ESCUELA SUPERIOR POLITECNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

ESCUELA DE INGENIERÍA ELECTRÓNICA Y TECNOLOGÍA EN COMPUTACIÓN

"IMPLEMENTACIÓN DE UN SISTEMA DE GRABACIÓN REMOTA DE UN AVR MEDIANTE ETHERNET PARA CONTROLAR UN MOTOR CC"

TESIS DE GRADO

Previa la obtención del título de

INGENIERO EN ELECTRÓNICA Y COMPUTACIÓN

Presentado por:

PEDRO FERNANDO ESCUDERO VILLA

RIOBAMBA - ECUADOR

2009

Agradezco a mi familia por el apoyo incondicional que me ha brindado especialmente en los momentos más difíciles de mi carrera.

Dedico el presente trabajo de Tesis a mi Madre por respaldarme en todas mis decisiones para poder conseguir mis aspiraciones.

NOMBRE	FIRMA	FECHA
Dr. Romeo Rodríguez C. **DECANO FACULTAD INFORMATICA Y ELECTRONICA**	**……**	**……..**
Ing. Paúl Romero. **DIRECTOR ESCUELA DE INGENIERIA ELECTRONICA Y TECNOLOGIA EN COMPUTACION**	**……**	**……..**
Ing. Paúl Romero. **DIRECTOR DE TESIS**	**……**	**……..**
Ing. Jorge Yuquilema. **MIEMBRO DEL TRIBUNAL**	**……**	**……..**
Lic. Carlos Rodríguez **DIRECTOR CENTRO DE DOCUMENTACIÓN**	**……**	**……..**

NOTA DE LA TESIS **……**

“Yo, Pedro Fernando Escudero Villa, soy responsable de las ideas, doctrinas y resultados expuestos en esta tesis; y, el patrimonio intelectual de la Tesis de Grado pertenece a la ESCUELA SUPERIOR POLITECNICA DE CHIMBORAZO”.

_____

Pedro Fernando Escudero Villa

INDICE DE ABREVIATURAS

ANSI. - American National Standards Institute

API. - Application Programming Interface

ARP. – Address Resolution Protocol (Protocolo de resolución de direcciones)

AVR.- Advanced Virtual Risc

DIP. - Dual In-Line Package (Encapsulado Lineal Doble)

EEPROM. - Electrically Erasable Programmable Read-Only Memory (Memoria de solo lectura electricamente borrable)

GPL. – General Public Licence (Licencia Pública General)

ICMP. - Internet Control Message Protocol (Protocolo de control sobre internet)

IP. – Internet Protocol (Protocolo de internet)

ISP. - Internet Service Provider (Proveedor de servicios de internet)

LAN. – Local Area Network (Red de area local)

LQFP. - Low Quad Flat Pack (Encapsulado plano cuadruple bajo)

MAC.- Media Access Control (Control de Acceso al Medio)

NRWW.- No-Read-While-Write (No Lectura mientras se Escribe)

OSI. – Open System Interconnection (Interconexión de Sistemas Abiertos)

PC. – Personal Computer (Computadora Personal)

PCB. – Printed Circuit Board (Circuito impreso)

RAM. – Random Access Memory (Memoria de acceso aleatorio).

RWW.- Read-While-Write (Lectura mientras se Escribe)

SPI. – Serial Peripherical Interface (Interface de Periféricos Serial)

TCP. – Transmission Control Protocol (Protocolo para Control de Transmisión)

UDP. - User Datagram Protocol (Protocolo de Datagrama de Usuario)

VCC. – Voltage Continued Current **(**Voltaje de corriente continua)

VCD. – Voltaje de Corriente Directa

INDICE GENERAL

INDICE DE FIGURAS

INDICE DE TABLAS

INTRODUCCION

Antecedentes

Ethernet es la tecnología del Networking utilizada en muchas oficinas y hogares alrededor del mundo para comunicar y compartir recursos. Muchas redes Ethernet se conectan a un Router para conseguir acceso al Internet.

Un sistema de Ethernet es un dispositivo que permite fusionar dos ramas como son el networking y el control, permitiendo así generar varios sistemas derivados con una inteligencia computacional dedicada a realizar tareas simples, o un grupo de tareas relacionadas mediante un controlador.

La idea de unir dos mundos diferentes de la electrónica por medio de microprocesadores y sistemas de networking era casi imposible, ahora con el avance tecnológico a pasos agigantados en los últimos años de microprocesadores es posible unir estas áreas, lográndose una completa convivencia en la actualidad.

El Networking puede hacer a un sistema de control mucho mas avanzado sobre todo en lo que ha escalabilidad se refiere, reduciendo el tiempo de grabado en microcontroladores realizado en tiempos anteriores manualmente, desmontando y montando un nuevo microcontrolador en un sistema.

Uno de los factores que ha hecho posible la conexión de Ethernet a los sistemas de control es la disponibilidad de Chips Controladores que manejan en detalle las comunicaciones Ethernet.

Los microcontroladores AVR poseen un gran poder de procesamiento para los sistemas de control tornándose más rápidos y fáciles al manejar. Además los protocolos de comunicación a través de Ethernet e Internet se encuentran bien documentados y disponibles para el público sin ningún costo adicional.

Justificación Del Proyecto De Tesis

El diseño y la implementación de sistemas de Ethernet aplicados al uso de control se ha vuelto muy útil sobre todo, en donde el tiempo es un recurso muy preciado, por lo que es necesaria la implementación de un sistema que permita grabar un AVR a través de Ethernet sin tener que ser desmontado o remplazado manualmente permitiendo un óptimo trabajo de un motor de CC.

El implementar un sistema completamente dedicado a tal función se ha vuelto demasiado costoso si se usara una sola PC para tal función.

La idea de tener una sola computadora para este proceso seria un desperdicio de recursos, por lo que el sistema podría fácilmente conectarse a través de un sistema de red Ethernet existente.

La idea de este proyecto es dar una solución a ese problema, brindando escalabilidad a los sistemas de control y se pretende que sea el pilar fundamental para futuros desarrollos a través de Ethernet.

Objetivos

Objetivo General

Implementación de un sistema de grabación remota de un AVR mediante Ethernet para controlar un motor de CC.

Objetivos Específicos

- Investigar y Analizar el funcionamiento de los Sistemas de Red Ethernet.
- Investigar la mejor alternativa dentro de la familia de microcontroladores AVR que soporte grabación en circuito.
- Diseñar e implementar una interfaz web para la grabación del microcontrolador AVR con instrucciones que permitan al motor de corriente continua cambiar automáticamente de giro sea izquierda o derecha.
- Diseñar e implementar la interfaz de potencia desde el microcontrolador hacia el motor de corriente continua.

Hipótesis

Con La implementación de un sistema de grabación remota de un AVR mediante Ethernet, permitirá controlar los modos de trabajo de un motor de corriente continua, sea este paso a paso, cambio de giro automático o las dos a la vez.

Resultados

Una vez cumplido con los objetivos propuestos y haber realizado las investigaciones necesarias sobre el Control con Microcontroladores y Networking Embebido se ha comprobado la factibilidad de integrar estas tecnologías, pudiendo controlar a través de Ethernet un motor de corriente continua.

CAPITULO I

Control Automático y Networking Embebido

1.1 Control Automático

Un sistema de control es un ordenamiento de componentes físicos conectados de tal manera que el mismo pueda comandar, dirigir o regularse a sí mismo o a otro sistema. En el sentido más abstracto es posible considerar cada objeto físico como un sistema de control. Cada cosa altera su medio ambiente de alguna manera, activa o positivamente.

En la ingeniería y en la ciencia se restringe el significado de sistemas de control al aplicarlo a los sistemas cuya función principal es comandar, dirigir, regular dinámica o activamente.

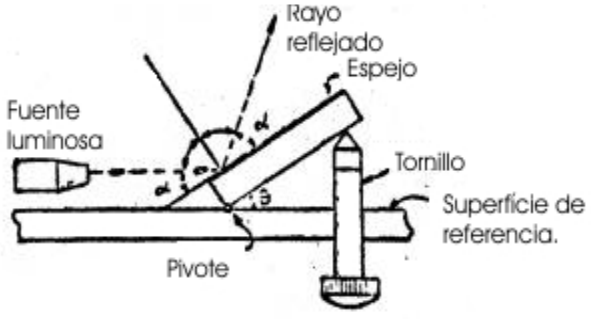


Figura I.1 Ejemplo de un sistema de Control

El sistema ilustrado en la Figura 1.1, que consiste en un espejo pivoteado en uno de sus extremos y que se puede mover hacia arriba o hacia abajo por medio de un tornillo en el otro extremo, se denomina propiamente un sistema de control. En ángulo de la luz reflejada se regula por medio del tornillo.

1.1.1 Beneficios

- Reducción en el tiempo de ejecución de rutinas
- Reducción de inversión de mano de obra en tareas no especializadas.
- Automatización de procesos repetitivos.
- Mejora de la calidad de los productos
- Economía de energía o potencia
- Economía de materiales y equipos industriales

1.1.2 Aplicaciones

- Control de Motores, Máquinas, herramientas, compresores y bombas, máquinas generadoras de energía eléctrica para el control de posición, velocidad y potencia.
- Industrias de procesamiento como la del petróleo, química, acero, energía y alimentación para el control de la temperatura, presión, caudal y variables similares.
- Manufactura de artículos como repuestos o partes de automóviles, heladeras y radio, para el control del ensamble, producción, tratamiento térmico y operaciones similares.
- Sistemas de transporte, como ferrocarriles, aviones, proyectiles y buques.

1.2 Networking Embebido

El término "embebido" también se lo conoce como "incrustado" o "embutido", es un sistema informático de uso específico construido dentro de un dispositivo mayor.

Está caracterizando que esos circuitos integrados son una parte integral del sistema en que se encuentran. Lo interesante de que un sistema sea "embebido" es que puede estar de tal forma

incrustado, puede quedar tan oculto a la vista humana, que la presencia de tales "chips" no resulte nada obvia a quien lo mira.

En un principio los sistemas embebidos eran aislados e incomunicados con el mundo exterior. En un ejemplo sencillo, el microcontrolador dentro de una maquina secadora estaba únicamente conectado a sus botones de control y motores, sin ofrecer ninguna forma de comunicación hacia el exterior a más de una simple visualización local si es que la tuviere. No todos los sistemas embebidos operan de esta manera. Algunos pueden comunicarse con otros muy próximos o cercanos a él, o en algunos casos a un dispositivo remoto, el cual puede estar en una computadora.

El Networking es un término común para métodos y técnicas usadas en el suministro de una comunicación confiable entre dispositivos separados geográficamente. El Networking requiere la implantación de estándares comunes y el uso de Hardware y Software compatible. Esto garantiza que la comunicación que se realiza entre maquinas o dispositivos puedan estar en arquitecturas y sistemas operativos completamente diferentes. La figura I.1 muestra dos aplicaciones donde el Networking Embebido puede ser usado

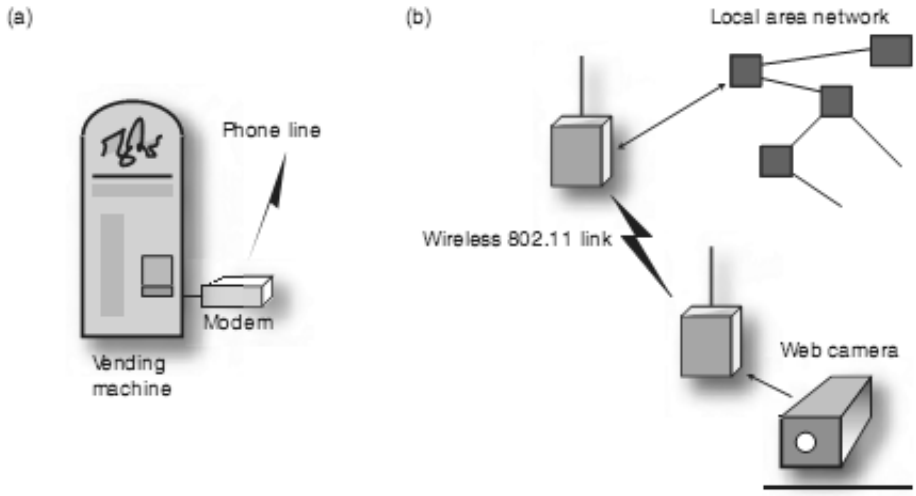


Figura I.2 Ejemplos de networking embebido

La figura I.2 (a) muestra una maquina expendedora de productos con un modem interno enlazado a una oficina central mediante una línea telefónica. El procesador dentro de la

maquina expendedora puede ser un microprocesador estándar con sus periféricos normales usados para el manejo de interruptores y motores que controlan a la maquina. La computadora en la oficina central puede iniciar llamadas a la maquina preguntando en intervalos definidos por el administrador del sistema su estado de funcionamiento. Por otra parte la maquina expendedora también puede informar a cerca de su estado de funcionamiento sin necesidad que la oficina central le pregunte. Este ejemplo sin embargo, acarrea recursos a la empresa pues se hace uso de la línea telefónica como medio de comunicación. Una forma de reducir costos es conectarlo al Internet a través de un ISP, ya que los costos de mantenimiento de una alternativa en red WAN son más baratos que un arrendamiento de línea dedicada.

La figura I.2 (b) muestra otro ejemplo aplicable del Networking embebido, donde una cámara IP usa una red local existente para enviar imágenes desde un parqueadero. Las imágenes son digitalizadas dentro de la cámara y enviadas en formato codificado a la oficina central, la cual decodifica las imágenes y las visualiza en la pantalla. Aquí se agrega una red Wireless al sistema, demostrando que un el Networking Embebido no se limita únicamente a tecnologías cableadas.

1.2.1 Beneficios

- Monitoreo y Control de ambientes industriales, domésticos y ofimáticos.
- Comunicación con otros dispositivos, ya que hay un número creciente de dispositivos que se conectan a Internet o a otras redes
- Disminución de costos de implementación, al usar infraestructura de red existente.
- Se pueden implementar servidores para propósitos específicos sin el uso de un computador completo (Web Servers Embebidos).

1.2.2 Aplicaciones

- Control de redes Industriales sobre Ethernet.
- Controles de refrigeración, calefacción, humedad, etc., en ambientes industriales.
- Estaciones Climatológicas

- Sistemas de seguridad y alarmas
- Sistemas de telefonía IP
- Monitoreo medico.

1.2.3 Protocolos

Los protocolos utilizados en Networking embebido son los mismos utilizados en el área de redes de computadoras dados por el RFC, se los puede implementar dependiendo de la plataforma de hardware que los soporten, siendo así, en las primeras versiones de sistemas de Ethernet embebido es posible implementar:

TCP/IP.- Protocolo para Control de Transmisión / Protocolo de Internet

ICMP.- Protocolo de control sobre internet

En actuales tecnologías de interfaces de Ethernet embebido como en los productos del fabricante Wiznet en su último lanzamiento del w5300 es posible implementar protocolos como:

HTTP.- Protocolo de transferencia de hipertexto

UDP.- Protocolo de Datagrama de Usuario

ARP.- Protocolo de resolución de direcciones.

DHCP.- Protocolo Configuración Dinámica de Anfitrión

PPPoE.- Protocolo Punto a Punto sobre Ethernet

IGMP.- Protocolo de administración de grupo de internet

CAPITULO II

Alternativas de tecnologías para el Control y Networking Embebido

2.1 Microcontroladores

Una de las principales herramientas que se utiliza hoy en la actualidad para el control, por lo que se debe escoger la serie, tecnología y fabricante a utilizar en la implementación del sistema. Existen diversos fabricantes y multitud de modelos que dificultan esta tarea. Según volumen de ventas y diversidad de modelos se puede establecer como principales a los siguientes fabricantes.

- Microchip Technology Corp.
- Atmel Corp.
- STMicroelectronics
- Motorola Semiconductors Corp.

2.1.1 Requerimientos de Implementación

Los fabricantes listados anteriormente ofrecen una amplia gama de productos de difícil selección a simple vista, por lo que se deben considerar algunos parámetros para la selección de la mejor alternativa según lo que requiera la implementación del sistema, como los siguientes:

- El microprocesador debe ser de 8 bits, pues no se necesita demasiada carga de procesamiento. El microprocesador debe tener la memoria de programa (Memoria Flash) en un rango entre 22 Kb y 32 Kb, teniendo en cuenta toda la programación de Firmware para el manejo de Networking será almacenada en la memoria flash del microprocesador.
- El encapsulado del microcontrolador de ser de preferencia de tipo DIP por la facilidad para realizar pruebas de Hardware sobre un Breadboard.
- El microcontrolador debe tener necesariamente soporte de protocolo SPI para la comunicación con el Controlador Ethernet.
- Memoria EEPROM para almacenamiento permanente de datos
- El número de Pines para entradas/salidas debe ser mayor a 32, ya que se necesitarán para incluir más módulos.

2.1.2 Alternativas de Microcontroladores

Por la facilidad de encontrar en el mercado local dispositivos de fabricantes como MICROCHIP y ATMEL se tomó estas alternativas de dispositivos para el estudio. Tomando en cuenta los requerimientos para la implementación se muestran los resultados en la Tabla (II.1)

Tabla II.1 Características de los microcontroladores que se ajustan a las necesidades del sistema.
Fuente: Paginas Web Oficiales de los Fabricantes

Alternativas	Arquitectura	Memoria FLASH	Memoria EEPROM	Pines entrada/salida	SPI	Encapsulado
PIC18F4520[(1)]	8 bits	32 Kbyte	256 byte	36	SI	DIP
ATmega32	8 bits	32 Kbyte	1 Kbyte	32	SI	DIP

[(1)] Se escogió el PIC18F4520 por tener el menor costo de entre los productos de MICROCHIP con similares características.

2.1.3 Elección de la mejor Alternativa en microcontroladores

Es necesario establecer métricas para escoger la mejor opción de microprocesador, estas son:

Precio: es la más importante, y fundamental para abaratar costos de producción.

Disponibilidad del componente en el país: permite saber que tan difícil es encontrarlo en el mediolocal.

Para realizar la elección de la mejor alternativa se consultó los precios y disponibilidad con APM Micro una empresa establecida en la ciudad de Quito, pues es la más especializada en el país en tecnología embebida. De las cuales se obtuvieron los datos de la Tabla II.2.

Tabla II.2 Alternativas de microprocesadores en el mercado.
Fuente: APM Micro, Quito-Ecuador

Alternativa	Precio	Disponibilidad
PIC18F4520	$ 18	Limitada
ATmega32	$ 11	Excelente

No es necesario realizar ningún análisis más para determinar que el **ATmega32** del fabricante ATMEL es la mejor opción en microcontroladores para implementar el sistema de control.

ATMEGA 32 de ATMEL

Esta familia está basada en una nueva arquitectura RISC que incorpora memoria Flash para el programa y memoria EEPROM para los datos. Además esta arquitectura fue diseñada para ser totalmente compatible con lenguaje C, permitiendo trabajar en alto nivel.

Tabla II.3 Características Básicas del microcontrolador ATMega32.
Fuente: Sitio web oficial de ATMEL(1) - Hoja de técnica del ATMega32

Características Básicas
Alto-performance, Baja-potencia, Microcontrolador AVR de 8-bits
Arquitectura avanzada RISC
131 Instrucciones
32 x 8 registros de propósito general
Sobre los 16 MIPS un rendimiento de procesamiento a 16 MHz
32K Bytes de auto programación en sistema, Auto programable, memoria Flash
1024 Bytes EEPROM

2K Byte SRAM interna Ciclos de escritura/borrado: 10,000 Flash/100,000 EEPROM Retención de Datos: 20 años 85℃/100 años a 25℃ Programación en sistema por programa de arranque en chip Correcta operación de lectura mientras se escribe en el microcontrolador Programado de Bloqueo por seguridad de Software Extensiva soporte de depuración en chip
Características de Periférico
4 Canales PWM 8 Canales de 10bits ADC 7 Canales diferentes únicamente en paquete TQFP Serial programable USART Maestro/Esclavo SPI Interfaz serial Watchdog Programable, unTimer con oscilador separado en chip
Características especiales del microcontrolador
6 modos de Sleep: Idle, ADC reducción de ruido, Power-save, Power-down, Standby and Standby extendido 32 líneas programables de I/O 40-pin PDIP, 44-lead TQFP, y 44-pad QFN/MLF
Voltajes de operación
4.5 - 5.5V para ATmega32 Grados de velocidad 0 - 16 MHz para ATmega32

2.2 Networking Embebido

2.2.1 Requerimientos de Implementación

Para el diseño y la implementación del networking embebido se requiere:

- Que el dispositivo a ser utilizado sea capaz de interactuar con una red LAN, usando como medio de comunicación cable par trenzado del tipo categoría 5E, por ser el más común utilizado en una red local.
- El dispositivo ha utilizar debe soportar protocolos como: ICMP, UDP, TCP, IP, ARP, ETHERNET, SPI.
- Debe ser compatible mínimo con redes 10BASET en adelante y transmisión half duplex.

La comunicación con la red se realiza utilizando alguna alternativa de dispositivo para networking embebido, actualmente existen muchas, sin embargo escoger la adecuada resulta una tarea muy difícil de realizar. De la misma manera que se hizo con los microcontroladores procedemos a buscar la mejor alternativa para la implementación.

2.2.2 Alternativas de networking embebido

De igual manera que los microcontroladores hay muchos fabricantes que ofrecen hoy por hoy muy diversas alternativas.

Controladores Ethernet

Un controlador Ethernet es un dispositivo capaz de enviar información desde un microcontrolador hacia una red LAN. Sin embargo, todo el procesamiento necesario para enviar datos a través de las capas del modelo OSI es realizado por el microcontrolador.

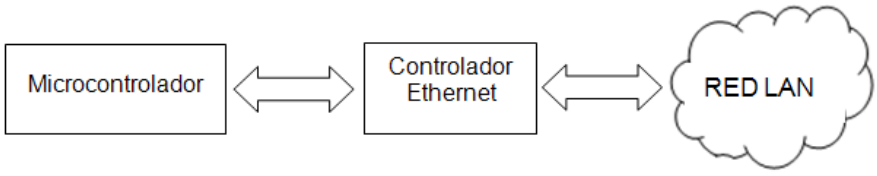


Figura II.3 Networking embebido (controladores Ethernet)

Este tipo de soluciones ofrece flexibilidad al permitir escoger la familia de microcontroladores más conveniente para realizar cualquier proyecto en networking embebido. Es la opción más adecuada en el caso de querer fabricar el sistema al más bajo costo, siendo esta alternativa la

que se utilizó para el diseño del sistema. Entre las alternativas de controladores Ethernet se encuentran:

1) CS8900A

Figura II.4 CS8900A

Descripción: Uno de los primeros en su tipo, y de los más difundidos en su época, está basada en tecnología ISA

Fabricante: CIRRUS LOGIC

Características Principales:

- 4 Kb de memoria buffer para transmisión/recepción de paquetes.
- Interfaz 10 Base T
- Interfaz para la comunicación con redes 10 Base 5, 10 Base 2, 10 Base F
- La comunicación con el CPU se hace únicamente con interfaz paralela

2) ENC28J60

Figura II.5 ENC28J60

Descripción: Muy versátil al ofrecer la posibilidad de comunicarse mediante el protocolo SPI con el CPU del sistema. Maneja a la perfección las primeras dos capas del modelo OSI.

Fabricante: MICROCHIP TECHNOLOGY CORP

Características Principales:

- Compatible con redes ethernet 10/100/1000 Base T.
- MAC integrada con interfaz 10 Base T.
- Soporta comunicación Full Duplex y Half Duplex.
- Comunicación SPI a velocidades de hasta 20 Mhz
- 8 Kb de memoria buffer para transmisión/recepción de paquetes.
- Filtrado de tramas mediante varias opciones de configuración

3) WIZNET 5300

Figura II.6 WIZNET 5300

Descripción: Es lo más nuevo en controladores Ethernet, pues ofrece toda la implementación del Modelo TCP/IP embebida en su circuitería interna, esto significa un ahorro de código muy significativo.

Fabricante: WIZNET

Características Principales:

- Soporta protocolos embebidos mediante hardware TCP, UDP, IP, ARP, Ethernet, etc.
- Soporta hasta 8 comunicaciones por sockets a la vez.
- Interfaz de Red de hasta 50 Mbps.
- 128 Kb de memoria buffer para transmisión/recepción de paquetes.
- Comunicación con el CPU mediante interfaz paralela en Modo Directo e Indirecto.
- Comunicación SPI con el CPU.

