada: le permite al protocolo de Internet saber a qué equipo enviar un datagrama, si el equipo de destino no se encuentra en la red de área local.

<table>
<tr><td>Versión
(4 bits)</td><td>Longitud del encabezado
(4 bits)</td><td>Tipo de servicio
(8 bits)</td><td colspan="2">Longitud total
(16 bits)</td></tr>
<tr><td colspan="3">Identificación
(16 bits)</td><td>Indicador
(3 bits)</td><td>Margen del fragmento
(13 bits)</td></tr>
<tr><td colspan="2">Tiempo de vida
(8 bits)</td><td>Protocolo
(8 bits)</td><td colspan="2">Suma de comprobación del encabezado
(16 bits)</td></tr>
<tr><td colspan="5">Dirección IP de origen (32 bits)</td></tr>
<tr><td colspan="5">Dirección IP de destino (32 bits)</td></tr>
<tr><td colspan="5">Datos</td></tr>
</table>

•**Versión (4 bits):** es la versión del protocolo IP que se está utilizando (actualmente se utiliza la versión 4 IPv4) para verificar la validez del datagrama. Está codificado en 4 bits.

•**Longitud del encabezado o IHL por Internet Header Length (Longitud del encabezado de Internet) (4 bits):** es la cantidad de palabras de 32 bits que componen el encabezado (Importante: el valor mínimo es 5). Este campo está codificado en 4 bits.

•**Tipo de servicio (8 bits)**: indica la forma en la que se debe procesar el datagrama.

•**Longitud total (16 bits):** indica el tamaño total del datagrama en bytes. El tamaño de este campo es de 2 bytes, por lo tanto el tamaño total del datagrama no puede exceder los 65536 bytes. Si se lo utiliza junto con el tamaño del encabezado, este campo permite determinar dónde se encuentran los datos.

•**TTL o Tiempo de vida (8 bits):** este campo especifica el número máximo de enrutadors por los que puede pasar un datagrama. Por lo tanto, este campo disminuye con cada paso por un enrutador y cuando alcanza el valor crítico de 0, el enrutador destruye el datagrama. Esto evita que la red se sobrecargue de datagramas perdidos.

•**Protocolo (8 bits):** este campo, en notación decimal, permite saber de qué protocolo proviene el datagrama.

- ICMP 1
- IGMP: 2
- TCP: 6
- UDP: 17

•**Suma de comprobación del encabezado (16 bits):** este campo contiene un valor codificado en 16 bits que permite controlar la integridad del encabezado para establecer si se ha modificado durante la transmisión. La suma de comprobación es la suma de todas las palabras de 16 bits del encabezado (se excluye el campo suma de comprobación). Esto se realiza de tal modo que cuando

se suman los campos de encabezado (suma de comprobación inclusive), se obtenga un número con todos los bits en 1.

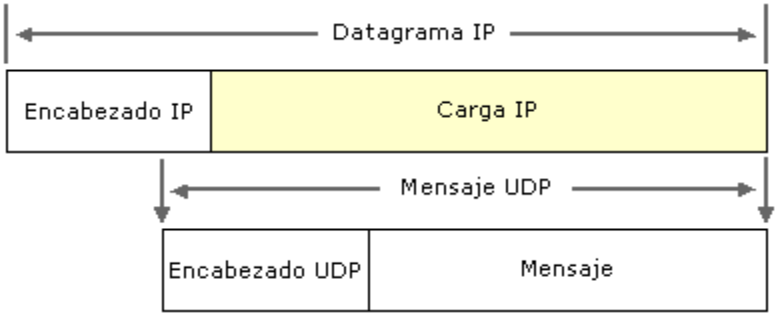
•**Dirección IP de origen (32 bits):** Este campo representa la dirección IP del equipo remitente y permite que el destinatario responda.

•**Dirección IP de destino (32 bits):** dirección IP del destinatario del mensaje.

Protocolo de Datagramas 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,



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.

ANEXO B

DETALLES DE PAQUETES UDP

CAPTURADOS

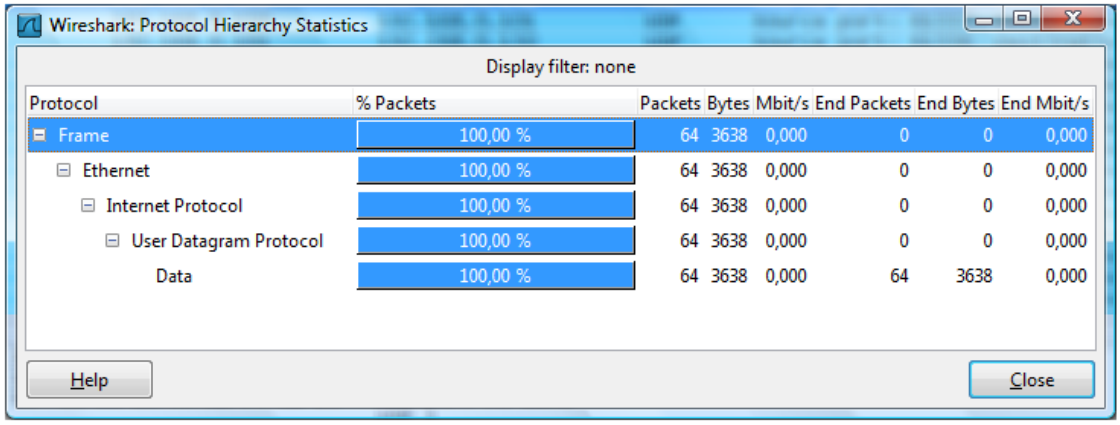
Observamos también el tipo de trama que para nuestro caso es IP (0x800); la versión del protocolo de Internet que es 4; la longitud de la trama que es 39, 027 en hexadecimal; el tiempo de vida de 128, 64 en hexadecimal; el checksum de la cabecera que en nuestro caso es e353, entre los parámetros más importantes de la trama.

```
0000  00 14 a5 76 19 3f 00 23  5a a6 64 6f 08 00 45 00    ...v.?.# Z.do..E.
0010  00 27 01 c5 00 00 80 11  b6 b5 c0 a8 00 65 c0 a8    .'...... .....e..
0020  00 96 f0 74 f0 75 00 13  e3 53 20 44 61 74 61 20    ...t.u.. .S Data
0030  55 44 50 20 31                                      UDP 1
```

