INSTITUTO TECNOLOGICO DE COSTA RICA

Escuela de Ingeniería en Electrónica

Instituto Costarricense de Electricidad

ICE

Desarrollo e implementación de puente para traducción de protocolo MODBUS a MODBUS TCP y viceversa

Informe de Proyecto de Graduación para optar por el Grado de Bachiller en Ingeniería Electrónica

César Tijerino Cerna

Cartago, Junio del 2002

Resumen

En el Centro de Servicio, Investigación y Desarrollo del Instituto Costarricense de Electricidad (I.C.E.), se realizan diferentes tipos de proyectos, orientados al mejoramiento de los servicios que presta esta importante institución al país.

Uno de los proyectos que el Centro ha empezado a desarrollar es el de colocar unidades de transmisión remota (UTR) en diferentes puntos de los embalses del I.C.E., para la medición de algunas variables físicas. Estas unidades son parte de un sistema SCADA y cada una comunica sus datos por medio del protocolo para redes industriales MODBUS.

El monitoreo de las unidades de transmisión remota se puede realizar conectándolas a una red de datos tipo Ethernet, usando una variable del protocolo MODBUS, llamada MODBUS TCP, pero la conexión de los aparatos no puede hacerse directamente, es necesario construir un traductor de protocolos. Según los requerimientos del Centro, el aparato debía ser programable y su costo final no debía sobrepasar los US$700; además, se debía programar en C.

Para realizar la conexión, se implementó un traductor de protocolos MODBUS a MODBUS TCP, usando una tarjeta de desarrollo tipo Rabbit 2000 TCP/IP. Se desarrolló también el programa de traducción de protocolos y el manejo de una conexión TCP/IP

Palabras claves: MODBUS, TCP, Rabbit, UTR, puente, Ethernet, traductor.

Abstract

In the Centro de Servicio, Investigación y Desarrollo of the Instituto Costarricense de Electricidad (I.C.E.), different types of projects are made, for the improvement of the services given by this institution.

One of the projects that the Center has begun to develop is to place some remote transmission units (RTU) in different points around the dams of the I.C.E., for the measurement of some physical variables. These units are part of a SCADA system and each one communicates its data using the protocol for industrial networks MODBUS.

The monitoring of these units can be made connecting them to an Ethernet network, using a variation of the protocol MODBUS, MODBUS TCP, but the connection of the devices cannot be made directly, is necessary to develop a translator of protocols. According to the requirements of the Center, the translator had to be programmable, its final cost did not have to exceed the USS700 and the translatiing application had to be made in C.

In order to make the connection, a translator of protocols MODBUS to MODBUS TCP was implemented, using a Rabbit 2000 TCP/IP development card. A program for translation of protocols and the handling of one TCP/IP socket was also implemented.

Keywords: MODBUS, TCP, Rabbit, bridge, Ethernet, translator.

INDICE GENERAL

INDICE DE FIGURAS

INDICE DE TABLAS

CAPÍTULO 1
INTRODUCCIÓN

1.1 Descripción de la Empresa

1.1.1 Descripción General

El Instituto Costarricense de Electricidad (ICE), es una institución autónoma del Estado, con personería jurídica propia, que se financia con fondos propios y externos a través de la venta de servicios de energía y telecomunicaciones.

El Instituto Costarricense de Electricidad (ICE) fue creado por el Decreto - Ley No.449 del 8 de abril de 1949 como una institución autónoma, con personalidad jurídica y patrimonio propio. Está dotado de plena autonomía e independencia administrativa, técnica y financiera. Al ICE le corresponde, por medio de sus empresas, desarrollar, ejecutar, producir y comercializar todo tipo de servicios públicos de electricidad y telecomunicaciones, así como actividades o servicios complementarios a estos.

Como objetivos primarios el ICE debe desarrollar, de manera sostenible, las fuentes productoras de energía existentes en el país y prestar el servicio de electricidad. A su vez, se encarga de desarrollar y prestar los servicios de telecomunicaciones, con el fin de promover el mayor bienestar de los habitantes del país y fortalecer la economía nacional.

El ICE, constituido por los sectores ICE – Electricidad e ICE – Telecomunicaciones y sus empresas, la Compañía Nacional de Fuerza y Luz (CNFL) y Radiográfica Costarricense S.A. (RACSA) es un Grupo que se ha

caracterizado por una gestión de desarrollo de clase internacional tendiente a satisfacer las necesidades evolutivas que plantean los clientes y un entorno altamente competitivo.