2.2.3 Selección de la mejor alternativa

Para realizar la selección de la mejor alternativa en controladores Ethernet se tomaron en cuenta las los siguientes aspectos:

Precio: Un factor muy importante pues se necesita es abaratar costos al máximo posible.

Disponibilidad: Para establecer si es posible conseguir el componente en el medio local.

Facilidad de programación: Muy importante pues se necesita saber que tan difícil es manejarlo al momento del diseño del firmware.

Facilidad de Implementación: en el aspecto hardware, para determinar qué tan costoso es implementarlo en el sistema, pues se debe tomar en cuenta si se necesitan elementos o equipos adicionales a los disponibles.

Ahora es muy importante realizar una priorización de las métricas pues algunas son más importantes que otras. Sobre una calificación de 10 puntos se asignaron diferentes a cada una de ellas (Tabla II.4).

Tabla II.4 Priorización de las métricas para el estudio de la mejor alternativa

Métricas	Calificación según su importancia
Precio	1
Disponibilidad	1
Facilidad de programación	3
Facilidad de Implementación	5

Como se observa en la Tabla II.1 la facilidad de fabricación es el aspecto más importante pues abaratar costos de producción es el objetivo principal, esto es tomado muy en cuenta debido al tipo de encapsulado que el fabricante del componente distribuye. Se requiere primordialmente que el encapsulado sea tipo DIP (Dual In-Line Package), pues este permite realizar pruebas y modelado de fabricación mediante un breadboard, además de facilitar la fabricación del PCB (Printed Circuit Board) para la presentación final del sistema. Esto no ocurre con el encapsulado tipo LQFP para los que se necesitan adaptadores especiales para conectarlo sobre un breadboard, y ni hablar de la fabricación del PCB pues para ensamblar el dispositivo se necesita equipo especial de soldadura. El segundo aspecto a considerar es la facilidad de

programación ya que el diseño del firmware del sistema depende mucho del tipo de plataforma en la que se trabaja.

2.2.4 Estudio de la mejor alternativa

Calificaciones individuales de las alternativas

Se necesita los valores de las métricas de cada una de las alternativas antes mencionadas. El precio fueron tomadas páginas oficiales de los fabricantes, en el país los precios son casi los mismos solo varían en unos pocos dólares (Tabla II.5).

Tabla II.5 Precio de cada una de las alternativas
Fuente: Paginas web oficiales de los fabricantes

Alternativa	Precio
CS8900A	$4,50
ENC28J60	$3,70
WIZNET 5300	$5,90

Las métricas restantes se calificaron sobre una puntuación de 3 (Tabla II.6)

Tabla II.6 Significancia para la calificación de métricas

Significancia	Puntuación
MALO	0
REGULAR	1
BUENO	2
EXCELENTE	3

Las calificaciones a cada una de las métricas para el estudio de la mejor alternativa se realizaron de acuerdo a la información obtenida de cada una de las páginas oficiales de los fabricantes (Tabla II.7).

Tabla II.7 Calificaciones de cada una de las métricas

Alternativa	Disponibilidad del componente en el país[2]	Facilidad de programación[3]	Facilidad de Fabricación del Sistema[2]
CS8900A	0	0	0
ENC28J60	1	1	3
WIZNET 5300	1	3	1

Elección de la mejor alternativa

Para escoger la mejor alternativa se utilizó el método de la matriz de puntos.

Tabla II.8 Calificación según el precio

[2] Fuente: APM Micro, Quito-Ecuador
[3] Fuente: Pagina Web Oficial del Fabricante

Factor / Alternativa	Precio				
	1	2	3	Sumatoria	Calificación Individual
CS8900A	0	1	-	1	0.33/1
ENC28J60	1	-	1	2	0.66/1
WIZNET 5300	-	0	0	0	0
Total	-	-	-	**3**	1

Tabla II.9 Calificación según la disponibilidad del componente en el país

Factor / Alternativa	Disponibilidad del componente en el país				
	1	2	3	Sumatoria	Calificación Individual
CS8900A	0	0	-	0	0
ENC28J60	1	-	1	2	0.5/1
WIZNET 5300	-	1	1	2	0.5/1
Total	-	-	-	**4**	1

Tabla II.10 Calificación según la facilidad de programación

Factor / Alternativa	Facilidad de programación				
	1	2	3	Sumatoria	Calificación Individual
CS8900A	0	0	-	0	0
ENC28J60	1	-	0	1	0.33/1
WIZNET 5300	-	1	1	2	0.66/1
Total	-	-	-	**4**	1

Tabla II.11 Calificación según la Facilidad de Fabricación del Sistema

Factor / Alternativa	Facilidad de Fabricación del Sistema				
	1	2	3	Sumatoria	Calificación Individual
CS8900A	0	0	-	0	0
ENC28J60	1	-	1	2	0.66/1
WIZNET 5300	-	1	0	1	0.33/1
Total	-	-	-	**4**	1

Tomando en cuenta la Tabla II.4 de priorización se necesita ponderar las calificaciones a esos valores de la tabla.

Tabla II.12 Valores de ponderación de las métricas

Métrica	Valor de Ponderación
Precio	0.1
Disponibilidad	0.1
Facilidad de programación	0.3
Facilidad de Fabricación del Sistema	0.5
Total	**1**

Ahora para el resultado definitivo se necesita ponderar a cada alternativa con su respectivo valor de calificación individual para cada una de las métricas.

$$CS8900A = 0.1 \times 0.33 = 0.033$$

$$ENC28J60 = (0.66 \times 0.1) + (0.5 \times 0.1) + (0.33 \times 0.3) + (0.66 \times 0.5) = \mathbf{0.5549}$$

$$WIZNET\ 5300 = (0.5\ \times 0.1) + (0.66\ \times 0.3) + (0.33\ \times 0.5)\ = 0.4328$$

Se observa el puntaje más alto en el ENC28J60, siendo así la mejor opción entre controladores Ethernet el **ENC28J60.**

2.3 Software de desarrollo para control y Ethernet embebido

Existen en el mercado una diversidad de software orientado al Control Automático y aun más para lo que es Ethernet embebido por lo que se ha tomado en cuenta algunos aspectos para una selección apropiada como son:

- Un lenguaje de programación mas utilizado para diseño de interfaces, manejo de puertos, programación de sockets
- Tipo de licencia que posea, o en su preferencia que sea bajo licencia pública GPL.
- Portabilidad, reutilización de código, y guías de ayuda.
- Existencia de grupos y comunidades en Internet, dedicadas a la elaboración y mantenimiento de proyectos para la plataforma ATMEL, que usan ANSI C como lenguaje de programación estándar.

Analizando los aspectos a considerar para la selección de software se ha escogido **AVRStudio** para el diseño de firmware, **wxWidgets y Code::Blocks** para el diseño del panel de control CPanel.

AVRStudio, es entorno de desarrollo integrado y simulación el cual es distribuido de manera gratuita por el fabricante ATMEL, este utiliza lenguaje ANSI C para el desarrollo, y las librerías que utiliza son enteramente de LIBC y AVR GCC, ambos son proyectos de desarrollo orientado al Código Libre. Al utilizar estas herramientas se logró abaratar aun más los costos de fabricación pues son libres y distribuidas bajo la Licencia Publica GPL

WxWidgets son bibliotecas multiplataforma y libres para el desarrollo de interfaces gráficas programadas en lenguaje C++. Están publicadas bajo una licencia LGPL, similar a la GPL con la excepción de que el código binario producido por el usuario a partir de ellas, puede ser propietario, permitiendo desarrollar aplicaciones empresariales sin coste, JULIAN(13).

Las wxWidgets proporcionan una interfaz gráfica basada en las bibliotecas ya existentes en el sistema (nativas), con lo que se integran de forma óptima y resultan muy portables entre distintos sistemas operativos. Están disponibles para Windows, MacOS, GTK+, Motif, OpenVMS y OS/2. También pueden ser utilizadas desde otros lenguajes de programación, aparte del C++: Java, Javascript, Perl, Python, Smalltalk, Ruby . La última versión es la II.8.9 de septiembre de 2008.

Code::Blocks, es una herramienta de entorno de desarrollo integrado (IDE en inglés) para el desarrollo de programas en lenguaje C++. Está basado en la plataforma de interfaces gráficas WxWidgets, lo que le permite correr libremente en diversos sistemas operativos, y es de licencia GPL.

Debido a que Dev-C++ es un IDE para programar en C y C++ y está creado en Delphi, surgió la idea y necesidad de crear un IDE hecho en los lenguajes adecuados: C y C++. Con esta motivación se creó el IDE **Code::Blocks**.

Por lo tanto se ha demostrado que la mejor alternativa en networking embebido es utilizar el controlador Ethernet **ENC28J60**, el cual se comunicará con el microprocesador **ATmega32**, controlado por una interfaz software diseñada en **CODE::BLOCKS**, consiguiendo así un diseño al más bajo costo del sistema de control al no usar herramientas de desarrollo propietarias, sino herramientas de desarrollo libres y con licencia pública GPL.

CAPITULO III

Diseño e Implementación del Sistema

3.1 Requerimientos de Hardware

En esta parte la comunicación con la interfaz Ethernet fue el principal objetivo del sistema de networking embebido implementado, sin embargo a partir de eso se vio en la necesidad de crear un control de mando para accionar el motor de CC una vez cargado el firmware Bootloader en el microcontrolador.

En el momento del diseño se consideraron conceptos importantes como:

Parámetros de configuración local: esto involucra la dirección MAC del sistema con el que se le identificará en la red LAN a nivel físico, la dirección IP (fija) con la que se le distingue a nivel lógico.

Parámetros de configuración remota: esto involucra únicamente la dirección IP del host, una dirección ya conocida, con el que identifica a donde se envían los datos y de donde se recibirá las ordenes a nivel lógico.

Ahora para cumplir con los objetivos del sistema, se necesita un adecuado diseño de hardware, en otras palabras se diseñó una plataforma sobre la cual se podrá realizar el diseño del firmware para que el sistema funcione correctamente. Es necesario definir un diagrama de bloques del Hardware requerido para diseñar el sistema (Figura III.7).

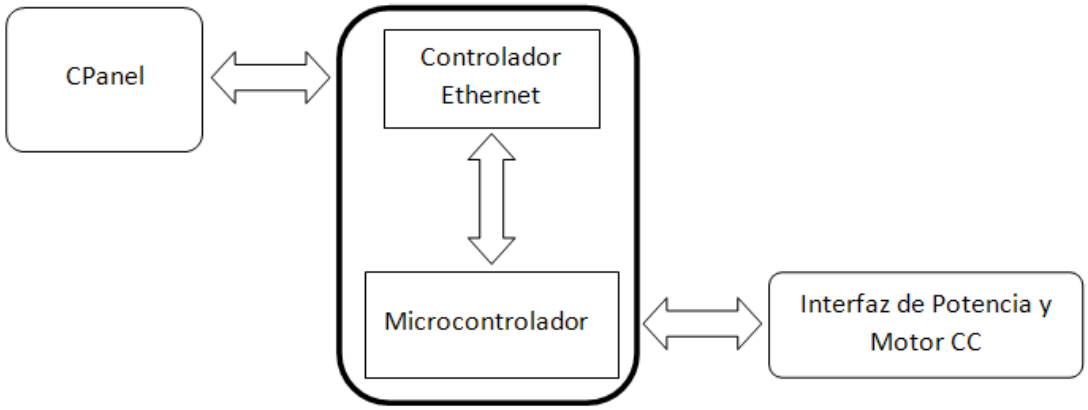


Figura III.7 Diagrama de bloques del sistema

3.2 Diseño de Hardware

3.2.1 Interfaz con la red

La Interfaz con la red se refiere a las conexiones para comunicación entre el microprocesador y el controlador Ethernet ENC28J60 mediante un bus serial en el que se utiliza el protocolo de comunicación SPI.

3.2.2 Protocolo SPI

Spi es un bus de tres líneas, sobre el cual se transmiten paquetes de información de 8 bits. Cada una de estas tres líneas porta la información entre los diferentes dispositivos conectados al bus. Cada dispositivo conectado al bus puede actuar como transmisor y receptor al mismo tiempo, por lo que este tipo de comunicación serial es full duplex. Dos de estas líneas trasfieren los datos (una en cada dirección) y la tercera línea es la del reloj.

Algunos dispositivos solo pueden ser transmisores y otros solo receptores, generalmente un dispositivo que tramite datos también puede recibir.

Un ejemplo podría ser una memoria EEPROM, el cual es un dispositivo que puede transmitir y recibir información.

Los dispositivos conectados al bus son definidos como maestros y esclavos. Un maestro es aquel que inicia la transferencia de información sobre el bus y genera las señales de reloj y control.

Un esclavo es un dispositivo controlado por el maestro. Cada esclavo es controlado sobre el bus a través de una línea selectora llamada **Chip Select o Select Slave**, por lo tanto es esclavo es activado solo cuando esta línea es seleccionada. Generalmente una línea de selección es dedicada para cada esclavo. En un tiempo determinado T1, solo podrá existir un maestro sobre el bus. Cualquier dispositivo esclavo que no este seleccionado, debe deshabilitarse (ponerlo en alta impedancia) a través de la línea selectora **(chip select).**

El bus SPI emplea un simple registro de desplazamiento para transmitir la información.

Especificaciones del Bus

Todas las líneas del bus transmiten la información sobre una sola dirección.

La señal sobre la línea de reloj (SCLK) es generada por el maestro y sincroniza la transferencia de datos.

La línea MOSI (Master Out Slave In) transporta los datos del maestro hacia el esclavo.

La línea MISO (Master In Slave Out) transporta los datos del esclavo hacia el maestro.

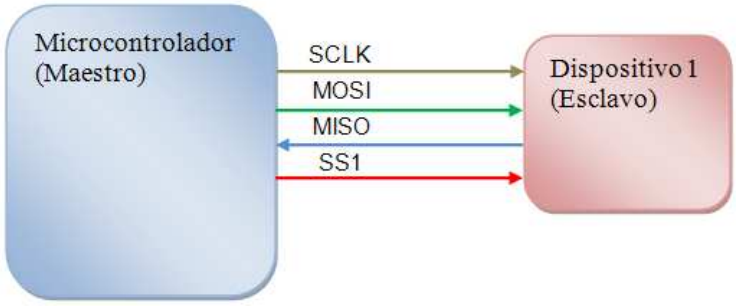


Figura III.8 Diagrama de bloques Protocolo SPI

3.2.3 Diseño e Implementación de la interfaz con la red

Las conexiones con el controlador Ethernet ENC28J60 deben estar como especifica el fabricante, MICROCHIP.

Se utiliza los pines designados para SPI del microcontrolador en el puerto B, que funcionan con una alimentación de 5V de CC, Como detalla la Figura III.9

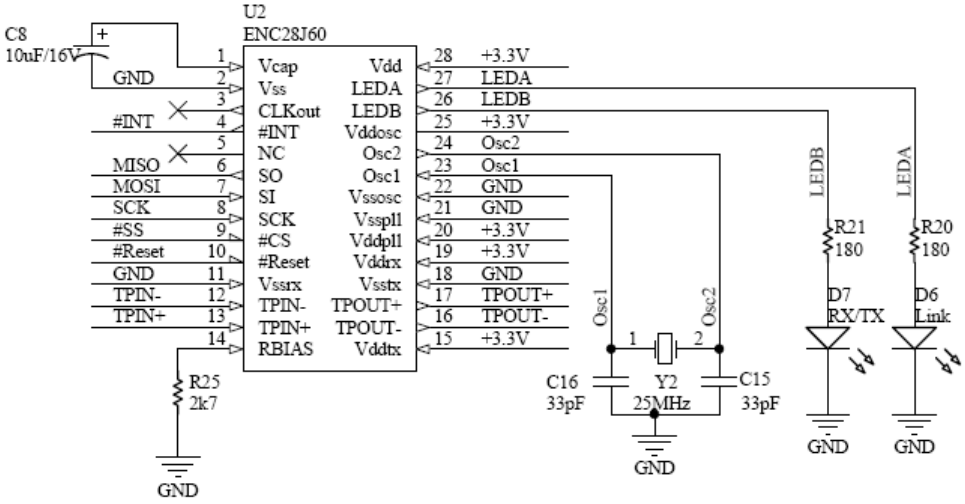


Figura III.9 Diagrama de hardware para el ENC28J60

3.2.4 Implementación del Microcontrolador ATMega32

La parte central del hardware la compone el microcontrolador y este bloque es implementado en su totalidad con el firmware del Sistema, el circuito físico del microprocesador fue tomado de la hoja de datos del ATMega32.

Las pines utilizados del microcontrolador son los de alimentación, cristal externo: VCC, GND, XTAL y RST de alimentación y reset respectivamente y configuración de protocolo SPI para la comunicación con el modulo ENC28J60, que son los pines señalados como MOSI, MISO, SCK, INT, RESET.

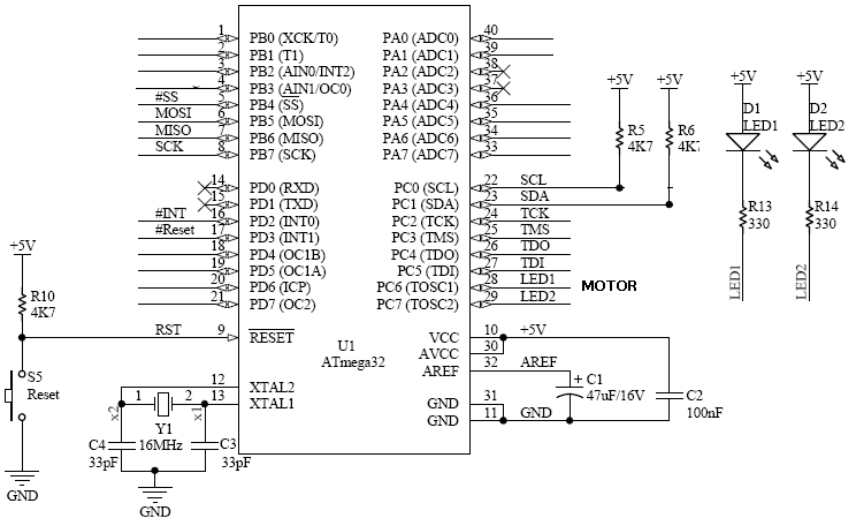


Figura III.10 Diagrama de hardware para el ATMega32

3.2.5 Diseño e Implementación de la etapa de Potencia y Motor

La etapa de adaptación o etapa de Potencia esta conformada por un puente H (L293D), un amplificador operacional (TL072CP), y un inversor (74LS04).

El objetivo de esta etapa es adaptar las salidas de bajo potencial del microcontrolador para alimentar el Motor de CC.

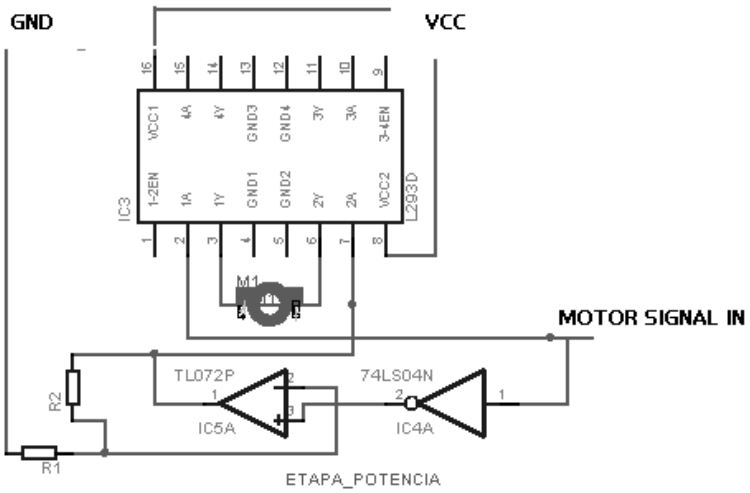


Figura III.11 Etapa de Potencia y Motor CC

3.2.6 Implementación Completa del Sistema

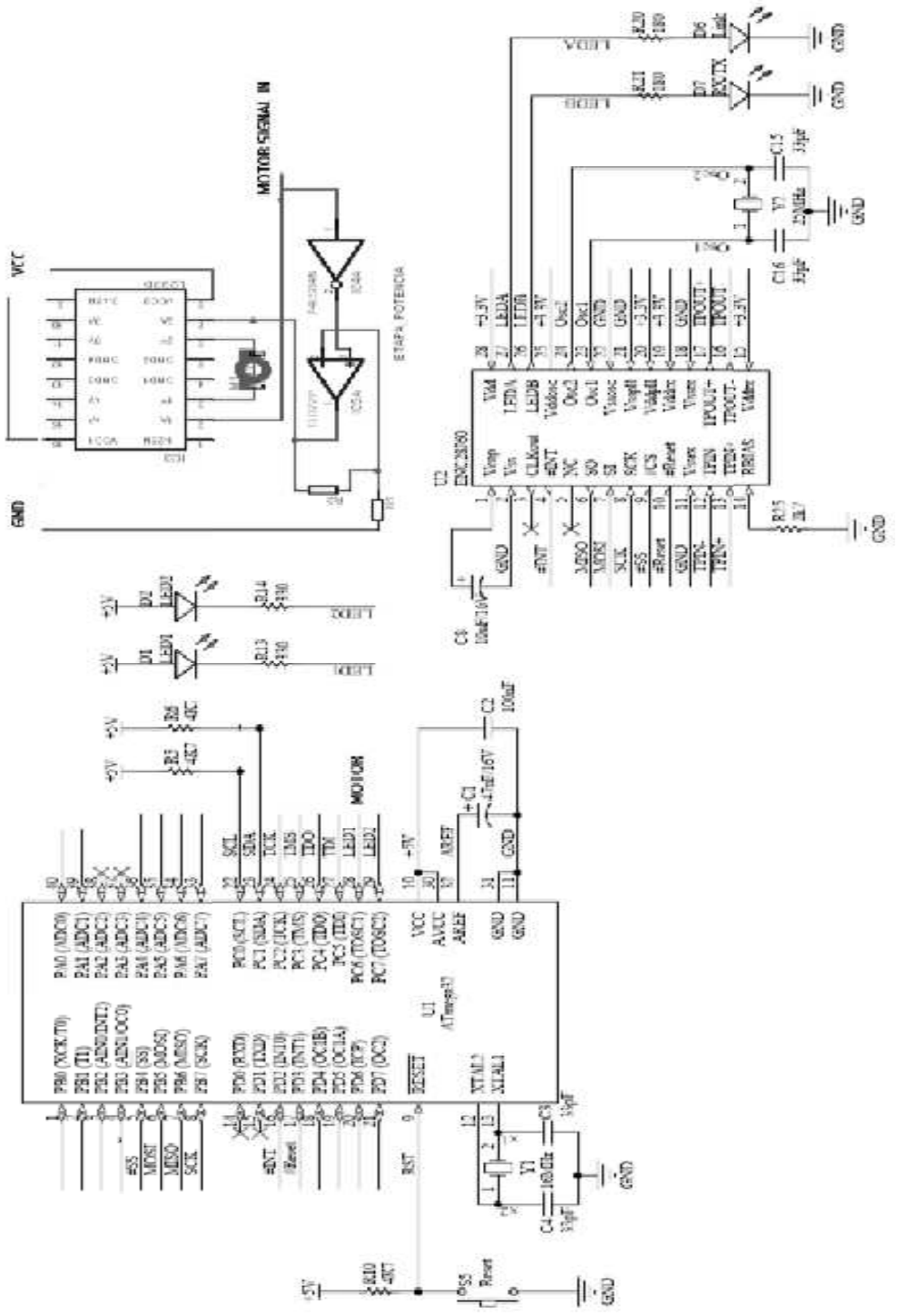


Figura III.12 Esquema del Sistema Completo

3.3 Diseño de Firmware

3.3.1 Consideraciones

El Objetivo del Diseño de firmware es permitir controlar la parte de hardware y establecer la comunicación con la interfaz Ethernet para salir a la red LAN además que deberá permitir una comunicación de transmisión y recepción de datos desde el host remoto hacia el microcontrolador y la etapa de potencia y motor que se lo llamara en esta sección como **módulo AVRM**.