A continuación tenemos una trama de un paquete UDP cuya longitud es variable; la trama comienza con los campos de los puertos fuente y destino son de 16 bits que identifican el proceso de origen y recepción. Debido a que UDP carece de un servidor de estado y el origen no solicita respuestas, el puerto origen es opcional. En caso de no ser utilizado, el puerto origen debe ser puesto a cero. A los campos del puerto del puerto destino le sigue un campo obligatorio que indique el tamaño en bytes del datagrama UDP incluidos los datos. El valor mínimo es 8 bytes. El campo de la cabecera restante es una suma de comprobación de 16 bits que abarca un pseudo-cabecera IP (con las IP origen y destino, el protocolo y la longitud del paquete UDP).

```
⊞ Frame 3 (53 bytes on wire, 53 bytes captured)
⊟ Ethernet II, Src: CompalIn_a6:64:6f (00:23:5a:a6:64:6f), Dst: GemtekTe_76:19:3f (00:14:a5:76:19:3f)
  ⊟ Destination: GemtekTe_76:19:3f (00:14:a5:76:19:3f)
      Address: GemtekTe_76:19:3f (00:14:a5:76:19:3f)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
  ⊞ Source: CompalIn_a6:64:6f (00:23:5a:a6:64:6f)
    Type: IP (0x0800)
⊟ Internet Protocol, Src: 192.168.0.101 (192.168.0.101), Dst: 192.168.0.150 (192.168.0.150)
    Version: 4
    Header length: 20 bytes
  ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 39
    Identification: 0x01c5 (453)
  ⊞ Flags: 0x00
    Fragment offset: 0
    Time to live: 128
    Protocol: UDP (0x11)
  ⊞ Header checksum: 0xb6b5 [correct]
    Source: 192.168.0.101 (192.168.0.101)
    Destination: 192.168.0.150 (192.168.0.150)
⊟ User Datagram Protocol, Src Port: 61556 (61556), Dst Port: 61557 (61557)
    Source port: 61556 (61556)
    Destination port: 61557 (61557)
    Length: 19
  ⊞ Checksum: 0xe353 [validation disabled]
⊟ Data (11 bytes)
    Data: 204461746120544502031
```

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% y el UDP en un 100%.



ANEXO C

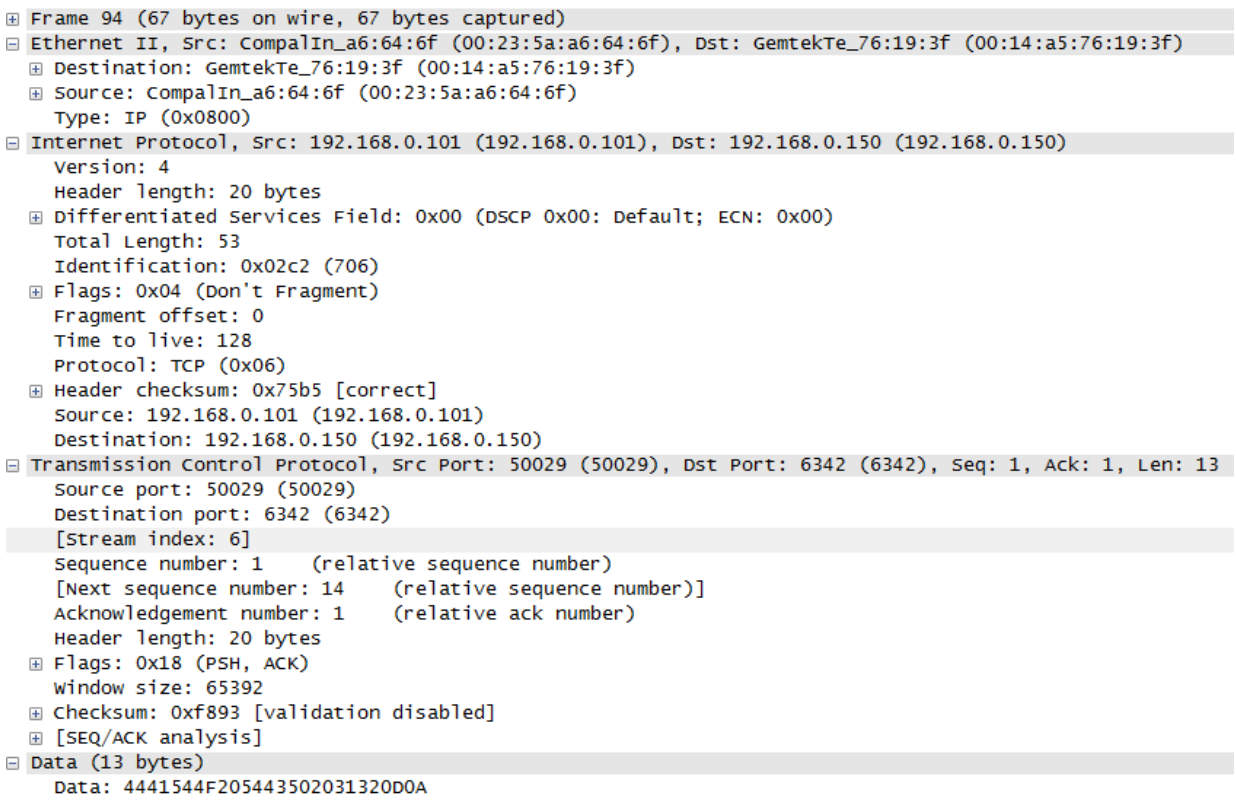
DETALLES DE PAQUETES TCP

CAPTURADOS

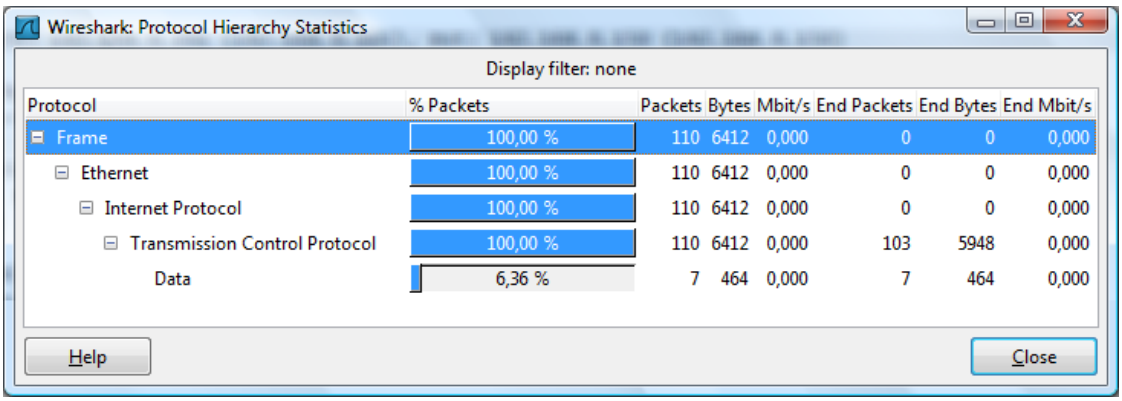
Observamos también el tipo de trama que para nuestro caso es IP (0x800); la versión del protocolo de Internet que es 4; la longitud de la trama que es 53, 035 en hexadecimal; el tiempo de vida de 128, 64 en hexadecimal; el checksum de la cabecera que en nuestro caso es 75b5, entre los parámetros más importantes de la trama.

```
0000  00 14 a5 76 19 3f 00 23  5a a6 64 6f 08 00 45 00      ...v.?.# Z.do..E.
0010  00 35 02 c2 40 00 80 06  75 b5 c0 a8 00 65 c0 a8      .5..@... u....e..
0020  00 96 c3 6d 18 c6 3f 4b  a7 13 19 dd 00 8c 50 18      ...m..?K ......P.
0030  ff 70 f8 93 00 00 44 41  54 4f 20 54 43 50 20 31      p.DATO TCP  1
0040  32 0d 0a                                            2..
```