3.3.2 Estructura de Firmware

El firmware se lo ha estructurado en dos partes:

- *Firmware Orientado al Manejo del Hardware*: funciona en un nivel físico, encargado del correcto funcionamiento de la parte física sobre la cual todo el proceso de networking se ejecutará, así como el control del módulo AVR.
- *Firmware Orientado al Manejo de Protocolos de Comunicación*: funciona en un nivel lógico, pues se e encarga de todo el proceso de networking para lograr una comunicación con un host remoto.

La estructura en si del firmware se muestra en la (Figura .III.13).

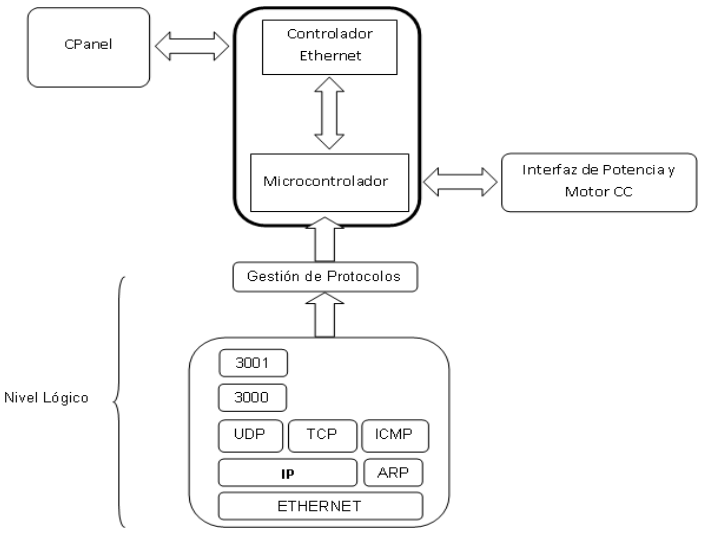


Figura III.13 Esquema del Firmware

3.3.3 Diagrama de Flujo de Firmware

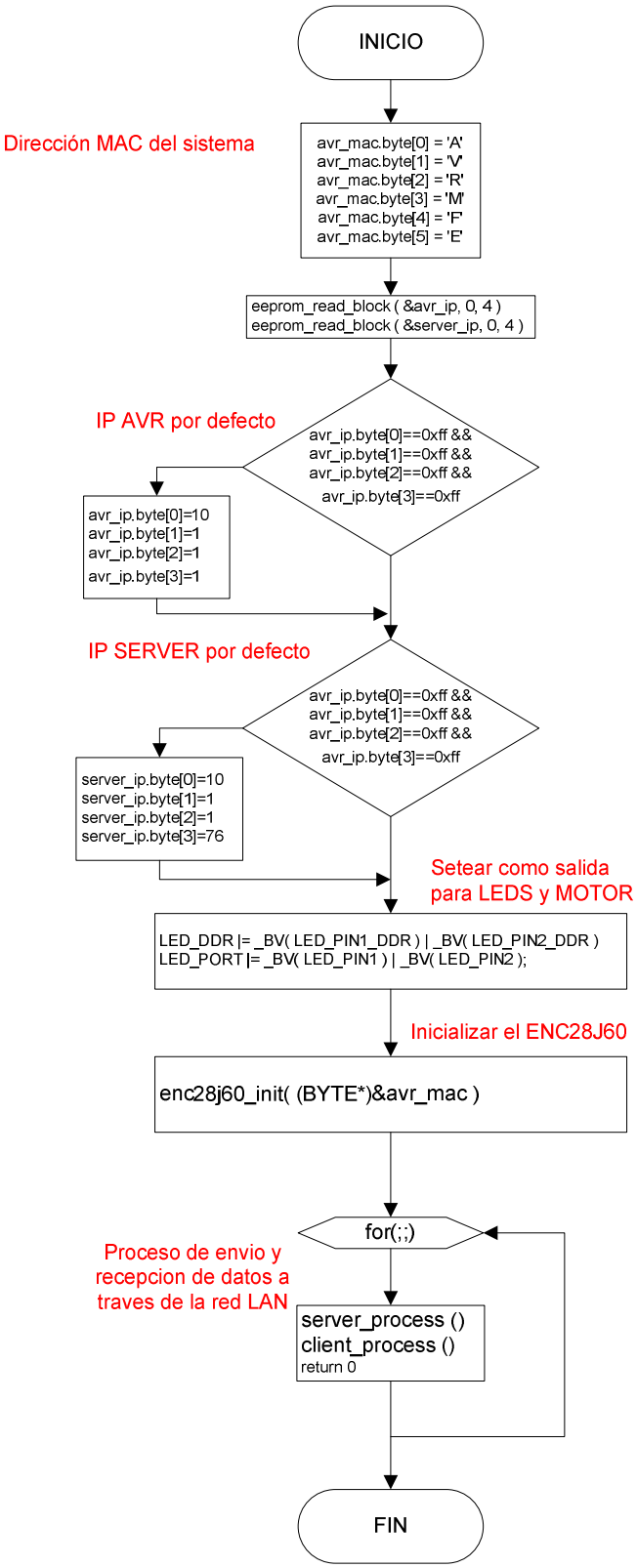


Figura III.14 Flujograma de Firmware

3.3.4 BOOTLOADER

El **Bootloader** (cargador de arranque) es un programa sencillo diseñado exclusivamente para preparar todo lo que necesita para que el firmware principal pueda funcionar.

EL Bootloader es cargado en el microcontrolador utilizando un programador básico fuera de circuito, una vez ahí y ya instalado en el circuito, permite descargar nuevo firmware al AVR usando únicamente el puerto serie, sin ningún hardware adicional en un momento de reset del sistema.

3.3.5 Organización de Memoria

La memoria flash esta dividida dentro de dos secciones, una la sección de Aplicación y otra la de Cargador “Loader”.

La sección de aplicación contiene el código de inicio de la aplicación, mientras la sección de Boot Loader contiene el código para el actual Self-Programming.

La instrucción SPM puede únicamente ser ejecutada desde la sección de Boot Loader. (Nota: El Boot Loader puede también ser utilizado por código de una Aplicación ordinaria).

La memoria flash es dividida dentro de paginas que contienen 32, 64, o 128 palabras cada una. Por ejemplo: un dispositivo de 8Kb de Flash y medida de pagina de 32 palabras (64bytes) tendrá por lo tanto un total de 128 paginas. La organización de memoria se muestra en la Figura III.15

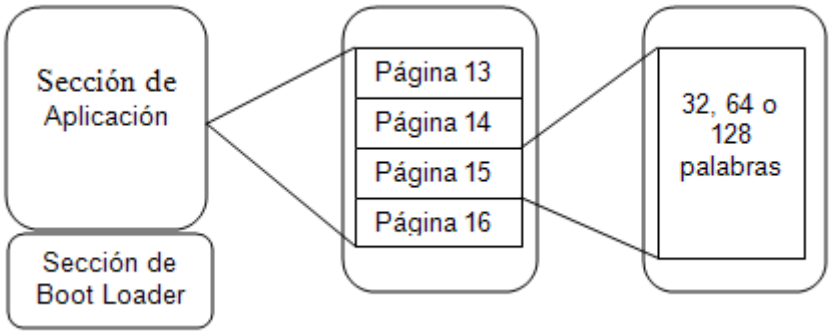


Figura III.15 Organización de memoria de un microcontrolador para Bootloader

La medida de la sección de Boot Loader puede ser seleccionada usando fuse Bits BOOTSZx

Los fuses selecciona una de las cuatro medidas predefinidas. El fuse bit BOOTZx puede ser cambiado usando una programación serial o paralela, MICROCHIP (2)

Si el Boot Loader una vez implementado puede llamar directamente desde el código de aplicación, usando jumps o interrupciones, o también por medio de programación del fuse bit BOOTRST, cuando el fuse BOOTRST es programado, el CPU iniciara la ejecución en la sección de Boot Loader en un Reset.

3.3.6 Capacidad de Lectura mientras se Escribe

Además de la división de seleccionable entre la Aplicación y el Boot Loader la flash is también divididas dentro de dos secciones de medidas arregladas

La primera sección de la de Lectura mientras se Escribe (RWW).

La segunda es la sección de No Lectura mientras se Escribe (NRWW).

La medida de la sección NRWW siempre es igual a la mas grande de la sección seleccionada para Boot Loader, Así la sección de Boot Loader ocupara la parte de la sección NRWW, Como ilustra en la Figura III.16

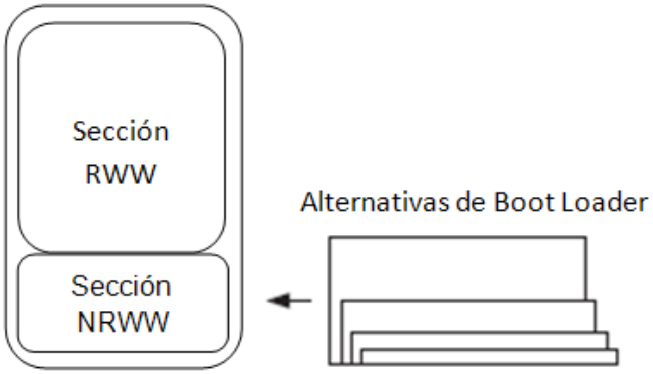


Figura III.16 Sección de RWW y NRWW

La diferencia entre secciones es que NRWW es accesible mientras se actualiza la sección de RWW. No es posible acceder a la sección RWW mientras se esta actualizando, cuando la sección NRWW esta actualizada ejemplo: actualizando el Boot Loader por si mismo, el CPU esta detenido mientras dura la operación. En otras palabras la sección no lee mientras esta escribiendo.

3.3.7 Estructura del diseño de Bootloader

El Bootloader diseñado para el sistema consta de dos partes:

Control de protocolos: Permite establecer comunicación con el host remoto desde donde se enviaran las actualizaciones de firmware.

Rutinas de descarga de firmware: Permite realizar todas las rutinas necesarias para poder descarga el firmware vía serial y programarlo en el AVR

Esta estructura se puede observar más claramente en la Figura III.17:

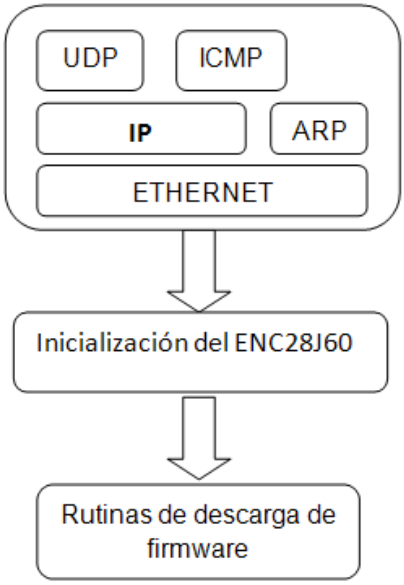


Figura III.17 Esquema del Firmware

3.3.8 Diagrama de Flujo de Bootloader

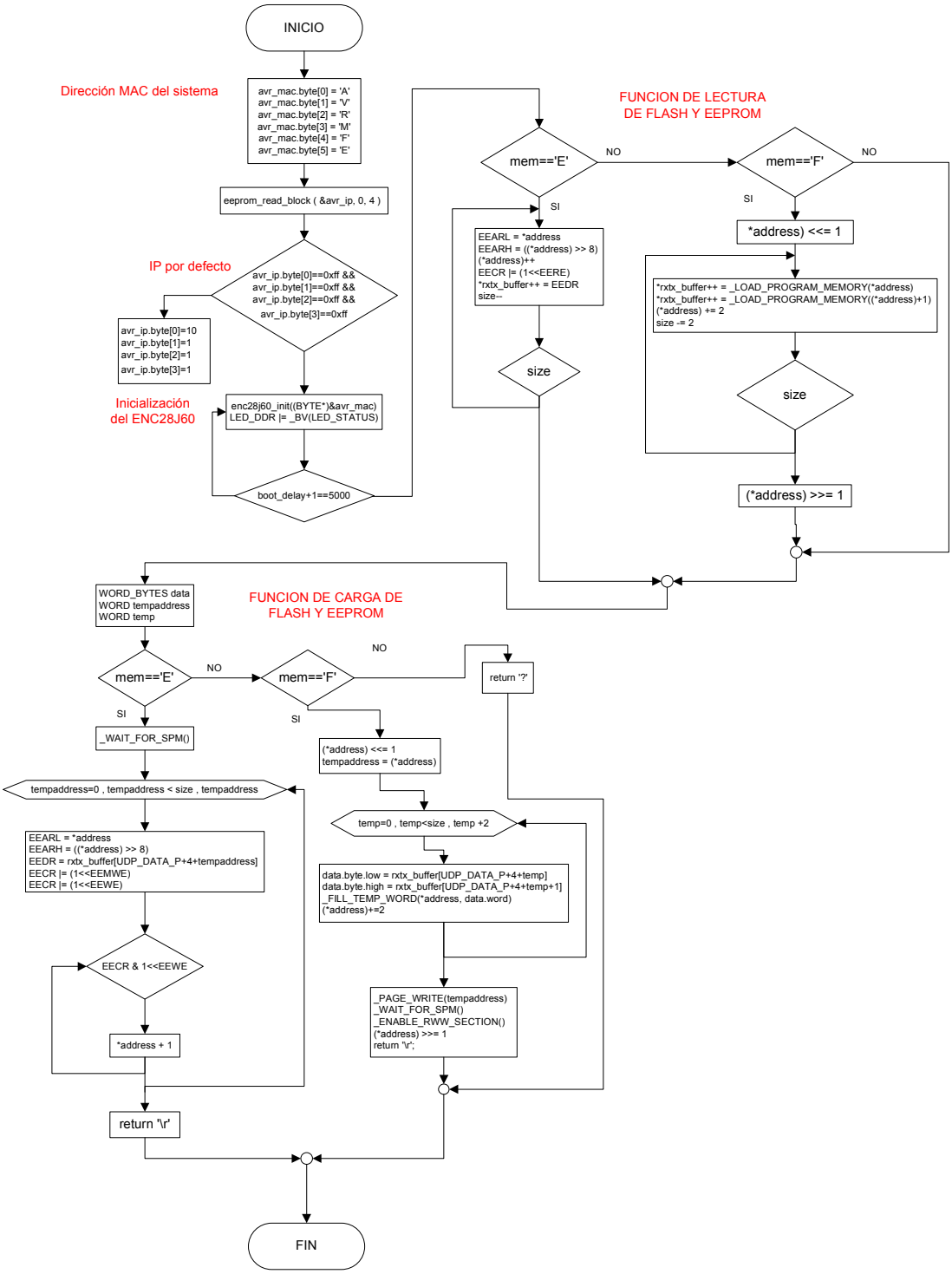


Figura III.18 Diagrama de Flujo de Bootloader

3.4 Diseño del Panel de control CPanel

3.4.1 Consideraciones

El Objetivo del Diseño del panel de control es tener una interfaz gráfica desde donde se enviarán y recibirán los datos de control y comunicación con el módulo AVRM, por lo que el diseño esta basado en la facilidad y simplicidad de manejo para el usuario de cada una de las herramientas.

3.4.2 Estructura del CPanel

La programación del CPanel esta estructurada en base a funciones realizadas en C++ , cada una de estas funciones orientadas a realizar una rutina en especial, por ejemplo:

void AVRMDialog::OnConnect.- Mediante esta función el CPanel se conecta al módulo AVRM mediante el envío y confirmación del comando ***"GA\r\n"*** explicados mas adelante en Capitulo III.7.2.

void AVRMDialog::OnOpenHex y void AVRMDialog::OnOpenEep.- Permiten localizar un archivo .hex y .eep respectivamente en el disco duro y prepararlo para enviarlo como actualización de firmware hacia el módulo AVRM.

void AVRMDialog::SendSL.- Permite controlar el estado del motor de corriente continua instalado en el módulo AVRM por medio del comando **"SL"** explicado mas adelante en Capitulo III.7.2.

void AVRMDialog::OnSocketEvent.- Permite recibir los datos del estado de configuraciones del AVRM, como: Direcciones IP del AVRM y Servidor y Estados de puertos.

void AVRMDialog::OnSetIp.- Permite Editar una nueva dirección IP en el módulo AVRM.

void AVRMDialog::OnUpgrade.-Permite enviar los archivos .hex y .eep hacia el módulo AVRM para ser programado en el microcontrolador.

La interfaz gráfica en donde se incorporan cada uno de estos comandos se ilustra en el Figura III.19

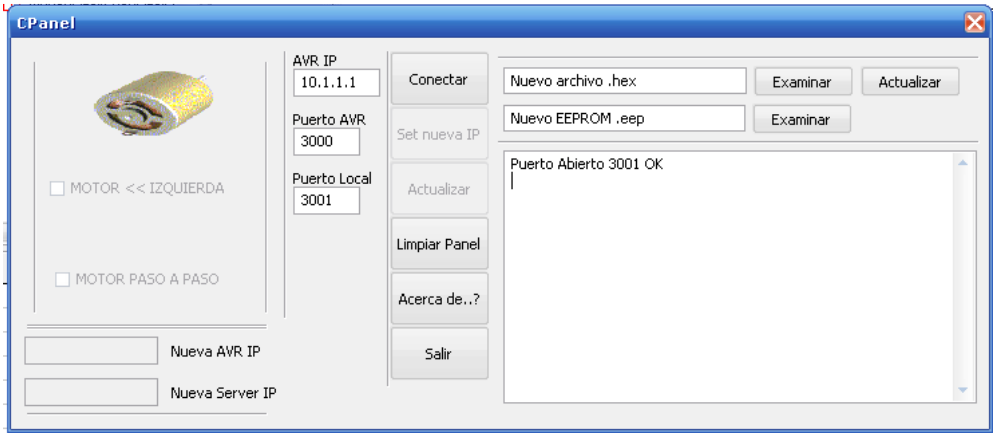


Figura III.19 Interfaz Gráfica del CPanel

3.4.3 Diagrama de Flujo de Firmware

La función inicial del CPanel comienza con la lectura de Dirección IP del host, apertura de puertos par envío y recepción de datos.

Los Diagramas de Flujo del resto de programación del CPanel se muestran en el (Anexo 1).

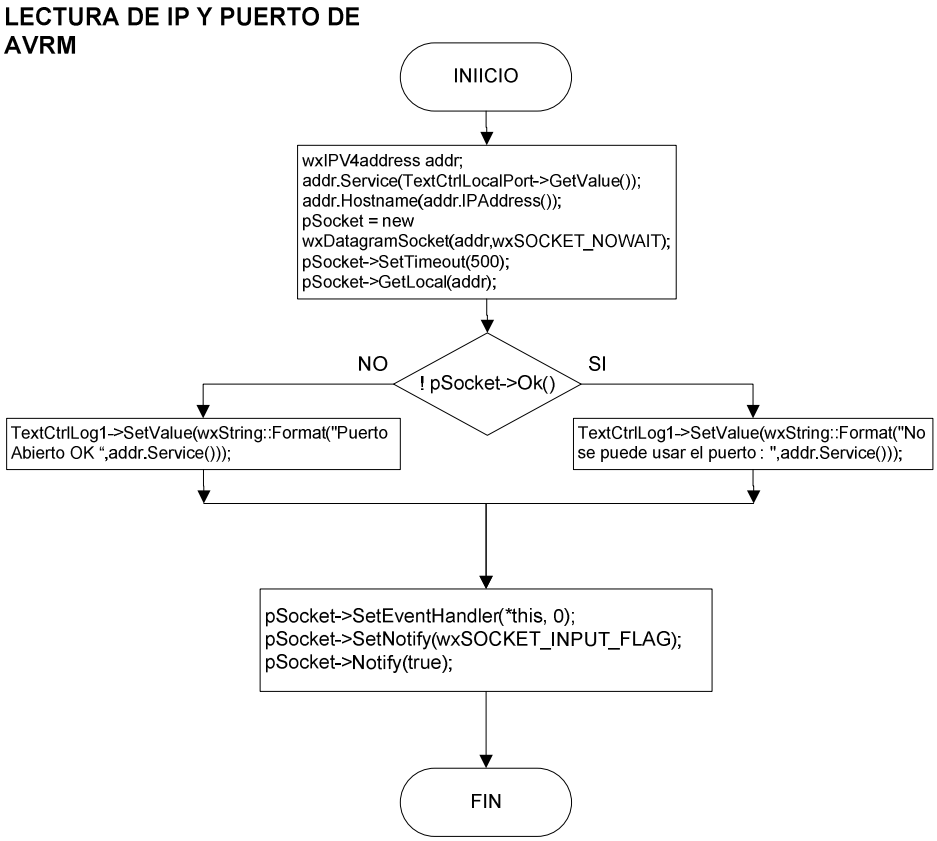


Figura III.20 Diagrama de flujo de función de Inicio del CPanel

3.5 Manejo del controlador Ethernet

El networking del sistema es realizado a través del circuito integrado ENC28J60 ya que este tiene embebido las rutinas para la gestión de las primeras dos capas del modelo OSI. Sin embargo por si solo no permite la comunicación por la red, MICROCHIP (2). Es necesario entonces seguir una secuencia de pasos para realizar una correcta inicialización, envío/recepción de tramas y una correcta administración del buffer de memoria. Las rutinas para el manejo de este dispositivo definidas en el archivo **enc28j60.c (**Anexo 1).

3.5.1 Inicialización del ENC28J60

Primero se inicializa el modulo SPI del ATmega32 esto implica que se debe asignar los valores adecuados a los registros *SPCR* y *SPSR* asignándoles 0x50 y 0x01 respectivamente, esto configura el Modulo SPI para trabajar en modo MASTER, con una frecuencia de reloj de 8 Mhz (Valor de la frecuencia en la línea SCK), una polarización de reloj en 0 y una fase de reloj en 0 (Modo 00), MICROCHIP (2), Estos parámetros son muy importantes pues solo de esta manera se obtiene una óptima comunicación entre el ATmega32 y el ENC28J60. Luego se envía a un comando Soft Reset (el formato de comandos se describe más adelante) para reiniciar el ENC28J60. Ahora un paso muy importante es la definición del tamaño del Buffer (espacio de almacenamiento) para transmisión y recepción de datos hacia y desde la red respectivamente.

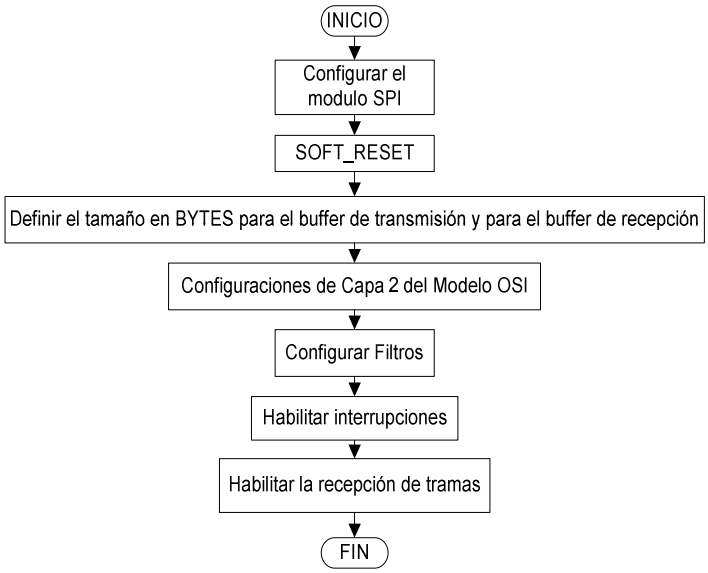


Figura III.21 Secuencia de inicialización del ENC28J60

Debido a que el tamaño del buffer total es de 8Kb (8192 bits), MICROCHIP (2), se definió un espacio desde el bit 0 hasta el 6691 para el buffer de recepción de datos y desde el bit 6662 hasta el 8192 (1500 bits, máximo tamaño de la trama Ethernet) para el buffer de transmisión. La secuencia de inicialización sigue con las configuraciones de la gestión MAC (Capa 2 del Modelo OSI), estos parámetros se indican en la Tabla III.13 en el orden de programación.

Tabla III.13 Pasos para las configuraciones de capa 2 en el ENC28J606

Orden	Operación
1	Habilitación de recepción de tramas en el dispositivo
2	Habilitar el relleno a 60 bytes del PAYLOAD y el cálculo del CRC de la trama si los datos a enviarse son menores al mínimo permisible en una red Ethernet (60 bytes)
3	Habilitar la retransmisión de tramas en caso de que el medio este ocupado
4	Configurar la ventana de colisión para evitar el problema de las colisiones tardías
5	Asignar los valores que sugiere el fabricante para configurar el espaciado entre tramas y el tiempo del mismo en 9.6µs
6	Asignar el máximo tamaño de la trama en 1518 bytes como establece el protocolo Ethernet
7	Asignar la dirección MAC que se le asignó a al dispositivo (“SANTYQ” = 53:41:4e:54:59:51)
8	Deshabilitar el retorno de retransmisión de tramas (Loopback)
9	Configurar los PINES del ENC28J60 conectados a los diodos LED para visualizar una el estado del enlace y un destello del LED a 140ms de velocidad en cada transacción transmisión/recepción

Lo siguiente es configurar los filtros de entrada para que solo reciba tramas BROADCAST cuyo protocolo de aplicación sea ARP y también UNICAST cuya dirección de destino coincida con la dirección MAC local, esto se hizo mediante un cálculo según lo estipulado por el fabricante, MICROCHIP (2), a continuación se detalla el proceso.

Para el Filtro de BROADCAST ARP se usó el Filtro Por Coincidencia De Patrón embebido en el ENC28J60 para el cual se tomo en cuenta la estructura de la trama detallada en la Figura III.22, pues si llegan peticiones ARP solo este tipo de tramas pasarán por el filtro. El filtro necesita que se asignen los valores adecuados en los registros EPMM0…EPMM7, EPMOH, EPMOL, EPMCSL, EPMCSH del ENC28J60.

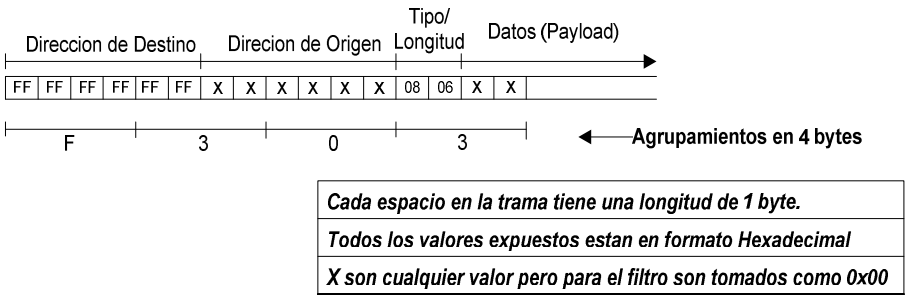


Figura III.22 Trama ethernet de una petición BROADCAST-ARP

Lo primero es definir un *Patrón* para que sea comparado con las tramas entrantes y como se ve en la Figura 4.22 el valor es 0x3F y 0x30 pues los valores necesitan ser agrupados en bytes para ser asignados solo a los registros EPMM0 y EPMM1 ya que son suficientes para expresar el patrón deseado. Ahora se necesita definir un desplazamiento desde el inicio de la trama hasta el inicio del patrón, en este caso el desplazamiento es cero pues el patrón se define desde el inicio de la trama, por lo tanto los valores de EPMOH y EPMOL serán 0x00. Por último se necesita asignar el valor del checksum (CRC) del grupo conformado por todos los bytes de interés dentro de la trama, en otras palabras la dirección de Destino (FF: FF: FF: FF: FF: FF) y el campo Tipo/Longitud (0x0806). El cálculo del checksum da como resultado 0xf7 y 0xf9 cuyos valores deben ser asignados a los registros EPMCSH y EPMCSL respectivamente.

Después de haber configurado los filtros se habilitó las interrupciones del ENC28J60 con esto al ocurrir cualquier evento se activará la bandera correspondiente y el microprocesador podrá realizar las rutinas pertinentes para dicho evento, por ejemplo el recibimiento de un nuevo paquete desde la Red. Por último en la rutina de inicialización se habilita al dispositivo para que empiece a recibir tramas de la red.

3.5.2 Comunicación entre el microprocesador y el ENC28J60

Para el envío/recepción de datos sobre el ENC28J60 el cual está en modo SLAVE se lo realizó mediante el envío de comandos cuyo formato es el descrito en la Figura III.23, de esta comunicación se encargan las funciones ***enc28j60ReadOp*** y ***enc28j60WriteOp*** (Anexo 1)**,** en estas funciones se utilizan llamadas a las MACROS:

- CSACTIVE: macro para empezar la transmisión SPI al poner un “0” en la línea CS.
- CSPASSIVE: Macro para terminar la comunicación SPI al poner un “1” en la línea CS.
- waitspi(): macro que realiza una demora hasta que la transacción SPI haya finalizado.

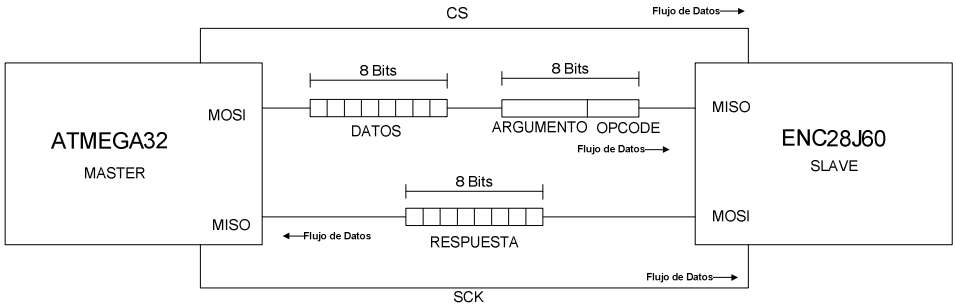


Figura III.23 Formato de envío de comandos hacia el ENC28J60

Tomando en cuenta que el ENC28J60 posee registros de control, repartidos en cuatro bancos diferentes la función ***enc28j60SetBank*** se encarga de conmutar a través de esos bancos, y por ende las funciones ***enc28j60Write*** y ***enc28j60Read*** hacen uso de las funciones anteriormente descritas para realizar un control total de lectura/escritura sobre el ENC28j60, en otras palabras se utilizó estas dos funciones para realizar toda la gestión de comunicación SPI. El ENC28J60 posee adicionalmente registros de control físicos encargados de toda la gestión de capa física del modelo OSI y para realizar operaciones de lectura/escritura de estos registros se utilizaron las funciones ***enc28j60_read_phyreg*** y ***enc28j60PhyWrite.***

3.5.3 Lectura/Escritura de paquetes sobre Ethernet

Lo último que falta para completar una completa comunicación con la interfaz Ethernet son las rutinas necesarias para leer y escribir sobre la Red misma. Esto se realizó con la implementación de las funciones ***enc28j60_packet_send*** para el envío una trama de datos y ***enc28j60_packet_receive*** para leer el contenido del buffer de recepción, se siguió el proceso exacto descrito por el fabricante, MICROCHIP (2).

3.6 Gestión de Protocolos

La comunicación a través de la red se realiza mediante protocolos de comunicación dados por el RFC, (Figura III.24) muestra los protocolos implementados en el sistema.

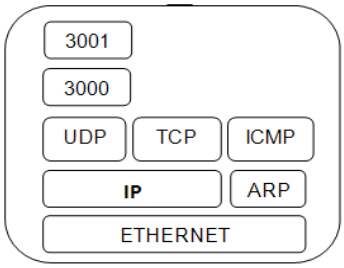


Figura III.24 Protocolos implementados en el sistema

3.6.1 Buffer de transmisión y recepción

Considerando que la trama Ethernet tiene una longitud de 1518 bytes, se integra al sistema una variable global rxtx_buffer, que tendrá el contenido exacto de la trama que se desea recibir o transmitir desde y hacia la red. Por lo que esta definida como un arreglo unidimensional tipo BYTE de 1518 posiciones mostrada en la (Figura III.25). Esta variable es global en todo el sistema, debido a que se la usa en diferentes funciones dentro del sistema.

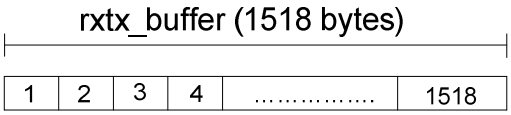


Figura III.25 Variable buffer rxtx_buffer para Tx/Rx

3.7 Protocolo UDP

Para la implementación de este protocolo se creó la función ***udp_generate_header*** (Anexo1) usada en la implementación tanto de firmware como de Bootloader. Esta función se encarga de generar la cabecera UDP de la trama.

El protocolo de datagrama de usuario da únicamente un mínimo servicio de transporte para una liberación de datagramas no garantizado permitiendo generar aplicaciones que no requieren un nivel de servicio TCP o que aspiren usar servicios de comunicaciones (ej. multicast o broadcast), no habilitado desde TCP.

Permite el envío de datagramas a través de la red sin que se haya establecido previamente una conexión por lo que se implementó especialmente para la descarga de firmware al microcontrolador.

Dentro de lo que es UDP se utiliza el puerto 3000 y 3001 (protocolos de información y alertas).

3.7.1 Esquema General

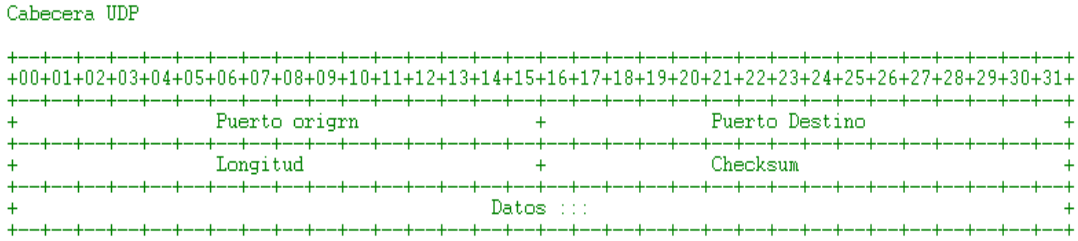


Figura III.26 Arquitectura de cabecera UDP

3.7.2 Estructuración

La trama UDP esta compuesta por un vector de datos que es enviado desde el CPanel o interfaz desarrollada en CODE::BLOCKS desde donde se controlara la recepción y envío de datos.

La estructura del vector de envío es la siguiente:

"GA" = comando que inicia el vector de envío hacia el módulo AVRM desde el CPanel

El formato de respuesta es: OKLLAAA.AAA.AAA.AAA;SSS.SSS.SSS.SSS;\r\n

LL = LED1 (Motor), LED2

ADC0 is ADC0 value

AAA.AAA.AAA.AAA = Es la dirección IP del AVR

SSS.SSS.SSS.SSS = Es la dirección IP del Servidor

';' = Es el fin de dirección IP

\r\n = Es el fin de comando

Por ejemplo: GA1010.1.1.1;10.1.1.76\r\n = LED1 encendido y Motor en movimiento, LED2 apagado, AVRM IP 10.1.1.1; SERVER IP 10.1.1.76

Todo esto es recibido mediante la función ***udp_receive*** y respondido por las funciones ***udp_puts_data*** y ***udp_puts_data_p*** (ANEXO1)

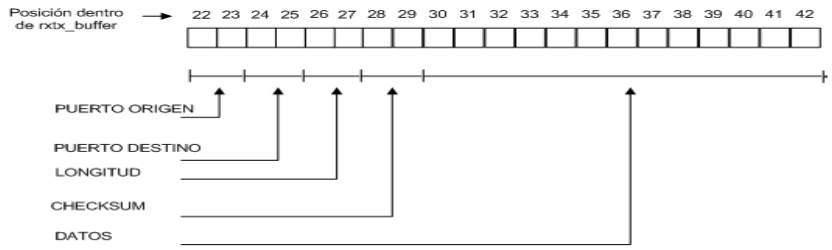


Figura III.27 Arquitectura de cabecera UDP en el rxtx_buffer

3.7.3 Puertos 3000, 3001

Los puertos 3000 y 3001 son implementados con el fin de establecer comunicación entre el módulo AVRM y el CPanel.

EL puerto 3000 es asignado al módulo AVRM y el puerto 3001 es asignado al host remoto como puerto local.

3.8 Protocolo Ethernet

Para la implementación de este protocolo se creo la función ***eth_generate_header*** *(*Anexo 1). Esta función genera la cabecera Ethernet de la trama.

Dentro de la trama la función ***eth_generate_header*** necesita el numero de protocolo de capa superior que se va ha utilizar y la dirección MAC de destino hacia la que se le enviará la trama, todas estas operaciones se realizan sobre la variable rxtx_buffer, generándose la cabecera completa de la trama lista para que las demás funciones realicen el proceso de encapsulación (Figura III.28), el valor de dirección MAC de origen es tomado de la variable local_mac.

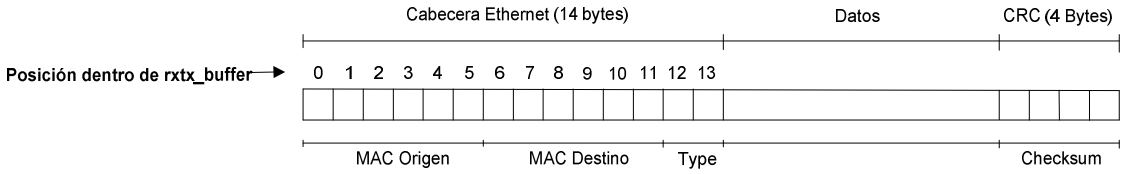


Figura III.28 Arquitectura de la Trama sobre la variable rxtx_buffer

3.8.1 Análisis del Checksum

Todos los protocolos implementados necesitan en su cabecera un Checksum o valor con el cual el receptor de datos sabe que recibió una trama, datagrama o segmento tal y como se envió. En caso de Ethernet el Checksum se ubicará al final de la trama y contendrá el valor de verificación de la cabecera y los datos.

En este caso el Checksum es calculado por el ENC28J60 en el momento de transmitir la trama. Esto se hizo en la inicialización del dispositivo indicándole que haga un relleno (PADDING) de la trama si los datos a enviarse son menores a 60 bytes y que añada un valor de CRC valido al

final antes de la transmisión, MICROCHIP (2). Con esto no es necesario preocuparse de calcular el CRC ni de que los datos que se envíen sean menores a 60 bytes pues el ENC28J60 lo hace automáticamente.

Sin embargo los protocolos de capas superiores como IP y TCP si necesitan un cálculo individual, por ello se creó la función ***software_checksum*** la cual realiza todo este proceso devolviendo un Checksum válido enviándole como parámetros la posición desde la cual se desea que empiece a calcular el Checksum dentro de la variable rxtx_buffer y una longitud de desplazamiento hasta donde se desea que se calcule.

3.9 Protocolo IP

Para la implementación del protocolo IP se genera la cabecera del datagrama y los valores adecuados en cada campo del mismo para una comunicación correcta a nivel de red. Esto se lo hace mediante la función ***ip_generate_header*** el cual se encarga de manipular la variable rxtx_buffer para que el datagrama se comunique con el servidor (Figura III.29).

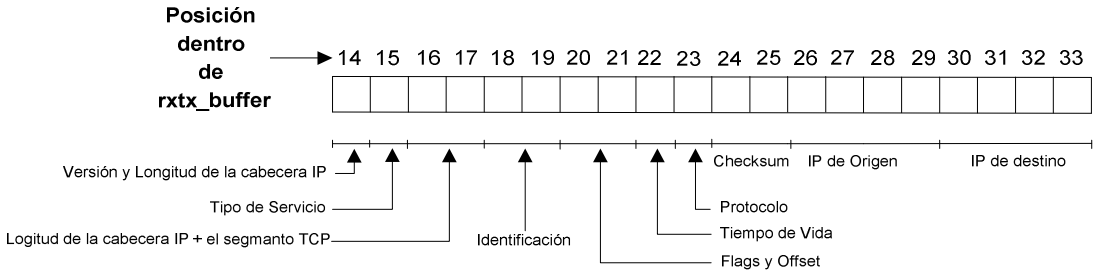


Figura III.29 Estructura del paquete IP en la variable rxtx_buffer

La fragmentación no se contempló en la implementación IP puesto que para el envío de datos planos no es necesario un datagrama mayor a los 1500 bytes, por ende el valor de las banderas (Flags) junto con el desplazamiento (Offset) será 0x00, y el campo identificación será un valor cualquiera. La longitud de la cabecera IP se expresa en grupos de 32 bits (4 bytes), en vista que la cabecera tiene 20 bytes de longitud (20 / 4 = 5), 5 será el valor a asignarse en este campo.

3.9.1 Esquema General

Figura III.30 Estructura del paquete IP

3.10 Protocolo ICMP

La implementación de ICMP de basa en la creación del paquete con la asignación adecuada de los valores a los diferentes parámetros que posee este protocolo (Figura III.31). La generación del paquete se realizó mediante la función ***icmp_generate_packet*** (Anexo 1). Esta función principalmente se encarga de generar el checksum para la cabecera ICMP y si se requiere se añade datos al azar al campo datos con una longitud máxima de 32 bytes (máximo valor permitido), ya que en el caso particular de una petición ICMP Echo se requiere de ello.

3.10.1 Esquema General

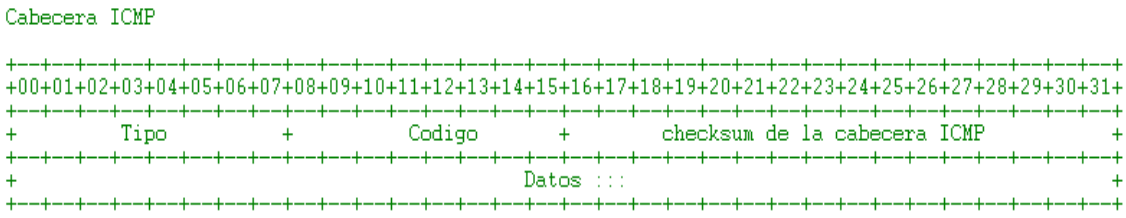


Figura III.31 Estructura del paquete ICMP

3.10.2 Petición ICMP

En la implementación de ICMP solo se integró la única utilidad de petición Echo pues será la única utilidad que se le ofrece al usuario del sistema para realizar pruebas de conectividad con entre el sistema y el host remoto. La petición ICMP se implementó a través de la función

icmp_send_request, la cual manipula la variable rxtx_buffer para crear una petición ICMP Echo (Figura III.32).

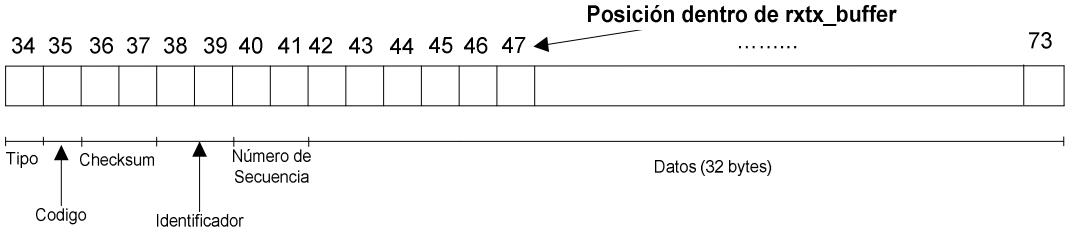


Figura III.32 Estructura de una petición ICMP Echo

3.10.3 Comando PING

La utilidad PING implementada en el sistema hace uso de una petición ICMP Echo, pero un envío de una petición ARP se realiza antes de la Petición Echo misma pues se necesita saber cual es la dirección MAC del host para que los dos dispositivos se comunique en la red, además de ser una buena manera de saber si el host se encuentra activo.

Para saber que el host respondió correctamente a la petición ICMP Echo se verifica que el campo protocolo de la cabecera IP dentro de la trama recibida tenga el valor 0x01 (ICMP) y también que el campo tipo en la cabecera ICMP de la trama recibida tenga un valor de 0x00 (Echo Reply). Todo este proceso se realiza en la función ***icmp_ping*** (Anexo 1). Como se muestra en la (Figura III.33).

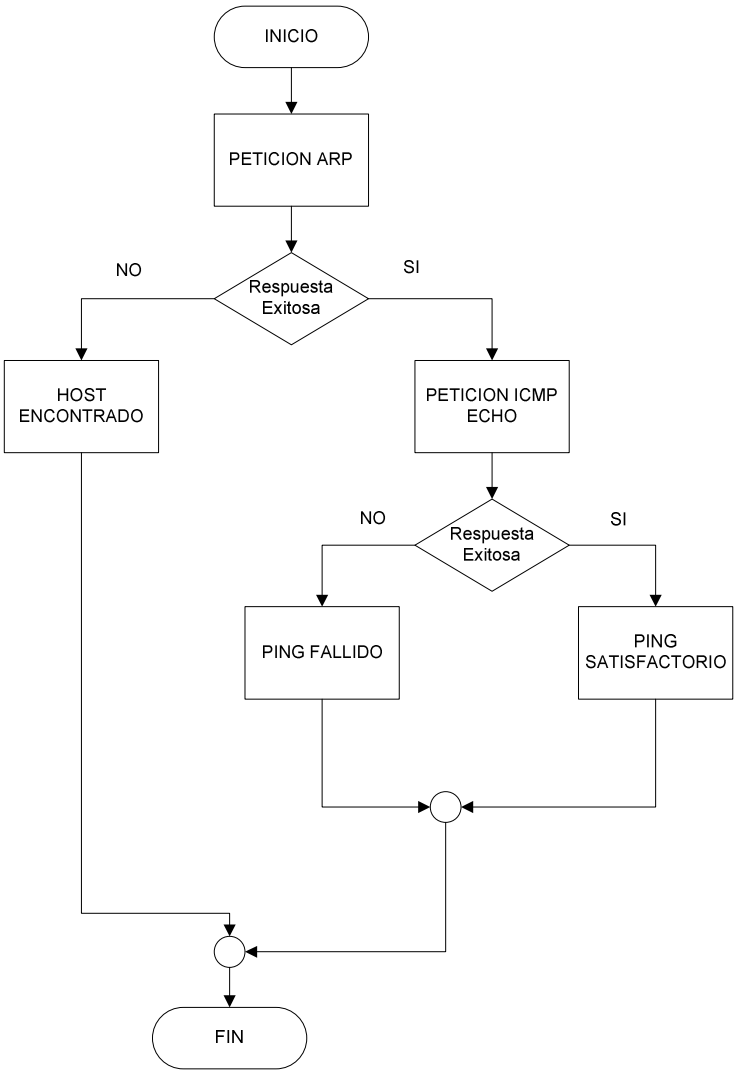


Figura III.33 Utilidad PING

CAPITULO IV

Análisis de Transmisión/Recepción de datos y Evaluación del Proyecto

4.1 Envío y Recepción de Datos

El envío de datos es realizado por la interfaz software CPanel que mediante el comando **"GA"** establece comunicación con el modulo AVRM (Anexo 1).

Una vez establecida la conexión es posible recibir datos del estado de los puertos del módulo AVRM, como:

- Dirección de giro del motor de CC
- Dirección IP del módulo AVRM
- Dirección IP del Servidor

El formato de respuesta que recibe el CPanel es:

OKLLAAA.AAA.AAA.AAA;SSS.SSS.SSS.SSS;\r\n como muestra la Figura IV.34, formato de respuesta explicado en el capitulo III.7.2

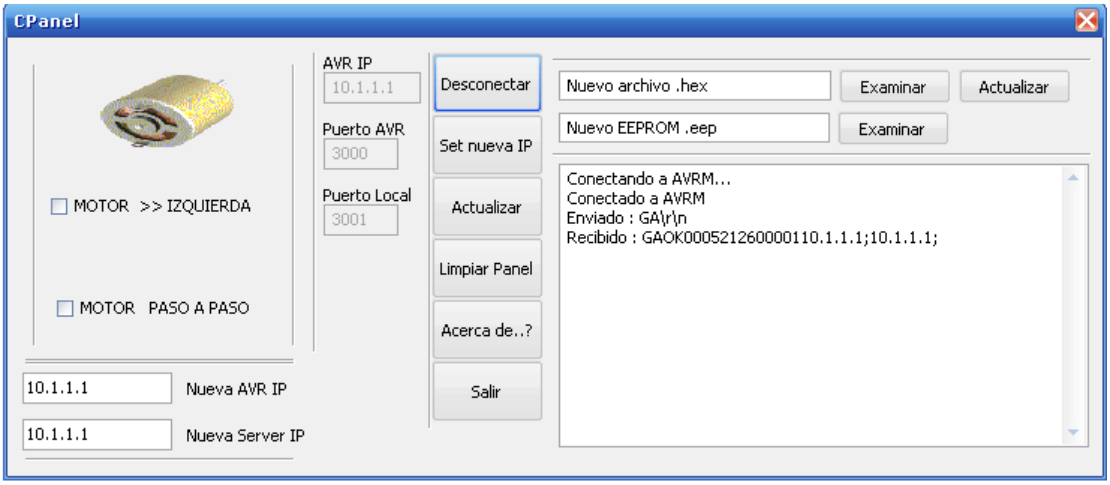


Figura IV.34 Cpanel, visualización de conexión establecida y recepción de datos

Como se muestra en la Figura IV.35, también es posible enviar datos al módulo AVRM como por ejemplo:

- Nueva dirección IP para el módulo AVRM
- Nueva dirección IP para el Servidor
- Estado del Motor y cambio de giro del motor de CC.