A continuación tenemos una trama de un paquete TCP cuya longitud es variable; la trama comienza con los campos de los puertos fuente y destino son de 16 bits que identifican el proceso de origen y recepción, seguido de estos dos campos tiene un numero de secuencia cuyo objetivo es para comprobar si un paquete se ha perdido y que llegue en el orden correcto; también tiene un campo se número de acuse de recibo, siendo este el que tiene el número de secuencia del siguiente segmento; longitud de cabecera; reservado; bits de control; ventana, que especifica el número de bytes que el receptor está actualmente esperando recibir; checksum; puntero urgente; un campo opcional y finalmente el dato.


Frame 94 (67 bytes on wire, 67 bytes captured)
Ethernet II, Src: CompalIn_a6:64:6f (00:23:5a:a6:64:6f), Dst: GemtekTe_76:19:3f (00:14:a5:76:19:3f)
Destination: GemtekTe_76:19:3f (00:14:a5:76:19:3f)
Source: CompalIn_a6:64:6f (00:23:5a:a6:64:6f)
Type: IP (0x0800)
Internet Protocol, Src: 192.168.0.101 (192.168.0.101), Dst: 192.168.0.150 (192.168.0.150)
Version: 4
Header length: 20 bytes
Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
Total Length: 53
Identification: 0x02c2 (706)
Flags: 0x04 (Don't Fragment)
Fragment offset: 0
Time to live: 128
Protocol: TCP (0x06)
Header checksum: 0x75b5 [correct]
Source: 192.168.0.101 (192.168.0.101)
Destination: 192.168.0.150 (192.168.0.150)
Transmission Control Protocol, Src Port: 50029 (50029), Dst Port: 6342 (6342), Seq: 1, Ack: 1, Len: 13
Source port: 50029 (50029)
Destination port: 6342 (6342)
[Stream index: 6]
Sequence number: 1 (relative sequence number)
[Next sequence number: 14 (relative sequence number)]
Acknowledgement number: 1 (relative ack number)
Header length: 20 bytes
Flags: 0x18 (PSH, ACK)
Window size: 65392
Checksum: 0xf893 [validation disabled]
[SEQ/ACK analysis]
Data (13 bytes)
Data: 4441544F20544350203132300D0A

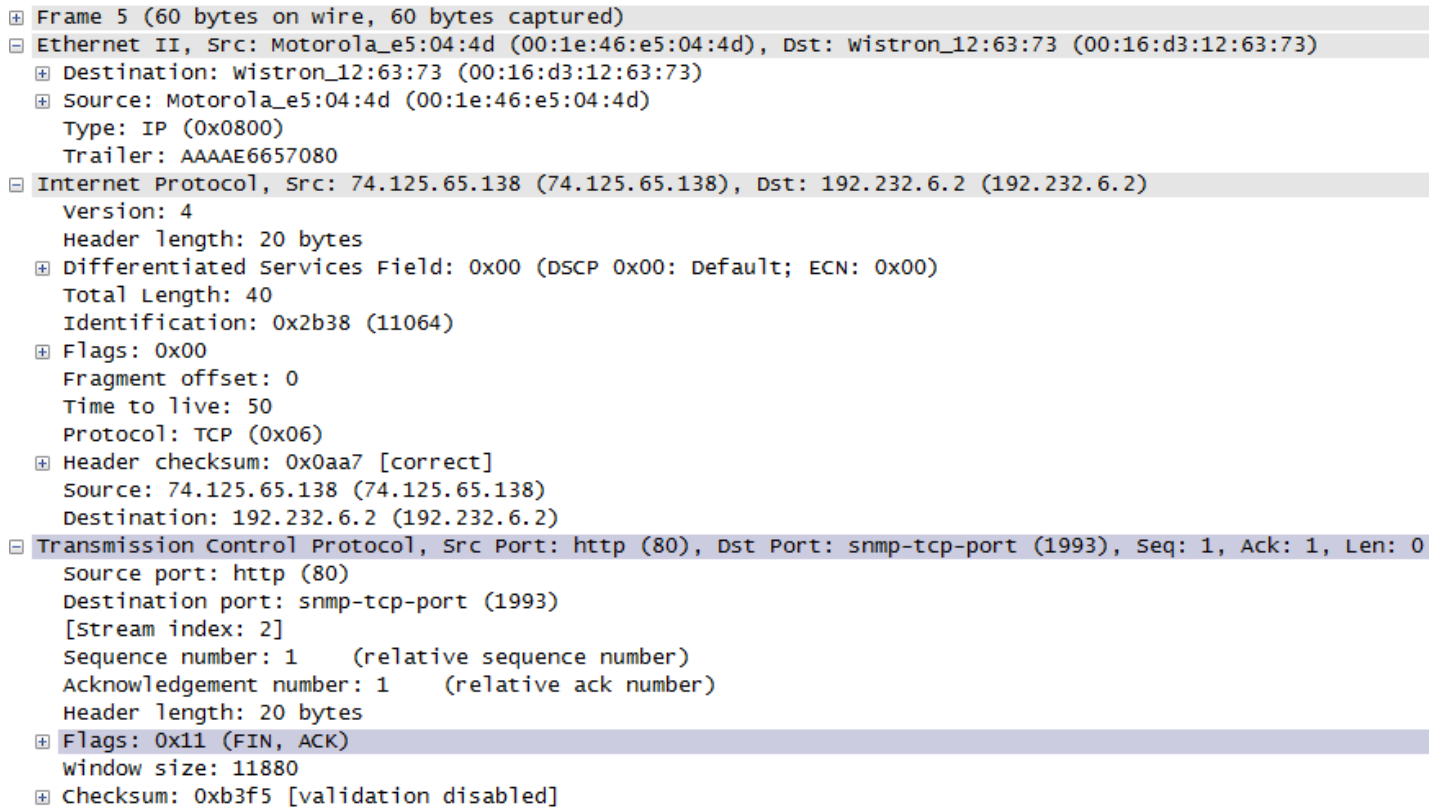
En el siguiente gráfico observamos, la jerarquía de los protocolos utilizados en la transmisión de paquetes en los que, como las cifras lo indican, el protocolo Ethernet se encuentra en el 100% de los paquetes, el protocolo Internet también se encuentra en un 100% y el TCP en un 100%.



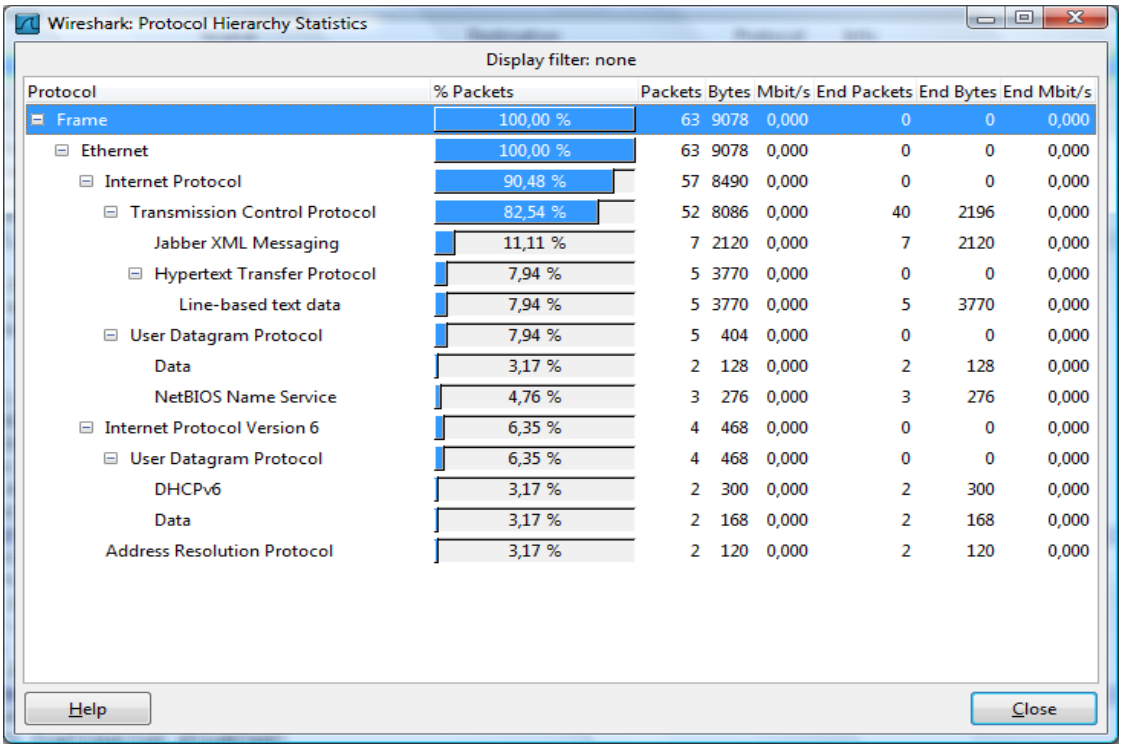
ANEXO D

DETALLES DE PAQUETES NO DIRIGIDOS CAPTURADOS

A continuación tenemos una trama de un paquete TCP no dirigido cuya longitud es variante; la trama comienza con los campos de los puertos fuente y destino son de 16 bits que identifican el proceso de origen y recepción, seguido de estos dos campos tiene un número de secuencia cuyo objetivo es para comprobar si un paquete se ha perdido y que llegue en el orden correcto; también tiene un campo de número de acuse de recibo, siendo éste el que tiene el número de secuencia del siguiente segmento; longitud de cabecera; reservado; bits de control; ventana, que especifica el número de bytes que el receptor esta actualmente esperando recibir; checksum; puntero urgente; un campo opcional y finalmente el dato.


Frame 5 (60 bytes on wire, 60 bytes captured)
Ethernet II, Src: Motorola_e5:04:4d (00:1e:46:e5:04:4d), Dst: Wistron_12:63:73 (00:16:d3:12:63:73)
Destination: Wistron_12:63:73 (00:16:d3:12:63:73)
Source: Motorola_e5:04:4d (00:1e:46:e5:04:4d)
Type: IP (0x0800)
Trailer: AAAAE6657080
Internet Protocol, Src: 74.125.65.138 (74.125.65.138), Dst: 192.232.6.2 (192.232.6.2)
Version: 4
Header length: 20 bytes
Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
Total Length: 40
Identification: 0x2b38 (11064)
Flags: 0x00
Fragment offset: 0
Time to live: 50
Protocol: TCP (0x06)
Header checksum: 0x0aa7 [correct]
Source: 74.125.65.138 (74.125.65.138)
Destination: 192.232.6.2 (192.232.6.2)
Transmission Control Protocol, Src Port: http (80), Dst Port: snmp-tcp-port (1993), Seq: 1, Ack: 1, Len: 0
Source port: http (80)
Destination port: snmp-tcp-port (1993)
[Stream index: 2]
Sequence number: 1 (relative sequence number)
Acknowledgement number: 1 (relative ack number)
Header length: 20 bytes
Flags: 0x11 (FIN, ACK)
Window size: 11880
Checksum: 0xb3f5 [validation disabled]

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 se encuentra en un 90,48%, el TCP en un 82,54%, el protocolo HTTP en un 7,94%, UDP en 7,94% y ARP en un 3,17%.