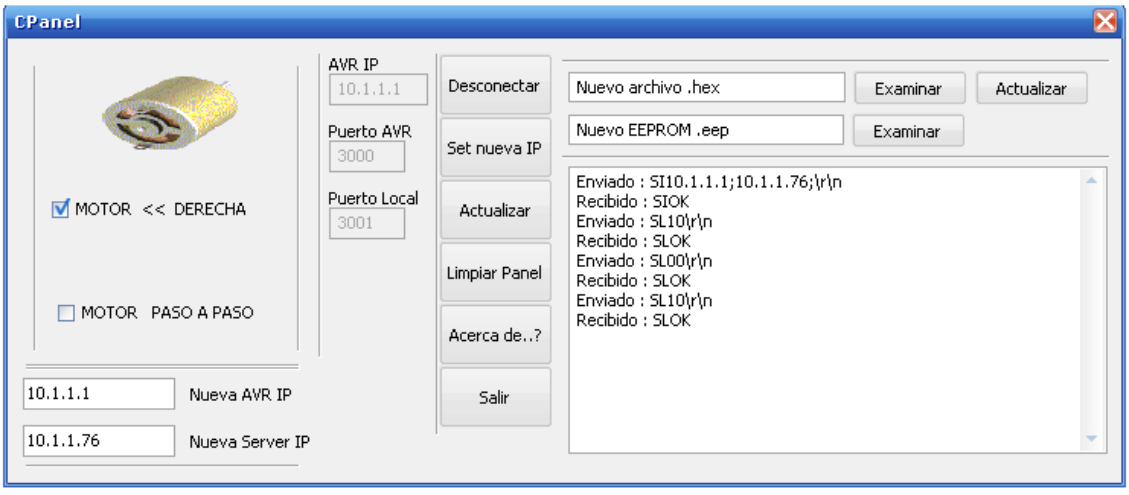


Figura IV.35 Cpanel, visualización de envío de datos

4.1.1 Control de Motor

Una vez pasado por el diseño de firmware para el control de motor de CC, el manejo del CPanel para este fin resulta muy sencillo ya que basta solo con marcar en el checkblok y cambiar de movimiento de derecha a izquierda o viceversa como muestra la Figura IV.36

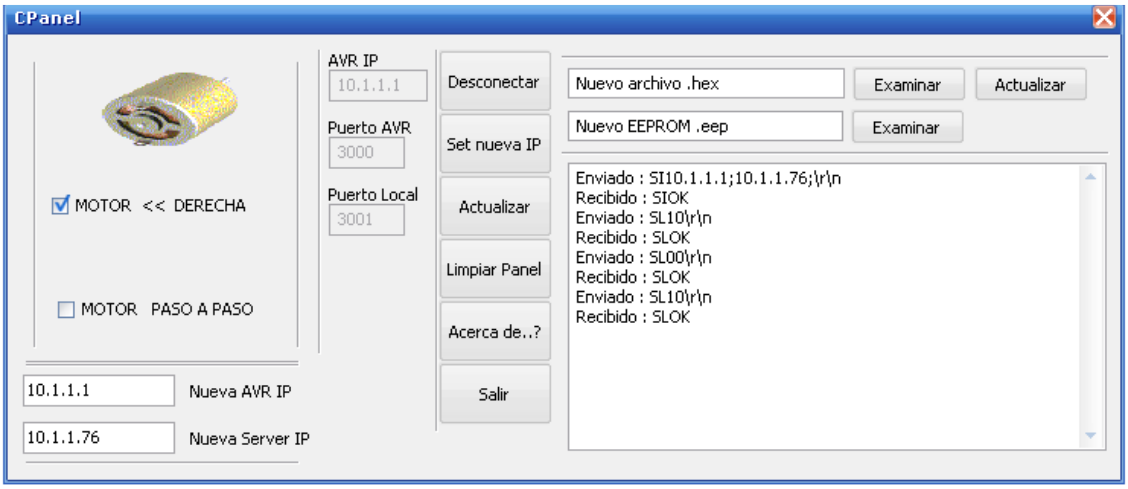


Figura IV.36 Cpanel, visualización control del motor de CC

4.1.2 Grabación de AVR o Actualización de firmware

La grabación remota de firmware en el AVR consta de 2 partes como se explicó en el Capitulo III.3.4.

La parte de lectura y descarga del archivo .hex, .eep y la parte de reseteo y ejecución del nuevo firmware en el AVR

El CPanel visualiza cada proceso que se ejecuta en el AVR como por ejemplo:

- Cálculo del tamaño del archivo .hex a cargar en Bytes
- Cálculo del tamaño del archivo .eep a cargar en Bytes
- Reseteo e Ingreso a proceso de descarga por Boot Loader
- Borrado del AVR
- Programación de flash y eeprom
- Reseteo del sistema y ejecución del nuevo firmware cargado en el AVR

En la Figura IV.37, muestra el proceso de carga del AVR o actualización de firmware.

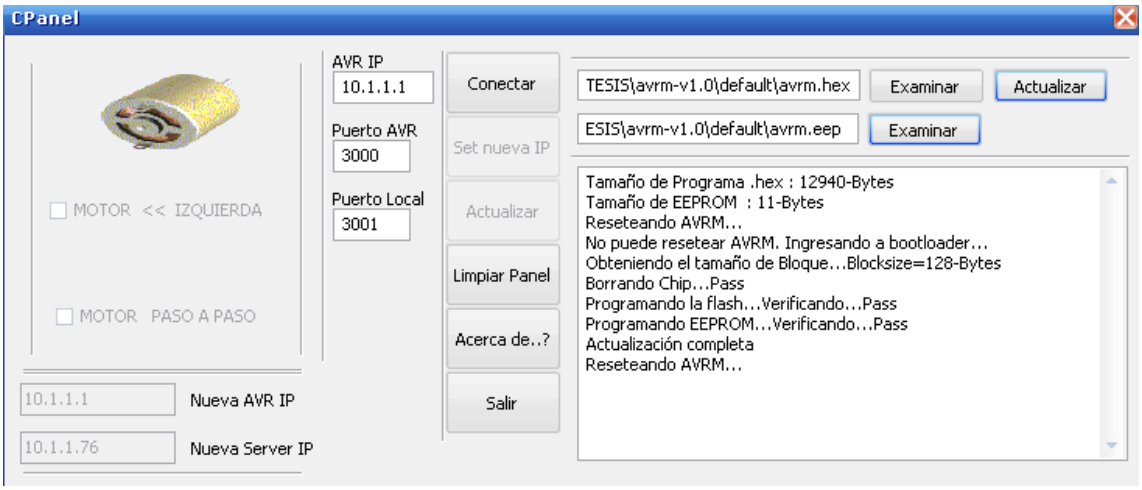


Figura IV.37 CPanel, visualización de cargado de AVR o actualización de firmware

4.1.3 Pruebas Realizadas

Para realizar pruebas se escogió una red local pública en donde el tráfico se vuelve pesado cuando varias personas envían y reciben peticiones web.

Para detectar la eficiencia del sistema se usó un analizador de protocolos o snifer (Wireshark), el cual me permite visualizar el proceso de envío y recepción de datos a través de la red.

4.1.4 Análisis de Funcionamiento de Protocolos

Con el analizador de protocolos se pudo medir la eficiencia de la implementación de protocolos, obteniendo los siguientes resultados.

- *Dirección del módulo AVRM:* 10.1.1.1
- *Dirección del servidor:* 10.1.1.76

En el (Anexo 2), se muestra detalladamente las capturas de tráfico que se generaron al enviar cada unos de los protocolos implementados en el sistema.

Protocolo ARP

La petición ARP es iniciada por el host remoto enviando la petición de IP numero: 10.1.1.1 generando un broadcast en toda la red, preguntando por la dirección MAC cuya dirección IP sea la 10.1.1.1 en donde el único que responde es el módulo AVR con una dirección MAC 41:56:52:50:4f:52, Esto se logra en un tiempo de respuesta de 20µs.

Protocolo ICMP

La petición ICMP echo request es iniciada por el host remoto con la dirección IP 10.1.1.76 con destino la IP 10.1.1.1 (Trama Nº 11), entonces el módulo AVR responde a esa petición (Trama Nº12) enviando un paquete ICMP echo reply. Esto se logra en un tiempo de respuesta de 32µs.

Protocolo UDP

La envío y recepción de datos en este caso el archivo .hex a o actualización de firmware a cargar en el AVR se lo hace a través del puerto 3000 y 3001, en el cual tiene una duración total de envío y recepción de 3.92093 segundos desde la Trama Nº1 hasta la Trama Nº 427, como muestra la captura del analizador de protocolos en el (Anexo 2).

4.2 Relación Costo Beneficio del Sistema

Existe una gran ventaja de implementar sistemas de control basados en networking embebido por la gran eficiencia y bajo costo que estos presentan al ser implementados, ya que se utiliza software con licencia GPL, que no representa ningún costo adicional al diseño del sistema como se ha demostrado en el Capitulo IV.

También se ha demostrado que el sistema diseñado esta basado en estudios para seleccionar la mejor alternativa de dispositivos para su implementación basados en parámetros como precio, disponibilidad y tecnología, Capitulo II.

4.2.1 Costos de Diseño e Implementación del Proyecto de Tesis

Los materiales utilizados en la fabricación del sistema fueron enteramente obtenidos en el mercado nacional, como se explicó en el (Capitulo 2). Utilizando las alternativas en Networking embebido más confiables y baratas.

En este caso el costo más elevado lo tiene el valor de Ingeniería para el diseño y mano de obra para la implementación como muestra la (Tabla IV.14).

Tabla IV.14 Costo total del sistema

Costos	**Valor**
Costos de Hardware	$ 70
Costo de Ingeniería e Implementación	$ 300
Total	**$ 370**

4.3 Resultados finales del Sistema

En el Capitulo 4, mediante las pruebas realizadas se demuestra confiabilidad y eficiencia del sistema, obteniendo tiempos de respuesta mínimos, casi imperceptibles al ojo humano.

La tiempo muerto mas extenso sucede cuando se realiza la carga o actualización de firmware en el AVR, debido a que se borra toda la aplicación del AVR, se carga una nueva y en el mismo instante comienza a ejecutarse las rutinas del nueva aplicación, todo este proceso tiene una duración máxima de 3.92093 segundos en la que el sistema de control de motores muestra una diminuta para en su funcionamiento normal que es imperceptible al ojo humano.

CONCLUSIONES

1) La utilización de Controladores Ethernet es una de las mejores alternativas en la implementación de sistemas que requieran conectarse a una red LAN mediante networking embebido ya que son una buena alternativa en función de eficiencia, bajo costo y disponibilidad.
2) Las prestabilidades que brindan los microcontroladores AVR facilitaron el diseño del auto programado en circuito en la implementación de carga del AVR desde un host remoto instalado en una red LAN, demostrando que son una buena alternativa de implementación para sistemas de este tipo.
3) La robustez, confiabilidad y facilidades de adquisición que presenta el software que funciona bajo licencia pública GPL permitieron abaratar considerablemente el costo final del sistema.
4) La implementación de una etapa de potencia para adaptar el motor de CC al módulo AVRM resultó ser una buena opción para adaptar o incorporar futuros módulos que requieran ser controlados por medio de este sistema.

RECOMENDACIONES

1) La seguridad es uno de los primeros aspectos a considerarse para desarrollos futuros, implementaciones como: cifrado, autenticación según como requiera un nuevo diseño.
2) Se puede añadir un visualizador o display para analizar las configuraciones hechas en el AVR como: direcciones IP, estado de los módulos controlados, etc., facilitando la información del usuario.
3) En diseños futuros, la implementación en base a IPv6 puede considerarse ya que solo depende del controlador Ethernet.
4) La implementación de más protocolos en la capa de aplicación es factible puesto que no es necesario realizar cambios en el hardware del sistema si no únicamente realizar arreglos en el firmware del sistema.

RESUMEN

El presente proyecto muestra el diseño y la implementación de un sistema de grabación remota de un AVR que se comunique mediante Ethernet dentro de una red local, para el control de un motor de Corriente Continua de 5 Voltios.

Para cumplir con este objetivo, se recurrió al uso de Networking Embebido para establecer comunicación con la red local mediante protocolos de comunicación como ICMP, ARP y UDP, a través de una interfaz de adaptación ENC28J60, controlada por medio de un microcontrolador ATMega32 programado en ANSI C, que hace las funciones de cerebro dentro del sistema.

Obteniendo así un diseño a bajo costo por los materiales utilizados, una implementación sencilla y fácil de realizar por el estudio de selección de la mejor alternativa en componentes electrónicos, obteniendo así un tiempo de implementación de 3 horas, un tiempo de programación en ANSI C máximo de 30 horas, demostrando eficiencia y confiabilidad de uso del diseño e implementación según parámetros como: tiempo máximo de grabación remota del microcontrolador 3.92093 segundos, tiempo de comunicación con la interfaz Ethernet 20µs, 32µs y 10ms para los protocolos ARP, ICMP, UDP respectivamente. Valores obtenidos al realizar pruebas utilizando un analizador de protocolos de comunicación y tráfico de red Wireshark, en una red local pública con altos márgenes de tráfico de internet.

El diseño y la implementación realizada alcanzó el objetivo de demostrar confiabilidad y eficiencia al poder controlar un motor de corriente continua de 5 voltios dentro de una red local, mediante un sistema de grabación remota de un microcontrolador utilizando protocolos de comunicación de red.

SUMMARY

The present project shows the design and implementation of remote recording system of an AVR to communicate through Ethernet within a local network, to control a 5-volt continuous current motor.

To accomplish this objective the engaged networking was used to establish communication protocols such as ICMP, ARP and UDP by means of an adaptation interface ENC28J60 controlled by a microcontroller ATMega32 programmed in ANSI C, which functions a brain within the system.

Thus a low-cost design of the used material, and implementations easy and simple to carry out through the selection study of the best alternative in electronic components is attained having an implementation time of 3 hours, an ANSI C maximum programming time of 30 hours, demonstrating efficiency and reliability on the design use and implementation according to parameters such as: maximum remote recording time of the microcontroller, 3.92093 seconds, 20µ, 32µ and 10ms Ethernet interface communication time for protocols ARP, ICMP and UDP respectively. These are the values obtained upon carrying out test using an analyzer of traffic and communication protocols of Wireshark network in a local public network with high internet traffic margins.

The design and the implementation reached the objective to demonstrate reliability and efficiency because a continuous current 5-volt motor is controlled within a local network, through a remote recording system of a microcontroller using network communication protocols.

GLOSARIO

ATmega32: Circuito Integrado perteneciente a la familia AVR de ATMEL con la capacidad de almacenar hasta 32Kb de datos en su memoria flash, de ahí su nombre.

AVR: Familia de microprocesadores RISC de la empresa ATMEL CORP. La arquitectura de los AVR fue concebida por dos estudiantes en el Norwegian Institute of Technology, y posteriormente refinada y desarrollada en Atmel Norway, la empresa subsidiaria de ATMEL, fundada por los dos arquitectos del chip.

Breadboard: placa de uso genérico reutilizable o semi permanente, usado para construir prototipos de circuitos electrónicos con o sin soldadura. Normalmente se utilizan para la realización de pruebas experimentales.

Broadcast: Paquete de datos enviado a todos los nodos de una red. Los paquetes o tramas broadcasts se identifican por una dirección broadcast, FF:FF:FF:FF:FF o 255.255.255.255 respectivamente.

Buffer: Memoria intermedia que se utiliza como memoria de datos temporal durante una sesión de trabajo

Cabecera: se refiere a la información suplementaria situada al principio de un bloque de información que va a ser almacenada o transmitida y que contiene información necesaria para el correcto tratamiento del bloque de información.

Checksum: Permite la verificación de los datos que se envían a través de las tramas de TCP/IP.

Datagrama: Agrupamiento lógico de información enviada como unidad de capa de red a través de un medio de transmisión sin establecer previamente un circuito virtual. Los datagramas IP son las unidades de información primaria de la Internet.

Ethernet: Especificación de red de área local (LAN) desarrollada en 1976 por Xerox, en cooperación con DEC e Intel, originalmente para conectar los miniordenadores del Palo Alto

Research Center (EEUU). Se trata de una red muy difundida, de la cual se derivó la norma (o estándar) IEEE 802.3 para redes de conexión.

ENC28J60: Circuito Integrado perteneciente al fabricante MICROCHIP TECHNOLOGY, que sirve como medio de comunicación entre un microprocesador y una red Ethernet.

Firmware: Es el " software que está dentro del hardware". Se refiere a los programas grabados en memorias no volátiles cuya función es asegurar su correcto funcionamiento.

Full-Duplex: Capacidad para la transmisión simultánea de datos entre la estación emisora y la estación receptora.

Half-Dyplex: Capacidad de transmisión de datos en una sola dirección a la vez entre una estación transmisora y otra receptora.

LIBC: biblioteca estándar de funciones C para proyecto GNU o glibc. Aunque existe un conjunto de funciones orientado al manejo de microprocesadores AVR.

Networking: Interconexión de estaciones de trabajo, dispositivos periféricos (por ejemplo, impresoras, unidades de disco duro, escáneres y CD-ROM) y otros dispositivos.

Paquete: Agrupación lógica de información que incluye un encabezado que contiene la información de control y (generalmente) los datos del usuario. Los paquetes se usan a menudo para referirse a las unidades de datos de capa de red. Los términos datagrama, trama, mensaje y segmento también se usan para describir agrupamientos de información lógica en las diversas capas del modelo de referencia OSI y en varios círculos tecnológicos.

Segmento: Unidad única de información de capa de transporte.

Trama: Unidad única de información de capa de enlace de datos.

Unicast: Mensaje que se envía a un solo destino de red.

ANEXOS

ANEXO 1:

1.1 Diagramas de flujo del firmware

MAIN.C

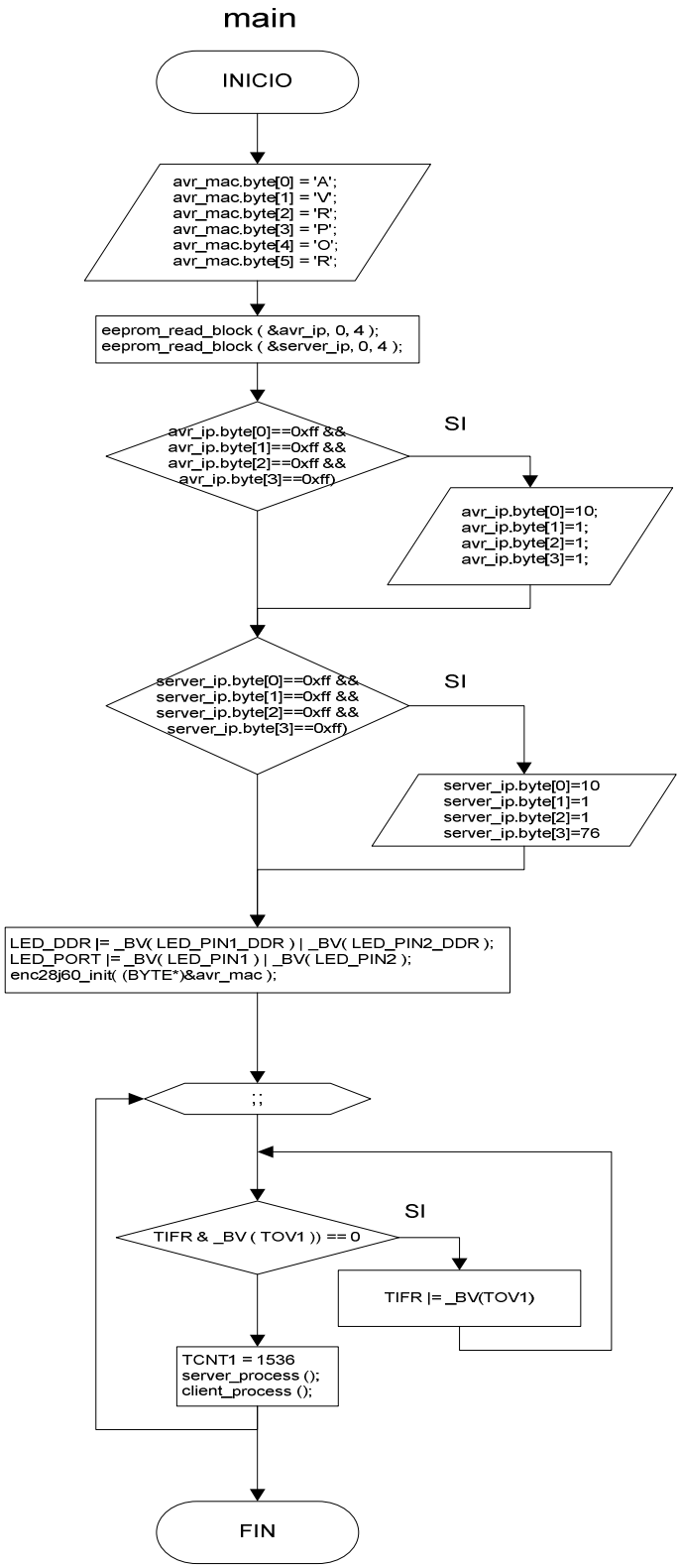
DESCRIPCION: Archivo principal del sistema contiene las estructuras centrales en las que se basa el sistema.

LIBRERIAS: includes.h

VARIABLES GLOBALES DEL ARCHIVO:

Variable	Tipo	Inicialización	Función
local_mac	MAC_ADDR	-	Define el valor de la dirección MAC local
local_ip	IP_ADDR	-	Define el valor de la dirección IP local
server_mac	MAC_ADDR	-	Define el valor de la dirección MAC del servidor
server_ip	IP_ADDR	-	Define el valor de la dirección IP del servidor
ee_local_ip	BYTE	10, 0, 0, 1	Define el valor de la dirección IP local, pero esta es almacenada en la memoria EEPROM del microprocesador
ee_server_ip	BYTE	10, 0, 0, 76	Define el valor de la dirección IP del servidor, pero esta es almacenada en la memoria EEPROM del microprocesador

FUNCIONES:


main
INICIO
avr_mac.byte[0] = 'A';
avr_mac.byte[1] = 'V';
avr_mac.byte[2] = 'R';
avr_mac.byte[3] = 'P';
avr_mac.byte[4] = 'O';
avr_mac.byte[5] = 'R';
eeprom_read_block (&avr_ip, 0, 4);
eeprom_read_block (&server_ip, 0, 4);
avr_ip.byte[0]==0xff &&
avr_ip.byte[1]==0xff &&
avr_ip.byte[2]==0xff &&
avr_ip.byte[3]==0xff)
SI
avr_ip.byte[0]=10;
avr_ip.byte[1]=1;
avr_ip.byte[2]=1;
avr_ip.byte[3]=1;
server_ip.byte[0]==0xff &&
server_ip.byte[1]==0xff &&
server_ip.byte[2]==0xff &&
server_ip.byte[3]==0xff)
SI
server_ip.byte[0]=10
server_ip.byte[1]=1
server_ip.byte[2]=1
server_ip.byte[3]=76
LED_DDR |= _BV(LED_PIN1_DDR) | _BV(LED_PIN2_DDR);
LED_PORT |= _BV(LED_PIN1) | _BV(LED_PIN2);
enc28j60_init((BYTE*)&avr_mac);
;;
TIFR & _BV (TOV1)) == 0
SI
TIFR |= _BV(TOV1)
TCNT1 = 1536
server_process ();
client_process ();
FIN

ENC28J60.C

DESCRIPCION: Contiene todas las funciones que permiten el manejo de la interfaz hacia Ethernet mediante el circuito integrado ENC28J60.