ANEXO E

DETALLES DE PAQUETES ARP

CAPTURADOS

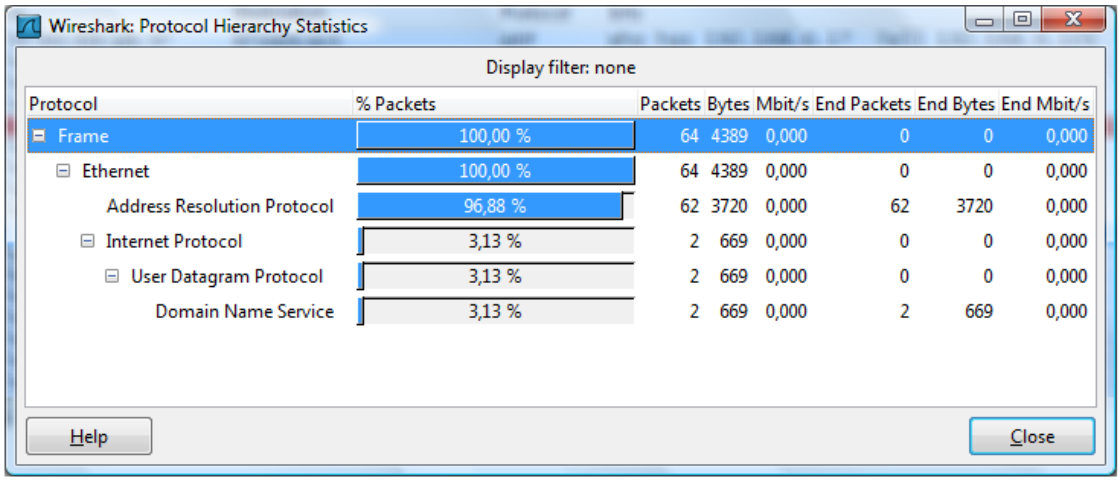
Observamos también el tipo de trama que para nuestro caso es IP (0x800); la longitud de la trama que es 60; entre los parámetros más importantes de la trama.

```
0000  ff ff ff ff ff ff 48 5d  60 69 a9 97 08 06 00 01        ......H] `i......
0010  08 00 06 04 00 01 48 5d  60 69 a9 97 c0 a8 00 67        ......H] `i.....g
0020  00 00 00 00 00 00 c0 a8  00 01 82 b1 d9 34 b4 c7        ........ .....4..
0030  7a 12 d7 2a 24 59 01 bd  00 00 00 00                    z..*$Y.. ....
```

A continuación tenemos una trama de un paquete ARP cuya longitud es de 60 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 48:5d:60:69:a9:97.

```
⊟ Frame 10 (60 bytes on wire, 60 bytes captured)
    Arrival Time: Apr 26, 2011 22:55:15.657127000
    [Time delta from previous captured frame: 5.038297000 seconds]
    [Time delta from previous displayed frame: 5.038297000 seconds]
    [Time since reference or first frame: 31.431810000 seconds]
    Frame Number: 10
    Frame Length: 60 bytes
    Capture Length: 60 bytes
    [Frame is marked: False]
    [Protocols in frame: eth:arp]
    [Coloring Rule Name: ARP]
    [Coloring Rule String: arp]
⊟ Ethernet II, Src: 48:5d:60:69:a9:97 (48:5d:60:69:a9:97), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
  ⊞ Destination: Broadcast (ff:ff:ff:ff:ff:ff)
  ⊞ Source: 48:5d:60:69:a9:97 (48:5d:60:69:a9:97)
    Type: ARP (0x0806)
    Trailer: 82B1D934B4C77A12D72A245901BD00000000
⊟ Address Resolution Protocol (request)
    Hardware type: Ethernet (0x0001)
    Protocol type: IP (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (0x0001)
    Sender MAC address: 48:5d:60:69:a9:97 (48:5d:60:69:a9:97)
    Sender IP address: 192.168.0.103 (192.168.0.103)
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 192.168.0.1 (192.168.0.1)
```

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 ARP se encuentra en un 100%, el protocolo Internet también se encuentra en un 3.13%, el UDP en un 3.13%, y el DNS en un 3,13% entre los más importantes.



ANEXO F

ARCHIVOS FUENTE DE LA PROGRAMACION DEL MICROCONTROLADOR

```
program Codigo_Ethernet_MicroSD

include "eth_enc28j60_utils"        ' LIBRERIA DE MODULO ENC28J60
include "eth_enc28j60"              ' LIBRERIA DE MODULO ENC28J60
'************************ RAM VARIABLES ***************************
dim myMacAddr   as byte[6]          ' my MAC address
   myIpAddr    as byte[4]           ' my IP address
   gwIpAddr    as byte[4]           ' gateway (router) IP address
   ipMask      as byte[4]           ' network mask (for example : 255.255.255.0)
   dnsIpAddr   as byte[4]           ' DNS server IP address
   IP_PC       as byte[4]

dim fat_txt as STRING[20]               ' "FAT16 not found",
dim file_contents as STRING[50]
dim filename as STRING[14]              '"MIKRO00xTXT"
dim caracter,flag as byte
dim i, size as dword
dim Buffer as STRING[512]
dim received_byte as char
dim archivo_creado   as word
'******* CREACION DE NUEVO ARCHIVO PARA ESCRIBIR DATOS *****
sub procedure M_Create_New_File()
   Mmc_Fat_Assign(filename, 0xA0)
   Mmc_Fat_Rewrite()                    ' To clear file and start with new data
```

```
end sub
'***** ESCRITURA DE ARCHIVO DESPUES DE LA ULTIMA LINEA ****
sub procedure M_Write_File()
     Mmc_Fat_Append()
     Mmc_Fat_Write(dato, cant_datos)
end sub
'**** CREACION DE ALGUNOS ARCHIVOS Y ESCRITURA DE DATOS ***
sub procedure M_Create_Multiple_Files()
  if(archivo_creado > 65535)then
     archivo_creado=0
  else
   filename[0] = (archivo_creado / 10000) + 48
   filename[1] = ((archivo_creado / 1000) mod 10) +48
   filename[2] = ((archivo_creado / 100) mod 10) +48
   filename[3] = ((archivo_creado / 10) mod 10) +48
   filename[4] = (archivo_creado mod 10) +48

   Mmc_Fat_Assign(filename, 0xA0)
   Mmc_Fat_Rewrite()
   Mmc_Fat_Write(dato , cant_datos )
   inc(archivo_creado)
 end if
end sub
'**** ABRIR ARCHIVOS EXISTENTES Y REESCRITURA DE ESTOS *****
```

```
sub procedure M_Open_File_Rewrite()
  dim loop_ as byte
   filename[7] = "C"
   Mmc_Fat_Assign(filename, 0)
   Mmc_Fat_Rewrite()
   for loop_ = 1 to 55
    file_contents[0] = loop_ div 10 + 64
    file_contents[1] = loop_ mod 10 + 64
    Mmc_Fat_Write(file_contents, 42) ' write data to the assigned file
   next loop_
end sub
'************* ABRIR ARCHIVOS Y AÑADIR DATOS ********************
sub procedure M_Open_File_Append()
    filename[7] = "B"
    Mmc_Fat_Assign(filename, 0)
    Mmc_Fat_Set_File_Date(2010,10,10,10,35,0)
    Mmc_Fat_Append()                    ' PREPARA ARCHIVO PARA AÑADIR
   Mmc_Fat_Write(" for mikroElektronika 2010\n", 27)    ' Write data to
assigned file
end sub
'* ABRIR ARCHIVOS, LEER DATOS DE ESTE Y PONER ESTE EN USART *
sub procedure M_Open_File_Read()
 filename[7] = "B"
 Mmc_Fat_Assign(filename, 0)
```

```
  Mmc_Fat_Reset(size)          ' To read file, sub procedure returns size of file
  for i = 1 to size
    Mmc_Fat_Read(caracter)
  next i
end sub
'********************** ELIMINAR UN ARCHIVO **********************
sub procedure M_Delete_File()
  filename[7] = "F"
  Mmc_Fat_Assign(filename, 0)
  Mmc_Fat_Delete()
end sub
'********************** CONFIGURACION DE SPI **********************
sub procedure Configuacion_SPI_Ethernet()
  PORTC.2 = 1
  Spi_init
  Spi_Ethernet_Init(PORTC, 0, PORTC, 1, myMacAddr, myIpAddr,
Spi_Ethernet_FULLDUPLEX)
   Spi_Ethernet_confNetwork(ipMask, gwIpAddr, dnsIpAddr)
end sub
'********************CONFIGURACIONN DE MICRO SD****************
sub procedure Configuacion_SPI_MMC()
    PORTC.1 = 1
    Spi_Init_Advanced(MASTER_OSC_DIV64,       DATA_SAMPLE_MIDDLE,
CLK_IDLE_LOW, LOW_2_HIGH)
    Delay_ms(500)
```

```
    if Mmc_Fat_Init(PORTC, 2) = 0 then
     Spi_Init_Advanced(MASTER_OSC_DIV4,        DATA_SAMPLE_MIDDLE,
CLK_IDLE_LOW, LOW_2_HIGH)
     Delay_ms(500)
     M_Create_Multiple_Files()
    end if
end sub
'***************************** PRINCIPAL ***************************
main:
    fat_txt = "FAT16 not found"
    file_contents = "XX MMC/SD FAT16 library by Anton Rieckert\n"
    filename = "xxxxxTXT"
    '--- prepare PORTB for signalling
    ADCON1 = 0x0F  'set all as Digital
    CMCON  = 0x07          ' turn off comparators
    INTCON = %00000000
    INTCON2 = %00000000
    PIE1=0
    PIR1=0
    IPR1=0
    TRISA = 0
    TRISB = 0
    TRISC=%10010000
    TRISD = 0
```