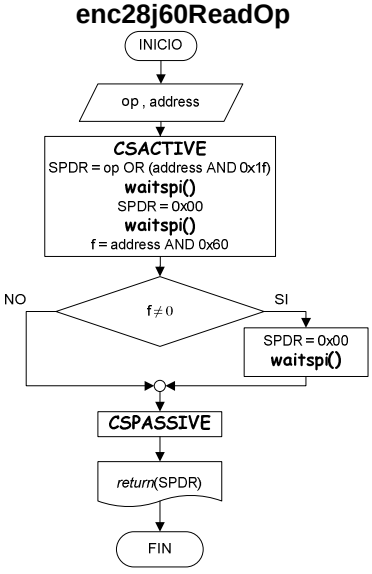
LIBRERIAS: includes.h

VARIABLES GLOBALES DEL ARCHIVO:

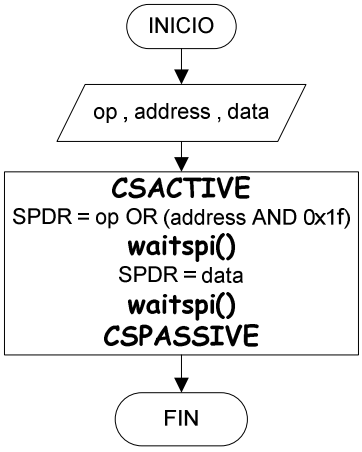
Variable	Tipo	Inicialización	Función
Enc28j60Bank	BYTE	-	Almacenar el banco de registros actual del ENC28J60
next_packet_ptr	WORD_BYTES	-	Almacenar la posición del puntero de lectura/escritura temporalmente

FUNCIONES:

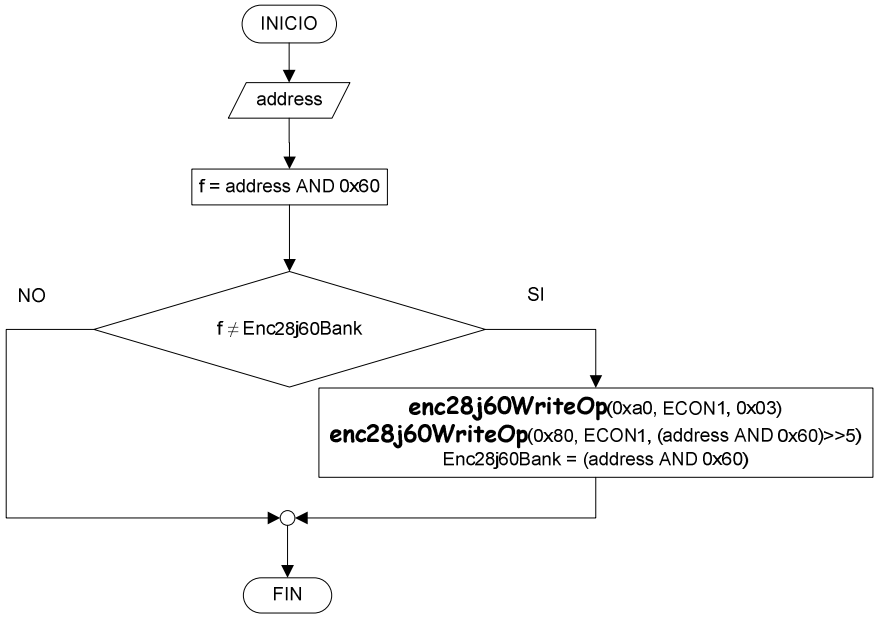
enc28j60ReadOp



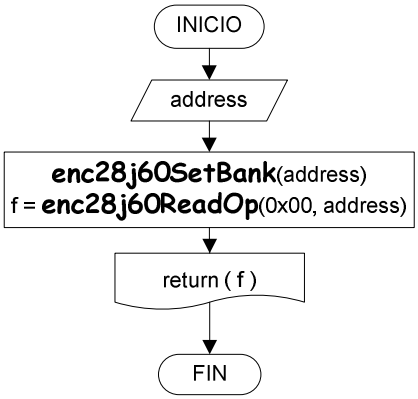
enc28j60WriteOp



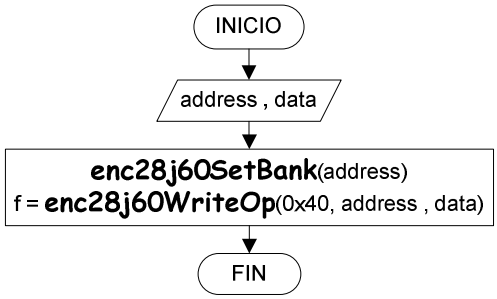
enc28j60SetBank



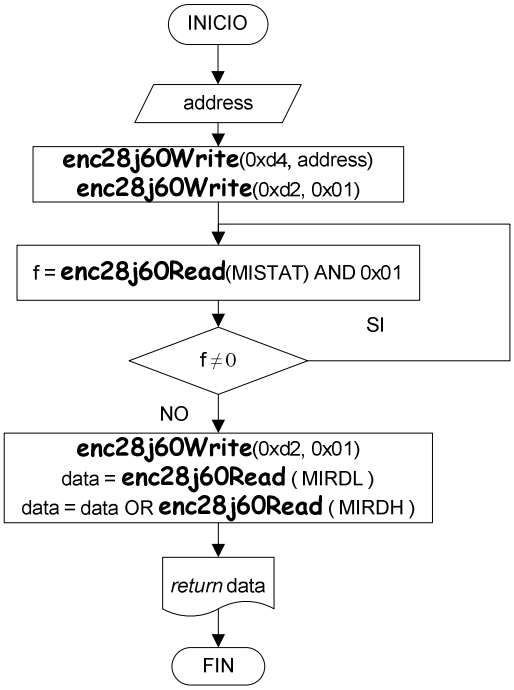
enc28j60Read



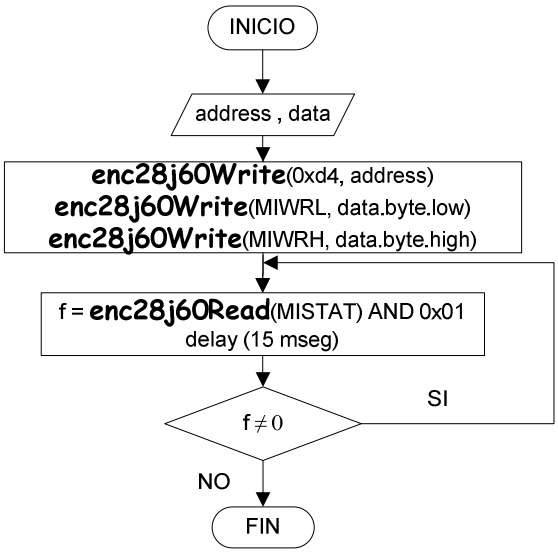
enc28j60Write



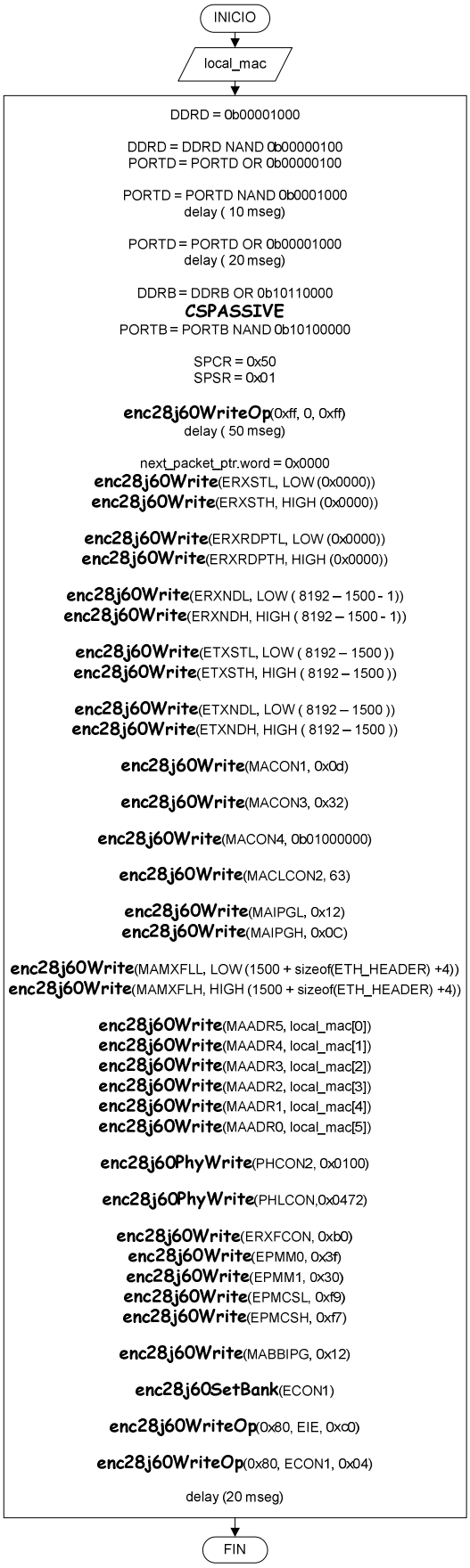
enc28j60_read_phyreg



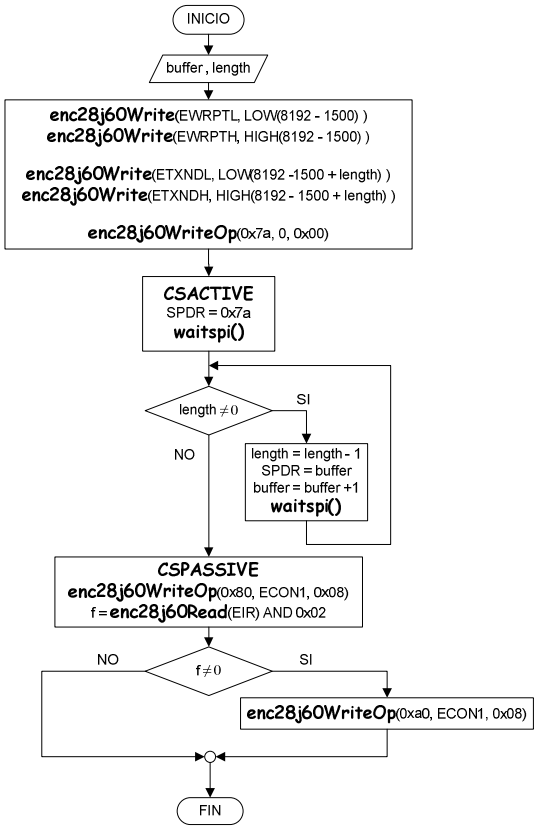
enc28j60PhyWrite



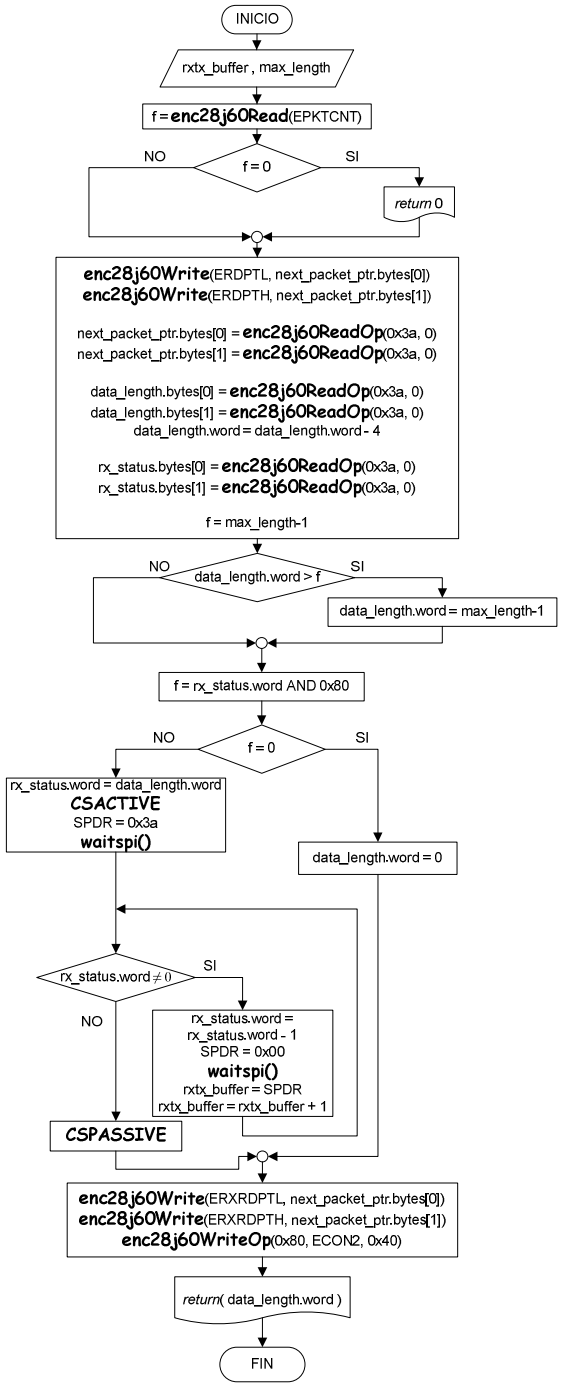
enc28j60_init



enc28j60_packet_send



enc28j60_packet_receive



UDP.C

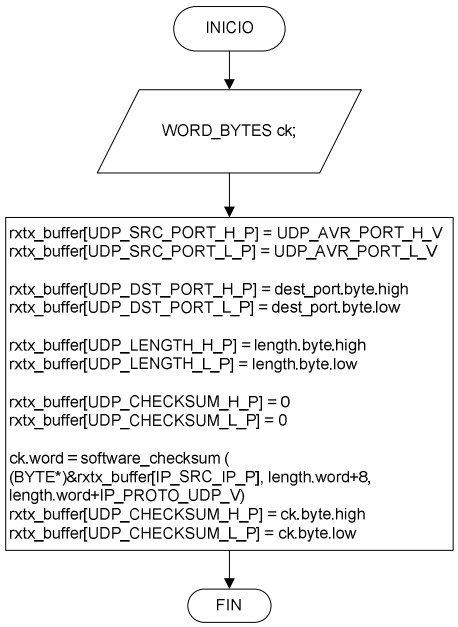
DESCRIPCION: Contiene todas las funciones que permiten la gestión del protocolo UDP.

LIBRERIAS: includes.h

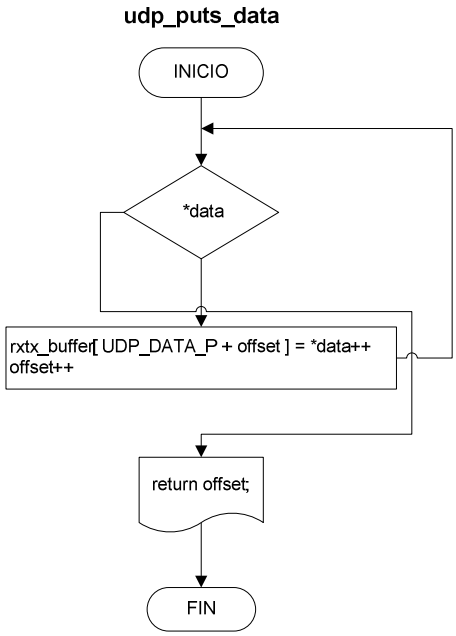
VARIABLES GLOBALES DEL ARCHIVO: Ninguna

FUNCIONES: udp_generate_header, udp_puts_data, udp_puts_data_p, udp_receive

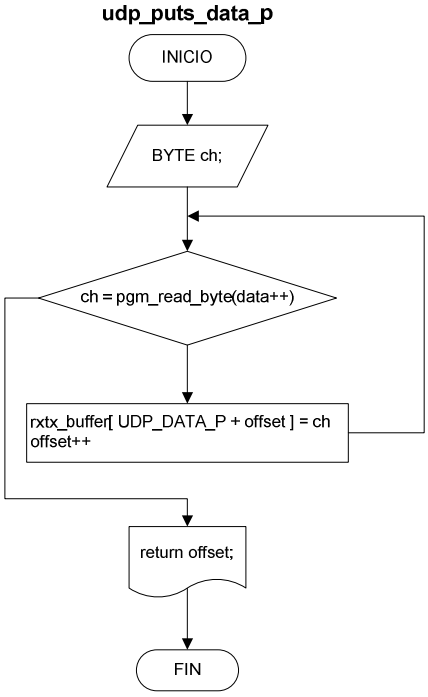
udp_generate_header

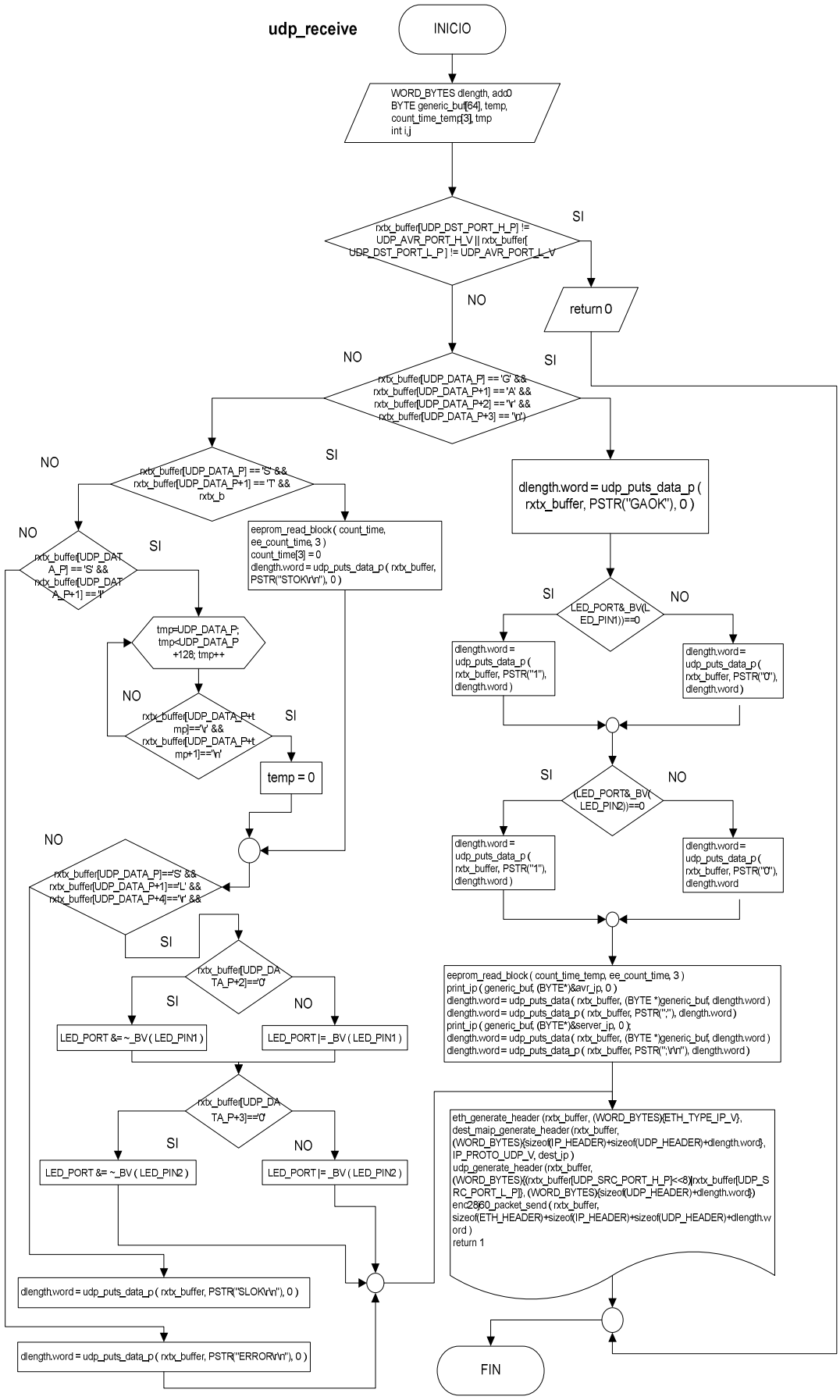


udp_puts_data



udp_puts_data_p




udp_receive
INICIO
WORD_BYTES dlength, add0
BYTE generic_buf[64], temp,
count_time_temp[3], tmp
int i,j
rxtx_buffer[UDP_DST_PORT_H_P] != UDP_AVR_PORT_H_V || rxtx_buffer[UDP_DST_PORT_L_P] != UDP_AVR_PORT_L_V
SI
NO
return 0
NO
SI
rxtx_buffer[UDP_DATA_P] == 'G' && rxtx_buffer[UDP_DATA_P+1] == 'A' && rxtx_buffer[UDP_DATA_P+2] == 'o' && rxtx_buffer[UDP_DATA_P+3] == 'o'
NO
SI
rxtx_buffer[UDP_DATA_P] == 'S' && rxtx_buffer[UDP_DATA_P+1] == 'T' && rxtx_b
dlength.word = udp_puts_data_p (rxtx_buffer, PSTR("GAOK"), 0)
NO
SI
rxtx_buffer[UDP_DATA_P] == 'S' && rxtx_buffer[UDP_DATA_P+1] == 'I'
eeprom_read_block (count_time, ee_count_time, 3)
count_time[3] = 0
dlength.word = udp_puts_data_p (rxtx_buffer, PSTR("STOK\r\n"), 0)
SI
NO
LED_PORT&_BV(LED_PIN1))==0
dlength.word = udp_puts_data_p (rxtx_buffer, PSTR("1"), dlength.word)
dlength.word = udp_puts_data_p (rxtx_buffer, PSTR("0"), dlength.word)
tmp=UDP_DATA_P; tmp<UDP_DATA_P +128; tmp++
NO
SI
rxtx_buffer[UDP_DATA_P+tmp]=='\r' && rxtx_buffer[UDP_DATA_P+tmp+1]=='\n'
temp = 0
SI
NO
(LED_PORT&_BV(LED_PIN2))==0
dlength.word = udp_puts_data_p (rxtx_buffer, PSTR("1"), dlength.word)
dlength.word = udp_puts_data_p (rxtx_buffer, PSTR("0"), dlength.word
NO
rxtx_buffer[UDP_DATA_P]=='S' && rxtx_buffer[UDP_DATA_P+1]=='L' && rxtx_buffer[UDP_DATA_P+4]=='r' &&
SI
rxtx_buffer[UDP_DATA_P+2]=='0'
SI
NO
LED_PORT &= ~_BV (LED_PIN1)
LED_PORT |= _BV (LED_PIN1)
rxtx_buffer[UDP_DATA_P+3]=='0'
SI
NO
LED_PORT &= ~_BV (LED_PIN2)
LED_PORT |= _BV (LED_PIN2)
eeprom_read_block (count_time_temp, ee_count_time, 3)
print_ip (generic_buf, (BYTE*)&avr_ip, 0)
dlength.word = udp_puts_data (rxtx_buffer, (BYTE *)generic_buf, dlength.word)
dlength.word = udp_puts_data_p (rxtx_buffer, PSTR(";"), dlength.word)
print_ip (generic_buf, (BYTE*)&server_ip, 0);
dlength.word = udp_puts_data (rxtx_buffer, (BYTE *)generic_buf, dlength.word)
dlength.word = udp_puts_data_p (rxtx_buffer, PSTR(";\r\n"), dlength.word)
eth_generate_header (rxtx_buffer, (WORD_BYTES){ETH_TYPE_IP_V}, dest_maip_generate_header (rxtx_buffer, (WORD_BYTES){sizeof(IP_HEADER)+sizeof(UDP_HEADER)+dlength.word}, IP_PROTO_UDP_V, dest_ip)
udp_generate_header (rxtx_buffer, (WORD_BYTES){(rxtx_buffer[UDP_SRC_PORT_H_P]<<8)|(rxtx_buffer[UDP_SRC_PORT_L_P]}, (WORD_BYTES){sizeof(UDP_HEADER)+dlength.word})
enc28j60_packet_send (rxtx_buffer, sizeof(ETH_HEADER)+sizeof(IP_HEADER)+sizeof(UDP_HEADER)+dlength.word)
return 1
dlength.word = udp_puts_data_p (rxtx_buffer, PSTR("SLOK\r\n"), 0)
dlength.word = udp_puts_data_p (rxtx_buffer, PSTR("ERROR\r\n"), 0)
FIN

ETHERNET.C

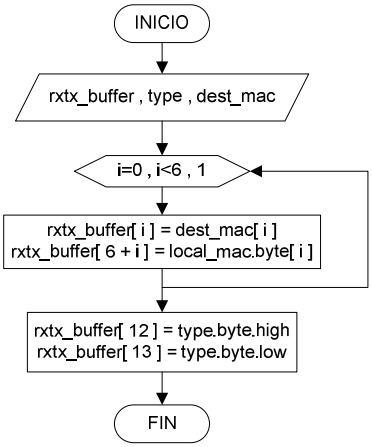
DESCRIPCION: Contiene todas las funciones que permiten formar la trama Ethernet.