PORTA=0

PORTB = 0

PORTC=0

PORTD=0

archivo_creado=0

flag=0

dato_recibido=0

myMacAddr[0] = 0x00

myMacAddr[1] = 0x14

myMacAddr[2] = 0xA5

myMacAddr[3] = 0x76

myMacAddr[4] = 0x19

myMacAddr[5] = 0x3F

myIpAddr[0] = 192

myIpAddr[1] = 232

myIpAddr[2] = 6

myIpAddr[3] = 10

gwIpAddr[0] = 192

gwIpAddr[1] = 232

gwIpAddr[2] = 6

```
gwIpAddr[3]  = 1


dnsIpAddr[0] = 200
dnsIpAddr[1] = 124
dnsIpAddr[2] = 224
dnsIpAddr[3] = 195


ipMask[0]    = 255
ipMask[1]    = 255
ipMask[2]    = 255
ipMask[3]    = 0


Configuacion_SPI_Ethernet()
while(1)
    if( Spi_Ethernet_doPacket()=2)then
             cant_datos=0
            for j = 0 to 511
              dato[cant_datos] = Spi_Ethernet_getByte()
              inc(cant_datos)
            next j
            dato[cant_datos] = 0
            dato_recibido=1
    end if
    if  (dato_recibido =1) then
```

```
        Configuacion_SPI_MMC()
        if(flag=0)then
          PORTD.0=1 flag=1
        else
          PORTD.0=0  flag=0
        end if
        Configuacion_SPI_Ethernet()
        dato_recibido=0
    end if
    Delay_ms(500)
    Spi_Ethernet_doPacket()
wend
    end.
```

ANEXO G

ARCHIVOS FUENTE DE LA

PROGRAMACION DEL MODULO

```
module eth_enc28j60_utils

    ' ****************************************************************
    ' VARIABLES
    include "eth_enc28j60_api"
    dim    txt        as string[11]
    dim    dato       as byte[512]
    dim    dato_recibido,cabecera as byte
    dim    cant_datos ,long_high, long_low ,j as  word

    ' ****************************************************************
    ' CONFIGURACION DE CLIENTE TCP
    sub function Spi_Ethernet_UserTCP(dim byref remoteHost as byte[4],
                        dim remotePort, localPort, reqLength as word) as word
    dim i as byte
                cant_datos=0
                result=0
                i=0
                for i = 0 to reqLength-1
                  dato[cant_datos] = Spi_Ethernet_getByte()
                  inc(cant_datos)
                next i
```

```
                dato[cant_datos] = 0

                dato_recibido=1

                ' return to the library with the number of bytes to transmit
end sub
' ************************************************************
' CONFIGURACION DE CLIENTE UDP
sub function Spi_Ethernet_UserUDP(dim byref remoteHost as byte[4],
                        dim remotePort, destPort, reqLength as word) as word
dim i as byte
      result = 0
      cant_datos=0
       result = result + reqLength
       for i = 0 to reqLength-1
       dato[cant_datos]= Spi_Ethernet_getByte()
       inc(cant_datos)
       next i
      dato[cant_datos] = 0
      dato_recibido=1
     '' end if
  end sub

end.
```

ANEXO H

EVALUACION ECONOMICA

a. ANÁLISIS DE COSTO

Hemos hecho un análisis detallado de los costos de nuestra tarjeta controladora y de los módulos utilizados, que al final nos permitirá dar una conclusión en cuanto al mismo.

b. 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.

i. ELEMENTOS DE TARJETA CONTROLADORA

ELEMENTOS			
COMPONETES	CANTIDAD	C/U	SUBTOTAL
Resistencia de 470 ohmios	2	0,05	
Capacitor de 470 microfaradios	2	$ 0,10	$ 0,20
Capacitor de 100 nanofaradios	2	$ 0,10	$ 0,20
Capacitor de 22 picofaradios	2	$ 0,10	$ 0,20
Diodo LED – RED	1	$ 0,10	$ 0,10
Diodo LED – GREEN	1	$ 0,10	$ 0,10
Microcontrolador 18f4520	1	$ 10,00	$ 10,00
Cristal	1	$ 2,00	$ 2,00
Regulador de voltaje	1	$ 0,50	$ 0,50
Zócalo de 40 pines	1	$ 0,25	$ 0,25
Jumper	1	$ 0,50	$ 0,50
Switch	1	$ 0,15	$ 0,15
		SUBTOTAL	$ 14,20
		IVA 12%	$ 1,70
		TOTAL	$ 15,90

Como observamos, se ha gastado un total de $15.90 en los elementos de la tarjeta controladora, valor que consideramos aceptable para la realización de este proyecto.

ii. MODULOS UTILIZADOS

MÓDULOS			
COMPONENTE	CANTIDAD	C/U	SUBTOTAL
Módulo MMC I&T	1	$ 20,00	$ 20,00
Módulo ENC28J60	1	$ 37,00	$ 37,00
		SUBTOTAL	$ 57,00
		IVA 12%	$ 6,84
		TOTAL	$ 63,84

En los módulos para el prototipo se gasto un total de $63.84, todos los módulos fueron conseguidos en el mercado local sin ninguna dificultad.

c. DISEÑO Y MONTAJE

Tarjeta Controladora

DESCRIPCION	CANTIDAD	SUBTOTAL
Diseño y elaboración de PCB	1	$ 45,00
Montaje y soldadura de elementos	1	$ 8,00
Desarrollo de software	1	$ 100
	SUBTOTAL	$1 53,00
	IVA 12%	$ 18,36
	TOTAL	$ 171,36

d. RESUMEN DE COSTOS

Elementos	$ 15,90
Módulos	$ 63,84
Tarjeta Controladora	$ 171,36
TOTAL	$ 251,10

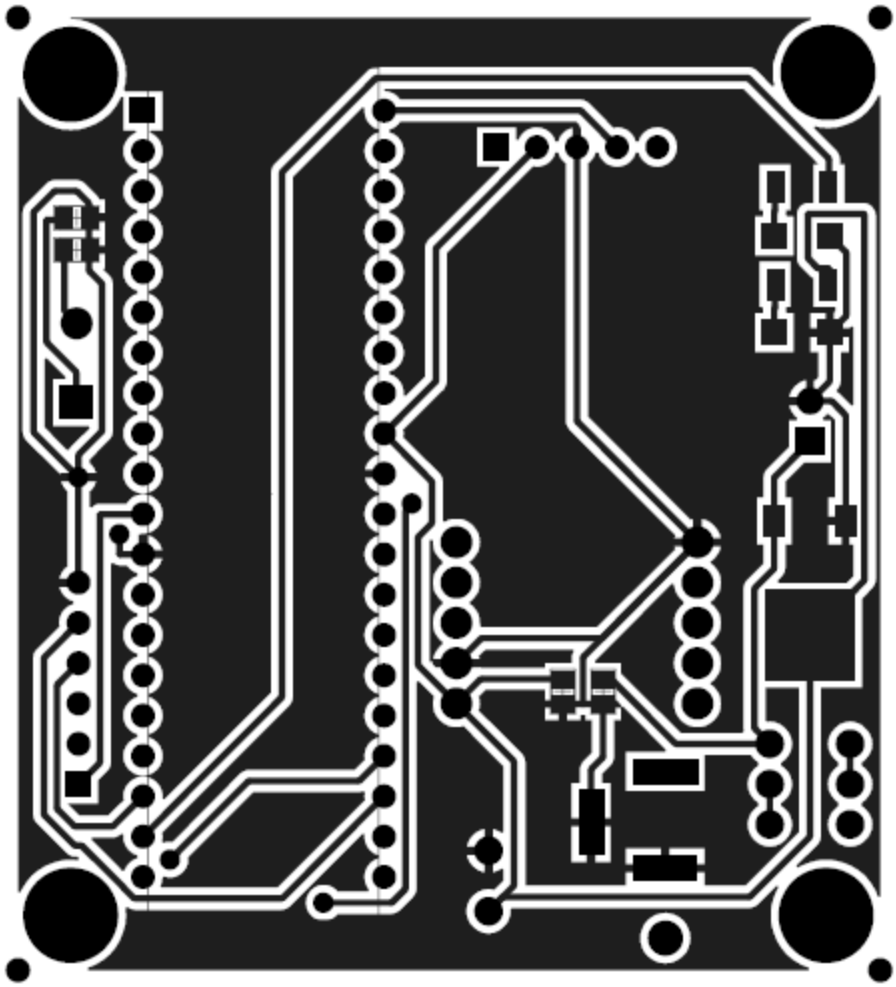
En esta sección, vemos el valor económico total utilizado en la elaboración de nuestro proyecto, un valor de $251.10; cabe recalcar, que este valor no incluye ciertos elementos que se adquirieron para realizar las pruebas necesarias para su buen funcionamiento.

ANEXO I

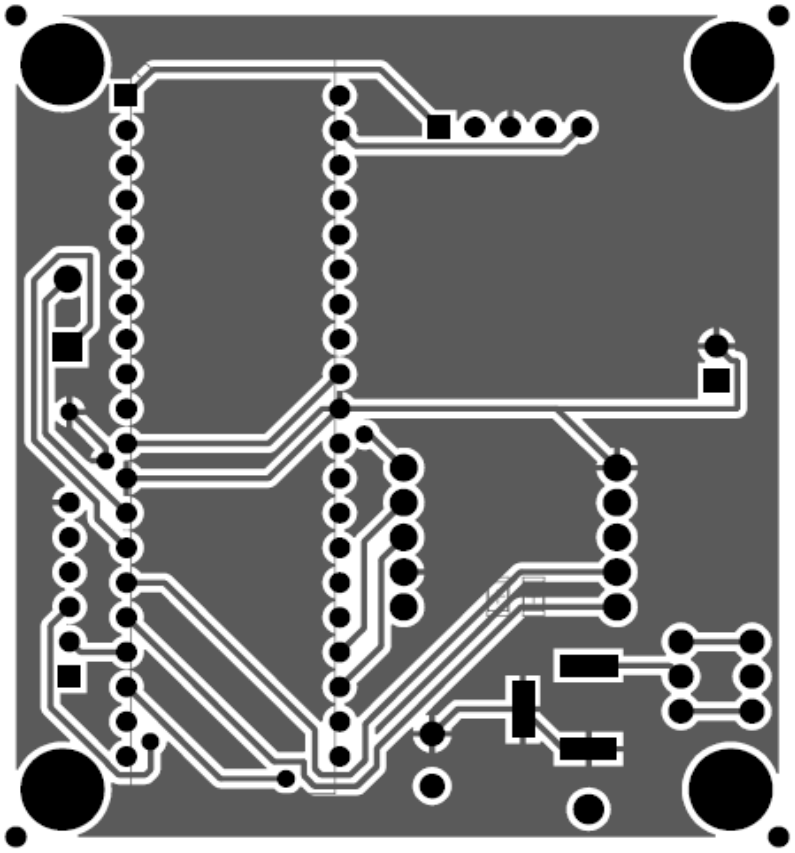
DISEÑO DE TARJETAS

DISEÑO PCB DE TARJETA CONTROLADORA

En el diseño PCB de la tarjeta controladora ser lo realizó en ARES, ésta es una aplicación de Proteus. Se lo desarrolló para que las pistas eléctricas de conductividad estén por las dos caras de la tarjeta. A continuación se muestra la cara frontal y la cara trasera de la tarjeta así como los módulos que la conforman.