LIBRERIAS: includes.h

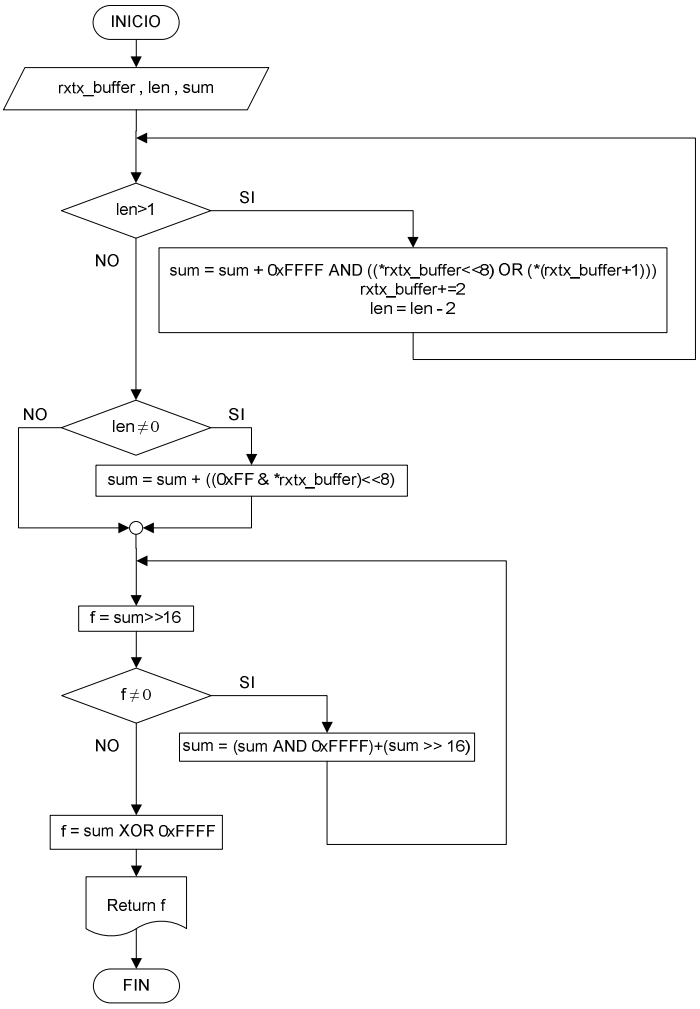
VARIABLES GLOBALES EN EL ARCHIVO: Ninguna

FUNCIONES:

eth_generate_header



software_checksum



ARP.C

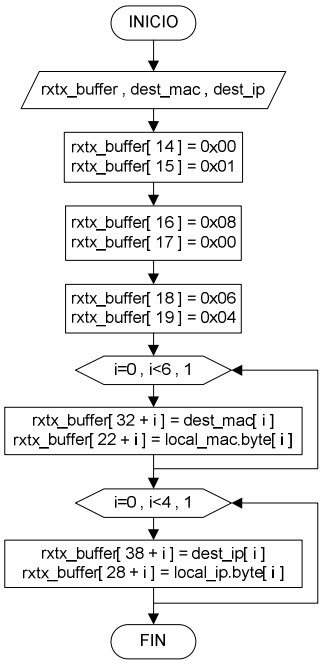
DESCRIPCION: Contiene todas las funciones que permiten la gestión del protocolo ARP.

LIBRERIAS: includes.h

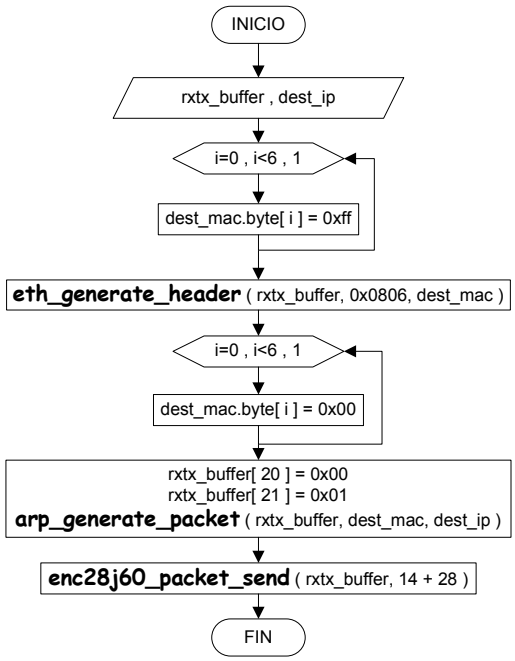
VARIABLES GLOBALES DEL ARCHIVO: Ninguna

FUNCIONES:

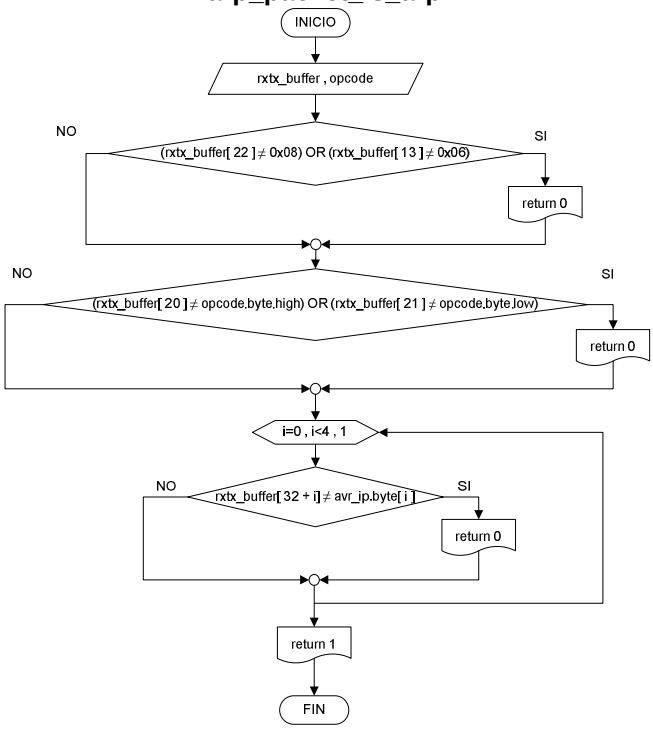
arp_generate_packet



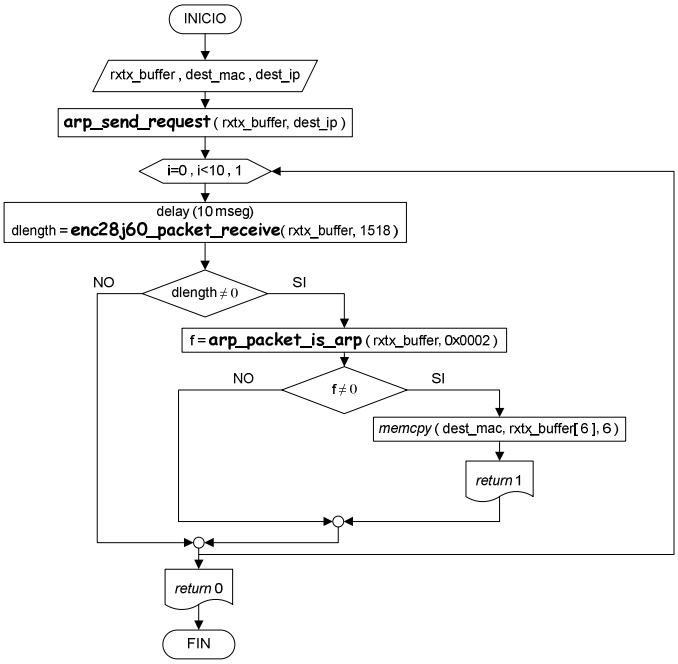
arp_send_request



arp_packet_is_arp



arp_who_is



IP.C

DESCRIPCION: Contiene todas las funciones que permiten la gestión del protocolo IP.

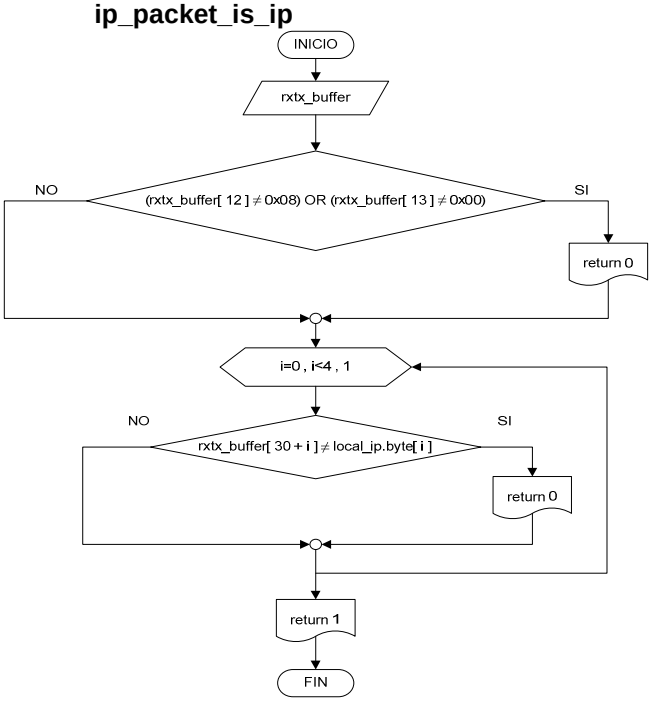
LIBRERIAS: includes.h

VARIABLES GLOBALES DEL ARCHIVO:

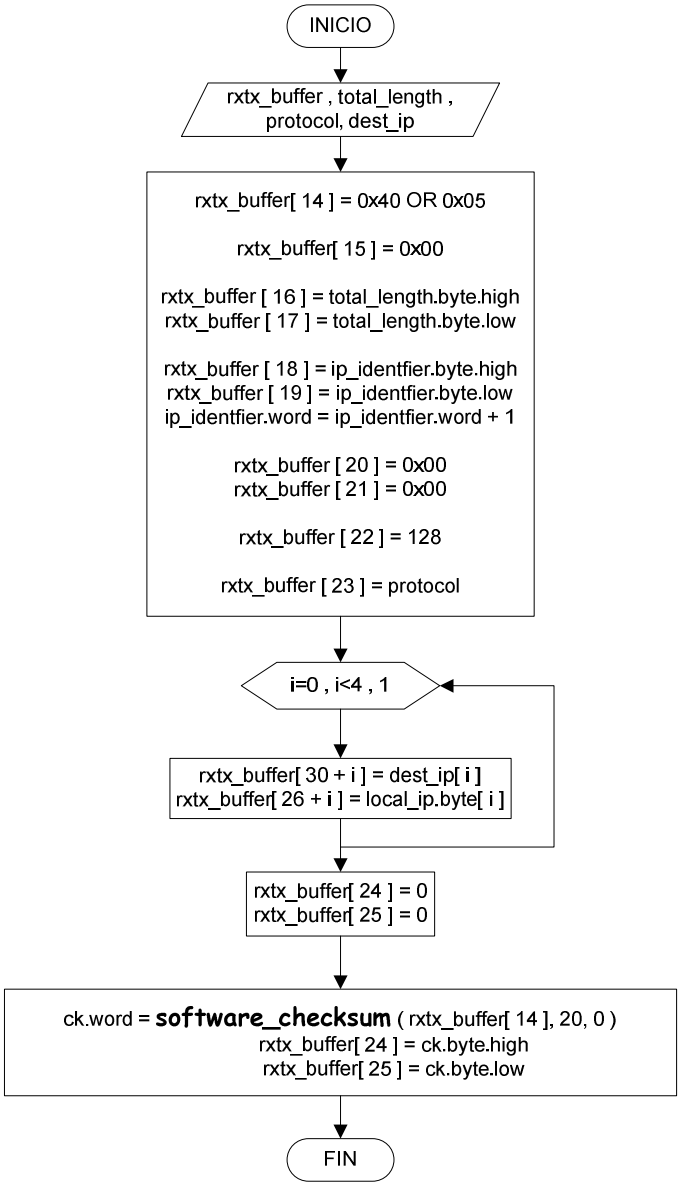
Variable	Tipo	Inicialización	Función
ip_identfier	WORD_BYTES	1	Identificador de la cabecera del Datagrama IP

FUNCIONES:

ip_packet_is_ip



ip_generate_header



ICMP.C

DESCRIPCION: Contiene todas las funciones que permiten la gestión del protocolo ICMP.

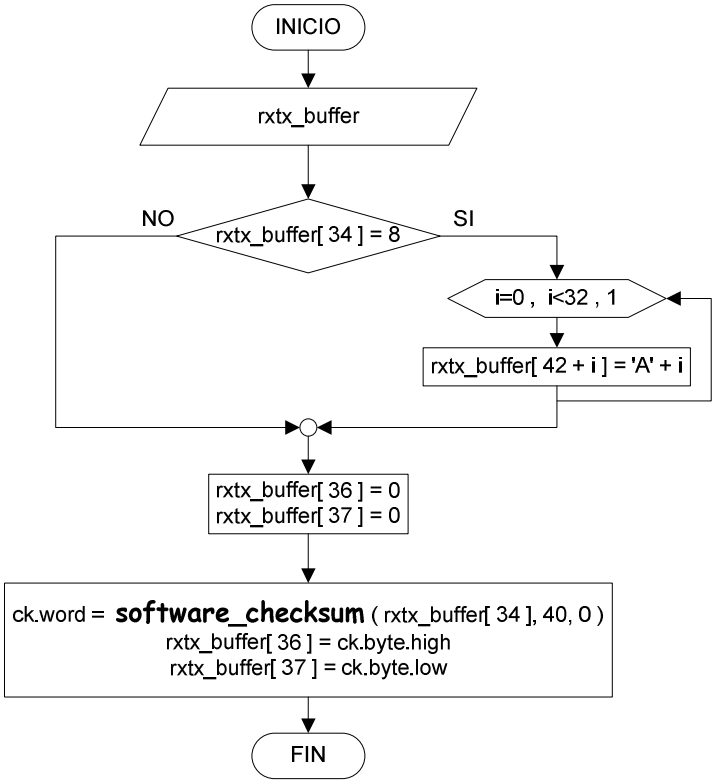
LIBRERIAS: includes.h

VARIABLES GLOBALESDEL ARCHIVO:

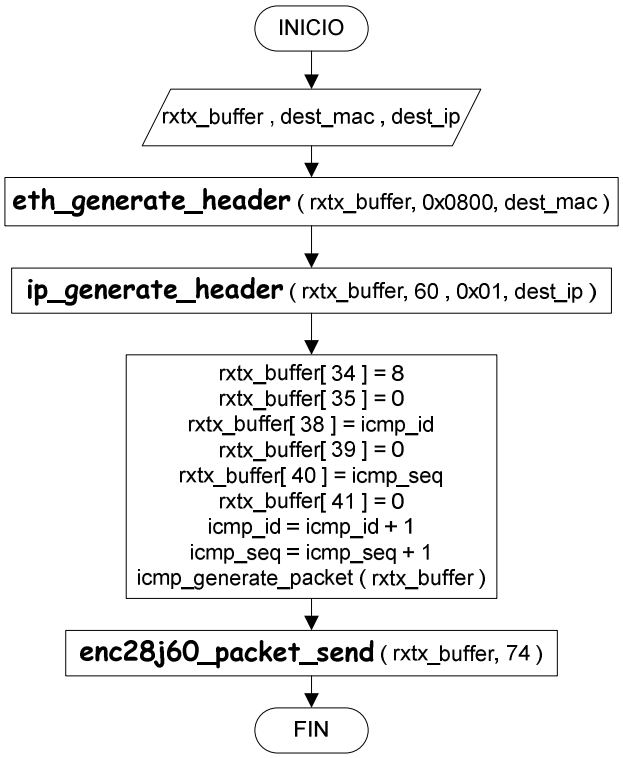
Variable	Tipo	Inicialización	Función
icmp_id	unsigned char	1	Identificador de la cabecera del Paquete ICM
icmp_seq	unsigned char	1	Identificador del numero de secuencia del paquete ICMP

FUNCIONES:

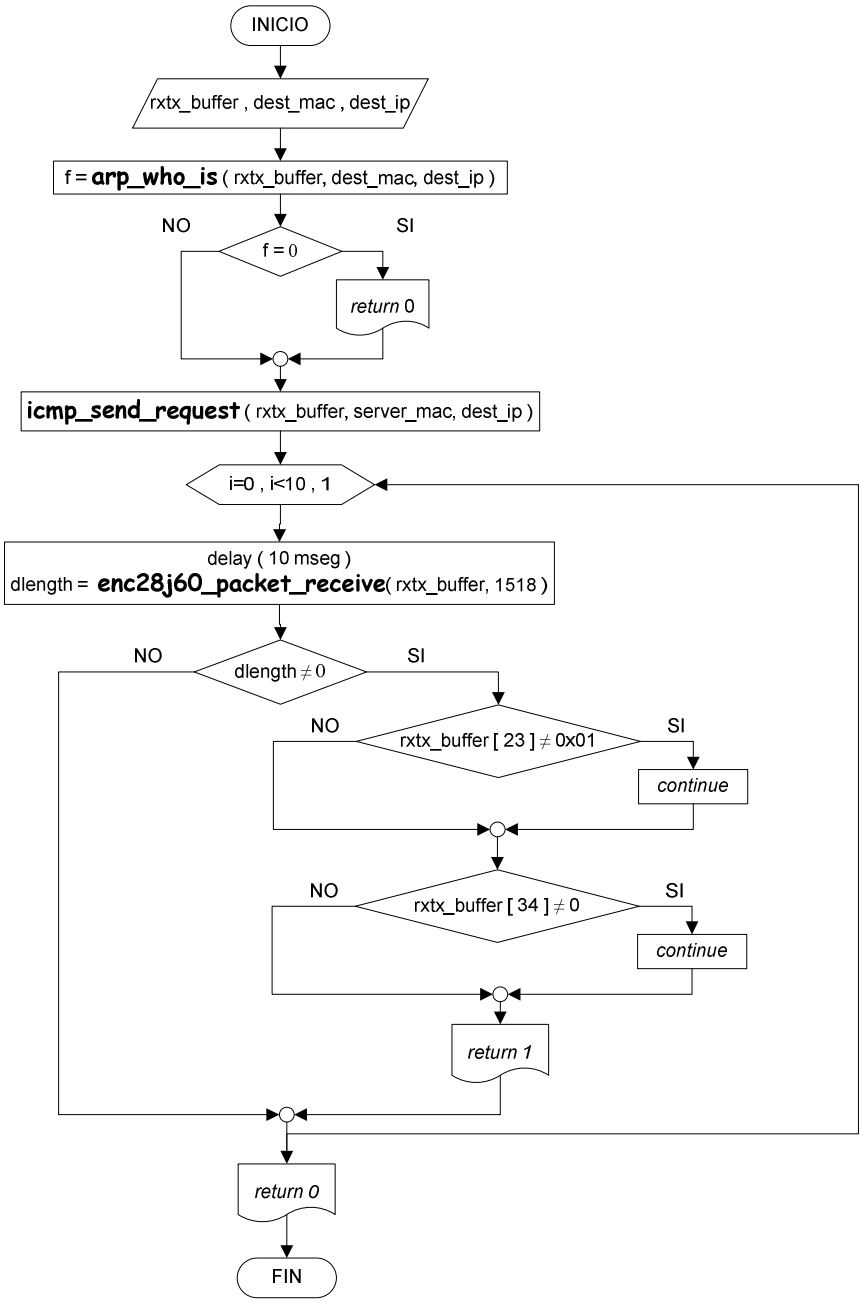
icmp_generate_packet



icmp_send_request



icmp_ping


INICIO
rxtx_buffer , dest_mac , dest_ip
f = arp_who_is (rxtx_buffer, dest_mac, dest_ip)
NO
SI
f = 0
return 0
icmp_send_request (rxtx_buffer, server_mac, dest_ip)
i=0 , i<10 , 1
delay (10 mseg)
dlength = enc28j60_packet_receive(rxtx_buffer, 1518)
NO
SI
dlength ≠ 0
NO
SI
rxtx_buffer [23] ≠ 0x01
continue
NO
SI
rxtx_buffer [34] ≠ 0
continue
return 1
return 0
FIN

1.2 Diagramas de flujo del CPanel.

AVRMain.cpp

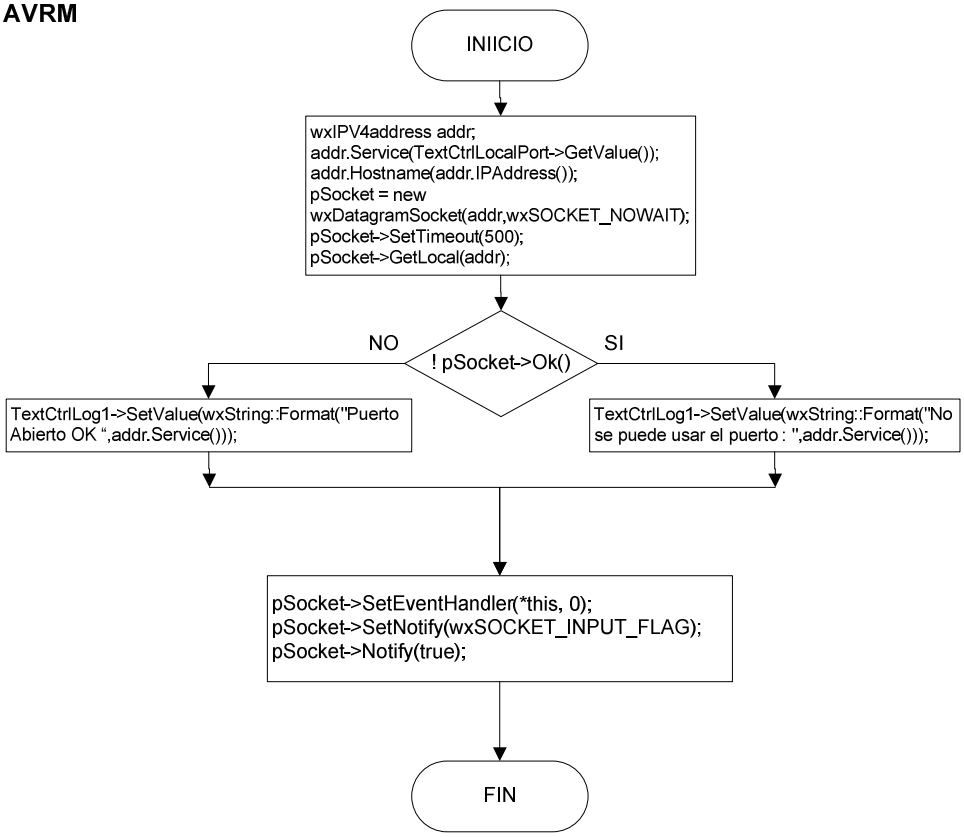
DESCRIPCION: Contiene todas las funciones que permiten controlar cada una de las funciones programadas en el firmware.

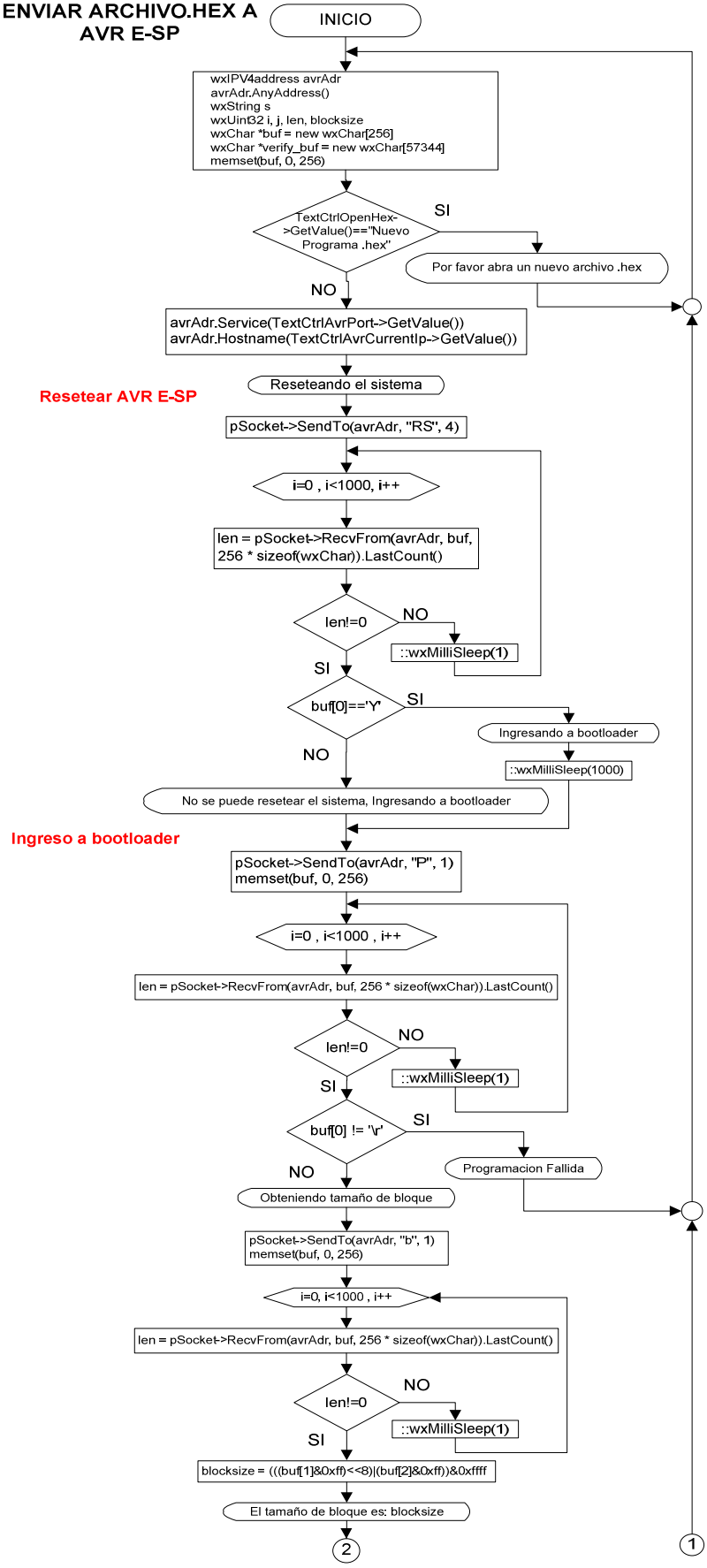
LIBRERIAS: sizer.h, stattext.h, textctrl.h, , checkbox.h, statline.h, statbmp.h, button.h, dialog.h, socket.h, utils.h, filedlg.h, file.h, hyperlink.h

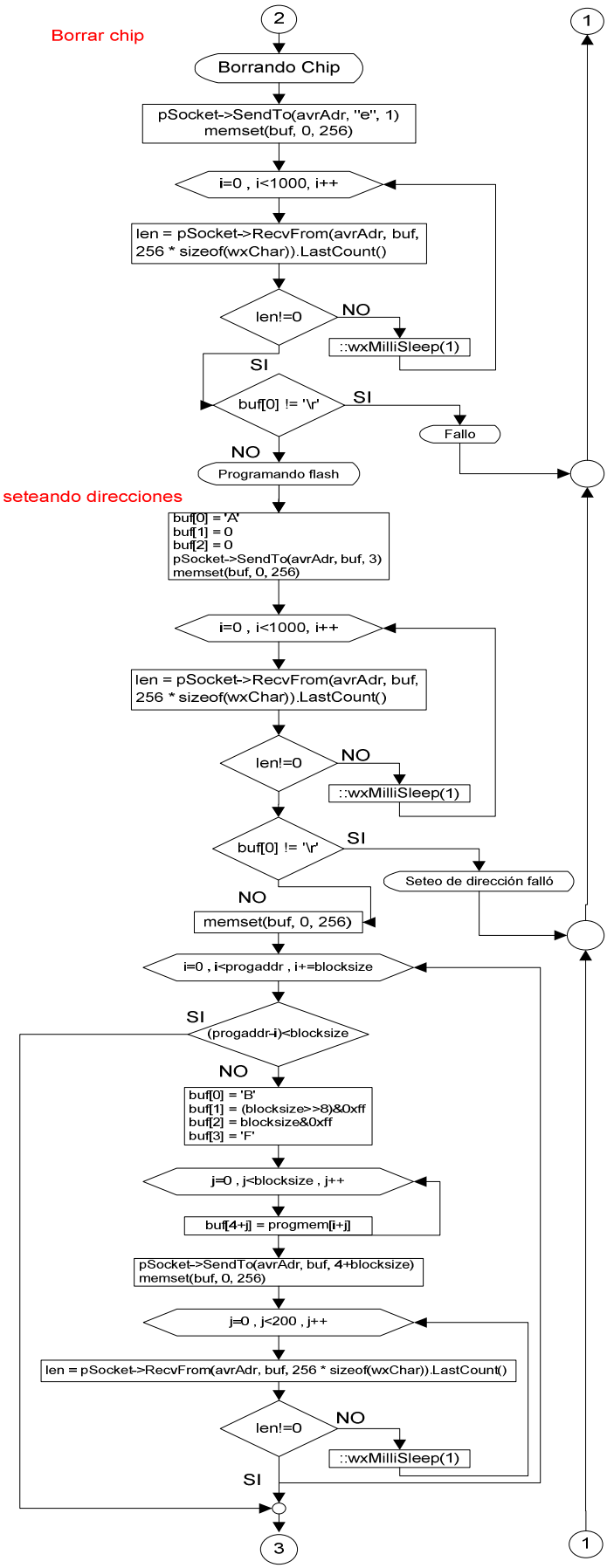
VARIABLES GLOBALES DEL ARCHIVO: Ninguna

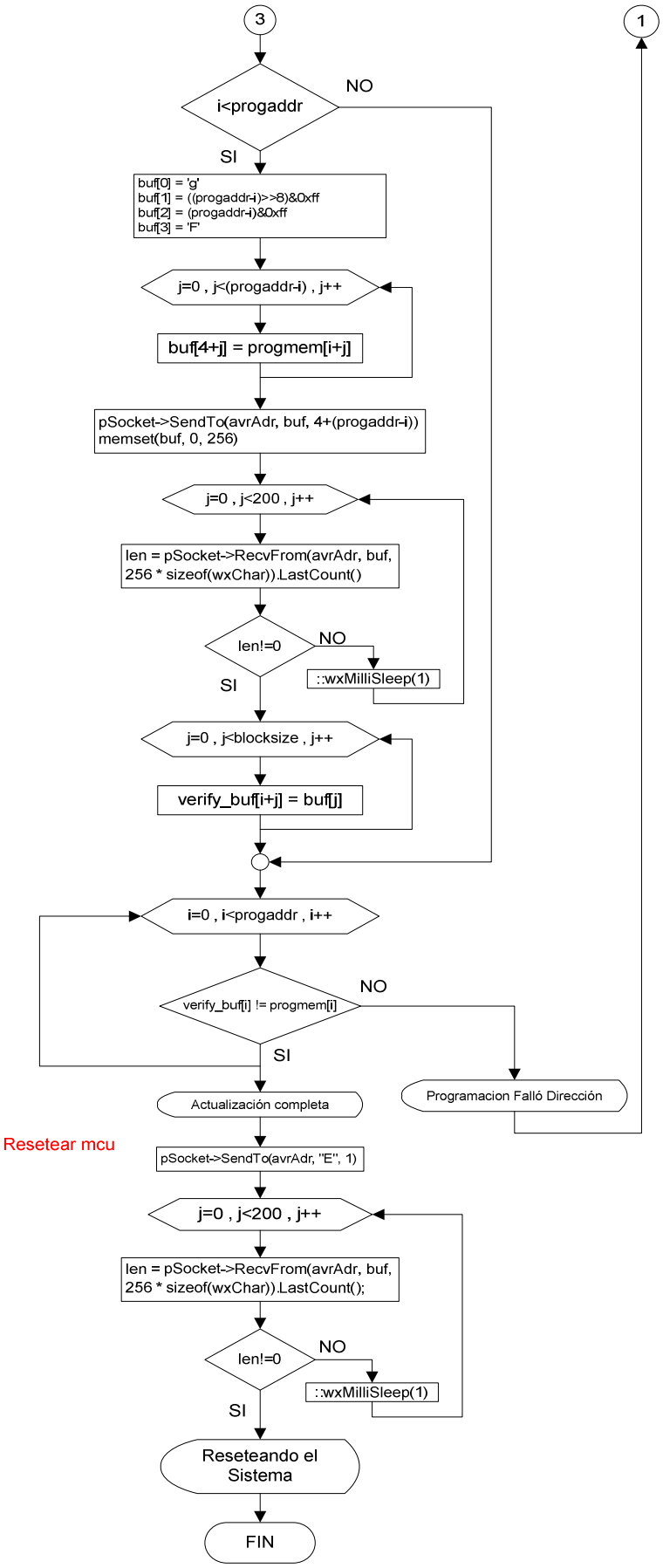
FUNCIONES: OnOpenHex, OnOpenEep, SendSL, OnConnect, OnSocketEvent, OnSetIp, OnCheckBoxLed1, OnCheckBoxLed2, HexConverter, OnUpgrade, OnRefresh

LECTURA DE IP Y PUERTO DE AVRM




ENVIAR ARCHIVO.HEX A AVR E-SP
INICIO
wxIPV4address avrAdr
avrAdr.AnyAddress()
wxString s
wxUint32 i, j, len, blocksize
wxChar *buf = new wxChar[256]
wxChar *verify_buf = new wxChar[57344]
memset(buf, 0, 256)
TextCtrlOpenHex->GetValue()=="Nuevo Programa .hex"
SI
Por favor abra un nuevo archivo .hex
NO
avrAdr.Service(TextCtrlAvrPort->GetValue())
avrAdr.Hostname(TextCtrlAvrCurrentIp->GetValue())
Reseteando el sistema
Resetear AVR E-SP
pSocket->SendTo(avrAdr, "RS", 4)
i=0 , i<1000, i++
len = pSocket->RecvFrom(avrAdr, buf, 256 * sizeof(wxChar)).LastCount()
len!=0
NO
::wxMilliSleep(1)
SI
buf[0]=='Y'
SI
Ingresando a bootloader
::wxMilliSleep(1000)
NO
No se puede resetear el sistema, Ingresando a bootloader
Ingreso a bootloader
pSocket->SendTo(avrAdr, "P", 1)
memset(buf, 0, 256)
i=0 , i<1000 , i++
len = pSocket->RecvFrom(avrAdr, buf, 256 * sizeof(wxChar)).LastCount()
len!=0
NO
::wxMilliSleep(1)
SI
buf[0] != '\r'
SI
Programacion Fallida
NO
Obteniendo tamaño de bloque
pSocket->SendTo(avrAdr, "b", 1)
memset(buf, 0, 256)
i=0, i<1000 , i++
len = pSocket->RecvFrom(avrAdr, buf, 256 * sizeof(wxChar)).LastCount()
len!=0
NO
::wxMilliSleep(1)
SI
blocksize = (((buf[1]&0xff)<<8)|(buf[2]&0xff))&0xffff
El tamaño de bloque es: blocksize
2
1


2
Borrar chip
Borrando Chip
pSocket->SendTo(avrAdr, "e", 1)
memset(buf, 0, 256)
i=0 , i<1000, i++
len = pSocket->RecvFrom(avrAdr, buf, 256 * sizeof(wxChar)).LastCount()
len!=0
NO
::wxMilliSleep(1)
SI
buf[0] != '\r'
SI
Fallo
NO
Programando flash
seteando direcciones
buf[0] = 'A'
buf[1] = 0
buf[2] = 0
pSocket->SendTo(avrAdr, buf, 3)
memset(buf, 0, 256)
i=0 , i<1000, i++
len = pSocket->RecvFrom(avrAdr, buf, 256 * sizeof(wxChar)).LastCount()
len!=0
NO
::wxMilliSleep(1)
buf[0] != '\r'
SI
Seteo de dirección falló
NO
memset(buf, 0, 256)
i=0 , i<progaddr , i+=blocksize
SI
(progaddr-i)<blocksize
NO
buf[0] = 'B'
buf[1] = (blocksize>>8)&0xff
buf[2] = blocksize&0xff
buf[3] = 'F'
j=0 , j<blocksize , j++
buf[4+j] = progmem[i+j]
pSocket->SendTo(avrAdr, buf, 4+blocksize)
memset(buf, 0, 256)
j=0 , j<200 , j++
len = pSocket->RecvFrom(avrAdr, buf, 256 * sizeof(wxChar)).LastCount()
len!=0
NO
::wxMilliSleep(1)
SI
3
1
1


3
1
i<progaddr
NO
SI
buf[0] = 'g'
buf[1] = ((progaddr-i)>>8)&0xff
buf[2] = (progaddr-i)&0xff
buf[3] = 'F'
j=0 , j<(progaddr-i) , j++
buf[4+j] = progmem[i+j]
pSocket->SendTo(avrAdr, buf, 4+(progaddr-i))
memset(buf, 0, 256)
j=0 , j<200 , j++
len = pSocket->RecvFrom(avrAdr, buf,
256 * sizeof(wxChar)).LastCount()
len!=0
NO
::wxMilliSleep(1)
SI
j=0 , j<blocksize , j++
verify_buf[i+j] = buf[j]
i=0 , i<progaddr , i++
verify_buf[i] != progmem[i]
NO
SI
Actualización completa
Programacion Falló Dirección
Resetear mcu
pSocket->SendTo(avrAdr, "E", 1)
j=0 , j<200 , j++
len = pSocket->RecvFrom(avrAdr, buf,
256 * sizeof(wxChar)).LastCount();
len!=0
NO
::wxMilliSleep(1)
SI
Reseteando el
Sistema
FIN

ANEXO 2: Capturas de tráfico con el Analizador de protocolos

Protocolo ARP (Detalles de la Trama)

Frame 1 (42 bytes on wire, 42 bytes captured)
Arrival Time: Feb 12, 2009 08:58:02.080109000
[Time delta from previous packet: 0.000000000 seconds]
[Time since reference or first frame: 0.000000000 seconds]
Frame Number: 1
Packet Length: 42 bytes
Capture Length: 42 bytes
[Frame is marked: False]
[Protocols in frame: eth:arp]
[Coloring Rule Name: ARP]
[Coloring Rule String: arp]
Ethernet II, Src: Elitegro_84:1f:95 (00:14:2a:84:1f:95), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Destination: Broadcast (ff:ff:ff:ff:ff:ff)
Source: Elitegro_84:1f:95 (00:14:2a:84:1f:95)
Type: ARP (0x0806)
Address Resolution Protocol (request)
Hardware type: Ethernet (0x0001)
Protocol type: IP (0x0800)
Hardware size: 6
Protocol size: 4
Opcode: request (0x0001)
Sender MAC address: Elitegro_84:1f:95 (00:14:2a:84:1f:95)
Sender IP address: 10.1.1.76 (10.1.1.76)
Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
Target IP address: 10.1.1.1 (10.1.1.1)

Protocolo ICMP

Peticion ICMP (Detalles de la trama)

Frame 11 (74 bytes on wire, 74 bytes captured)
Arrival Time: Feb 12, 2009 09:00:54.349925000
[Time delta from previous packet: 0.000007000 seconds]
[Time since reference or first frame: 172.269816000 seconds]
Frame Number: 11
Packet Length: 74 bytes
Capture Length: 74 bytes
[Frame is marked: False]
[Protocols in frame: eth:ip:icmp:data]
[Coloring Rule Name: ICMP]
[Coloring Rule String: icmp]
Ethernet II, Src: Elitegro_84:1f:95 (00:14:2a:84:1f:95), Dst: 41:56:52:50:4f:52
(41:56:52:50:4f:52)
Destination: 41:56:52:50:4f:52 (41:56:52:50:4f:52)
Source: Elitegro_84:1f:95 (00:14:2a:84:1f:95)
Type: IP (0x0800)
Internet Protocol, Src: 10.1.1.76 (10.1.1.76), Dst: 10.1.1.1 (10.1.1.1)
Version: 4
Header length: 20 bytes
Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
Total Length: 60
Identification: 0x00c7 (199)

Flags: 0x00
Fragment offset: 0
Time to live: 128
Protocol: ICMP (0x01)
Header checksum: 0x23ac [correct]
Source: 10.1.1.76 (10.1.1.76)
Destination: 10.1.1.1 (10.1.1.1)
Internet Control Message Protocol
Type: 8 (Echo (ping) request)
Code: 0
Checksum: 0x485c [correct]
Identifier: 0x0400
Sequence number: 256 (0x0100)
Data (32 bytes)

Echo reply (Detalles de la Trama)

Frame 12 (74 bytes on wire, 74 bytes captured)
Arrival Time: Feb 12, 2009 09:00:54.354076000
[Time delta from previous packet: 0.004151000 seconds]
[Time since reference or first frame: 172.273967000 seconds]
Frame Number: 12
Packet Length: 74 bytes
Capture Length: 74 bytes
[Frame is marked: False]
[Protocols in frame: eth:ip:icmp:data]
[Coloring Rule Name: ICMP]
[Coloring Rule String: icmp]
Ethernet II, Src: 41:56:52:50:4f:52 (41:56:52:50:4f:52), Dst: Elitegro_84:1f:95 (00:14:2a:84:1f:95)
Destination: Elitegro_84:1f:95 (00:14:2a:84:1f:95)
Source: 41:56:52:50:4f:52 (41:56:52:50:4f:52)
Type: IP (0x0800)
Internet Protocol, Src: 10.1.1.1 (10.1.1.1), Dst: 10.1.1.76 (10.1.1.76)
Version: 4
Header length: 20 bytes
Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
Total Length: 60
Identification: 0x0003 (3)
Flags: 0x00
Fragment offset: 0
Time to live: 128
Protocol: ICMP (0x01)
Header checksum: 0x2470 [correct]
Source: 10.1.1.1 (10.1.1.1)
Destination: 10.1.1.76 (10.1.1.76)
Internet Control Message Protocol
Type: 0 (Echo (ping) reply)
Code: 0
Checksum: 0x505c [correct]
Identifier: 0x0400
Sequence number: 256 (0x0100)
Data (32 bytes)

Protocolo UDP (Detalles de la primera y última trama)

Frame 1 (46 bytes on wire, 46 bytes captured)
    Arrival Time: Feb 12, 2009 09:32:07.476812000
    [Time delta from previous packet: 0.000000000 seconds]
    [Time since reference or first frame: 0.000000000 seconds]
    Frame Number: 1
    Packet Length: 46 bytes
    Capture Length: 46 bytes
    [Frame is marked: False]
    [Protocols in frame: eth:ip:udp:data]
    [Coloring Rule Name: UDP]
    [Coloring Rule String: udp]
Ethernet II, Src: Elitegro_84:1f:95 (00:14:2a:84:1f:95), Dst: 41:56:52:50:4f:52 (41:56:52:50:4f:52)
    Destination: 41:56:52:50:4f:52 (41:56:52:50:4f:52)
    Source: Elitegro_84:1f:95 (00:14:2a:84:1f:95)
    Type: IP (0x0800)
Internet Protocol, Src: 10.1.1.76 (10.1.1.76), Dst: 10.1.1.1 (10.1.1.1)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 32
    Identification: 0x0285 (645)
    Flags: 0x00
    Fragment offset: 0
    Time to live: 128
    Protocol: UDP (0x11)
    Header checksum: 0x21fa [correct]
    Source: 10.1.1.76 (10.1.1.76)
    Destination: 10.1.1.1 (10.1.1.1)
User Datagram Protocol, Src Port: 3001 (3001), Dst Port: 3000 (3000)
    Source port: 3001 (3001)
    Destination port: 3000 (3000)
    Length: 12
    Checksum: 0x72b9 [correct]
Data (4 bytes)

Ultima Trama

Frame 427 (60 bytes on wire, 60 bytes captured)
    Arrival Time: Feb 12, 2009 09:32:11.391261000
    [Time delta from previous packet: 0.000969000 seconds]
    [Time since reference or first frame: 3.914449000 seconds]
    Frame Number: 427
    Packet Length: 60 bytes
    Capture Length: 60 bytes
    [Frame is marked: False]
    [Protocols in frame: eth:ip:udp:data]
    [Coloring Rule Name: UDP]
    [Coloring Rule String: udp]
Ethernet II, Src: 41:56:52:50:4f:52 (41:56:52:50:4f:52), Dst: Elitegro_84:1f:95 (00:14:2a:84:1f:95)
    Destination: Elitegro_84:1f:95 (00:14:2a:84:1f:95)
    Source: 41:56:52:50:4f:52 (41:56:52:50:4f:52)
    Type: IP (0x0800)
    Trailer: 00000000000000
Internet Protocol, Src: 10.1.1.1 (10.1.1.1), Dst: 10.1.1.76 (10.1.1.76)

Version: 4
Header length: 20 bytes
Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
Total Length: 39
Identification: 0x035a (858)
Flags: 0x00
Fragment offset: 0
Time to live: 128
Protocol: UDP (0x11)
Header checksum: 0x211e [correct]
Source: 10.1.1.1 (10.1.1.1)
Destination: 10.1.1.76 (10.1.1.76)
User Datagram Protocol, Src Port: 3000 (3000), Dst Port: 3001 (3001)
Source port: 3000 (3000)
Destination port: 3001 (3001)
Length: 19
Checksum: 0xbab9 [correct]
Data (11 bytes)

BIBLIOGRAFIA

Libros

(1) ATMEL CORP. ATmega32 Data Sheet. Estados Unidos: Atmel, 2007. pp. 2-274.

(2) MICROCHIP TECHNOLOGY CORP. ENC28J60 Data Sheet. Estados Unidos: Microchip Technology, 2008. pp. 1-98.

(3) STALLING, W. Comunicaciones y Redes de Computadores. Traducido del inglés por Días, J. 7ma ed. Madrid: Pearson, 2004. pp 587-805.

(4) GARCÍA, B. Guía Completa de Protocolos de Telecomunicaciones. Traducido del inglés por Vuelapluma. 4ta. ed. España: McGRAW-HILL, 2002. pp. 382-481.

(5) PARDUE, J. C Programming for Microcontrollers. Knoxville: Smiley Micros, 2005. pp. 17-271.

(6) AXELSON, J. Embedded Ethernet and internet complete. Madison: Lakeview Research LLC, 2003. pp. 91-242.

(7) INSAM, E. TCP/IP Embedded Internet Applications. Estados Unidos: Newnes, 2003. pp. 5-28.

(8) LABROSSE, J. Embedded software. Estados Unidos: Newnes, 2008. pp. 241-332.

(9) MICROCHIP TECHNOLOGY CORP. AVR109: Self Programming. Estados Unidos: Microchip Technology, 2004. pp. 1 - 11.

(10) GADRE, D. Programming and Customizing the Avr Microcontroller. Estados Unidos: McGraw-Hill, 2001. pp. 29-108.

(11) BARRETT, F. and PACK, D. Atmel AVR Microcontroller Primer: Programming and Interfacing. Lecture #15. Southern Methodist University: Morgan & Claypool, 2008. pp. 114 – 126.

(12) SMART, J. and HOCK, K. Cross-Platform GUI Programming with wxWidgets. Versión 1.0. Estados Unidos: Pearson Education, 2006. pp. 1-744.

Bibliografía Internet

(13) Implementación de TCP/IP sobre ATmega88

http://tuxgraphics.org/electronics/200611/article06111.shtml

20081210

(14) Internetworking con microprocesadores PIC

http://www.ljcv.net/projects/pic10t/index.html

20080908

(15) Networking Embebido con ENC28J60

http://electronicfr.com/index.php/Microcontrollers-and-ethernet/

20082508

(16) Estándares RFC para la comunicación de Protocolos

http://www.ietf.org/

20081911

(17) Código de Ejemplo para el manejo del ENC28J60

www.microchip.com/Ethernet

20080510