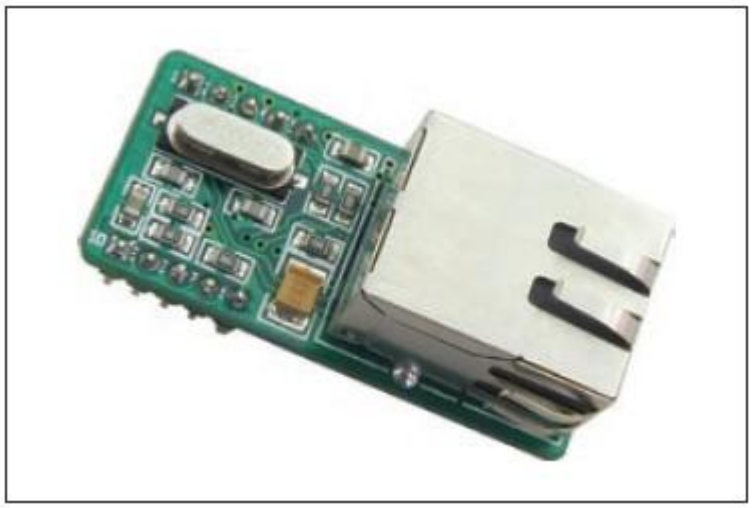


CARA FRONTAL

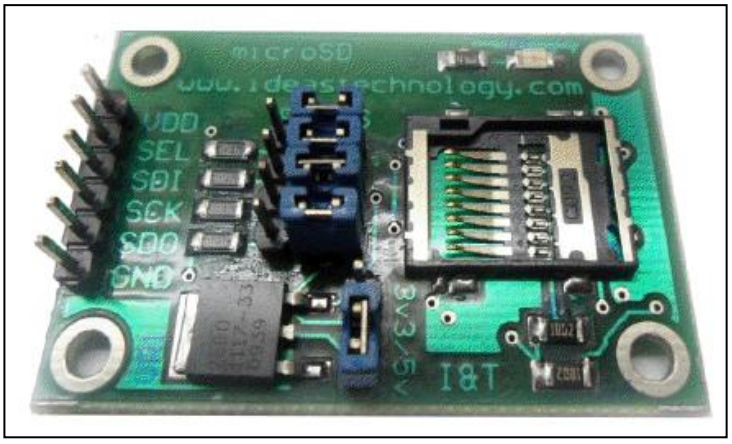


CARA TRASERA

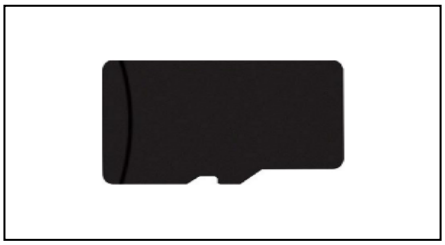
MODULO DE RED (ENC28J60)



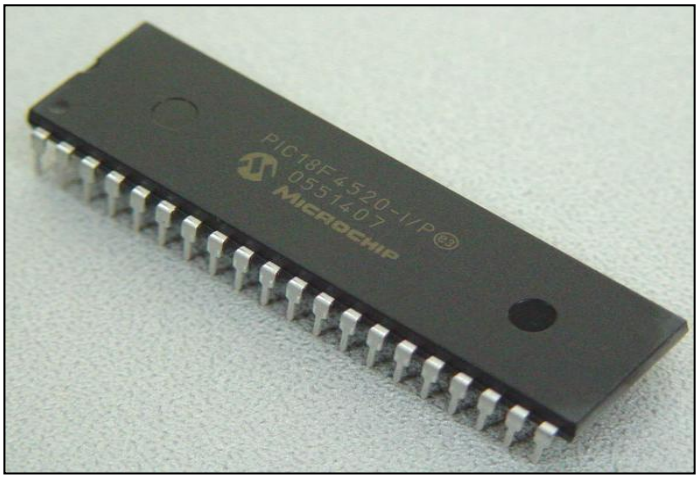
MODULO MICRO SD



MEMORIA MICRO SD

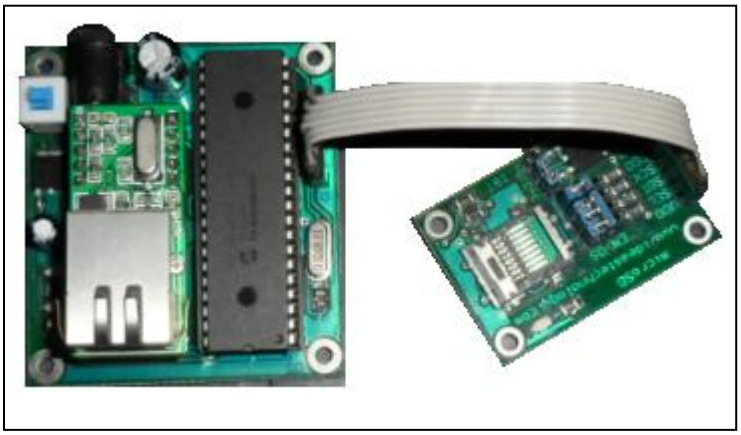


MICROCONTROLADOR (PIC 18F4520)



PROTOTIPO

En la imagen se muestra el prototipo ya desarrollado, como puede observar ahí encontramos el módulo de red y el de memoria que ya se encuentran integrados a la tarjeta controladora,



BIBLIOGRAFÍA Y REFERENCIAS

DE INTERNET

[1] Jimmy Wales y Larry Sanger, Enciclopedia Libre, http://es.wikipedia.org/wiki/, 23 Febrero 2011.

[2] Héctor Delgado Ureña Poirier y Juan Francisco Rodríguez Martín, Montaje y Configuración de una LAN: Ethernet, http://www.gobiernodecanarias.org/educacion/conocernos_mejor/paginas/ethernet.htm , 16 de Septiembre 2010.

[3] Sergio Vélez, Blog Modelo OSI', https://velezconde.wordpress.com/modelo-os, 3 de Agosto.

---

[1] Cisco System y Copyright; Manual CCNA Security CISCO versión 1.0;

2010