El Consejo Directivo del ICE esta constituido por siete miembros, los cuales son nombrados directamente por el Consejo de Gobierno. De este Consejo Directivo se nombra el Presidente Ejecutivo, que es el máximo jerarca en la estructura organizacional del ICE; y también se nombre un Vicepresidente y un Secretario.

El Consejo Directivo se encarga de nombrar a un Gerente General e internamente se elige un grupo de apoyo, lo cual se constituye en la Gerencia General del ICE.

La Gerencia General se divide en tres subgerencias: Electricidad, Gestión Administrativa y Telecomunicaciones. Cada una de estas subgerencias está integrada por Áreas y Unidades Estratégicas de Negocios (UEN) cuyos Directores son nombrados a nivel interno y por concurso.

1.1.2 Descripción del Departamento

El proyecto se desarrolló en la Unidad Estratégica de Negocio, Proyectos y Servicios Asociados; propiamente en el Centro de Servicio, Investigación y Desarrollo.

La misión de esta unidad es desarrollar y comercializar proyectos con calidad de clase mundial para continuar impulsando el desarrollo eléctrico, la especialización de su recurso humano y los productos y servicios asociados, satisfaciendo las necesidades y expectativas. Consta de varios laboratorios, entre los que están:

a. Laboratorio de Circuitos Impresos

b. Laboratorio de simulación de sistemas de potencia.

c. Laboratorio de corrosión.

El Centro de Servicio, Investigación y Desarrollo se dedica a investigar nuevas tecnologías que puedan aplicarse al sector de energía. Esta sección busca el desarrollo de proyectos particularmente en esta área.

En el departamento laboran aproximadamente 50 personas, de las cuales casi todos son ingenieros y personal técnico.

1.2 Definición del problema y su importancia

Uno de los protocolos de comunicación usados al nivel industrial para la comunicación entre dispositivos diversos (como, por ejemplo, PLCs) es el denominado *MODBUS*®, el cual se ha convertido en estándar para las industrias. Este protocolo es una estructura de mensaje, ampliamente usada para establecer comunicaciones maestro-esclavo entre aparatos “inteligentes”.

En el Centro de Servicio, Investigación y Desarrollo del Instituto Costarricense de Electricidad, se pensó en diseñar y construir un traductor de protocolo *MODBUS* para una unidad de transmisión remota (UTR) de un sistema *SCADA*, el cual se encuentra casi terminado en este momento (ver figura 1.1). Mediante ese sistema se transmitirá información de diversos instrumentos ubicados en las plantas de generación eléctrica y directamente en las redes de distribución eléctrica, los cuales proveerán al personal del ICE de datos acerca de diversos procesos físicos. La idea que se tiene ahora es la de poder conectar esas terminales a una red que utilice el protocolo *TCP/IP*, para agilizar el monitoreo de estas, poder centralizarlo e inclusive poder hacerlo a través de internet.

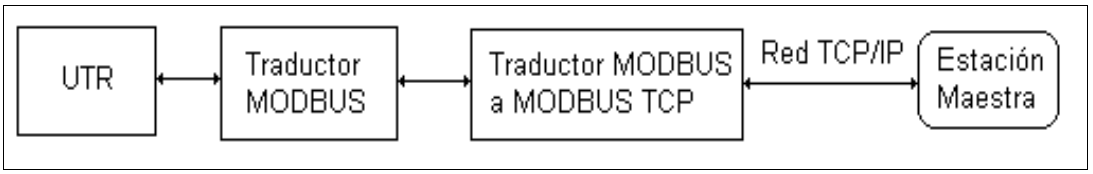


Figura 1.1 Modelo del sistema SCADA que se está desarrollando en el Centro de Sevicio, Investigación y Desarrollo del ICE para el monitoreo de variables físicas en plantas de generación de energía eléctrica y redes de distribución.

En el mercado internacional, existe una variante del protocolo *MODBUS* llamada *MODBUS TCP*, la cual hace posible la interconexión entre dispositivos *MODBUS* y redes que usan TCP/IP, lo cual ha posibilitado el monitoreo de dispositivos por medio de internet. Dado que *MODBUS TCP* se deriva del *MODBUS* original, la estructura de los mensajes, las alertas de errores y otras características del protocolo resultan similares, pero no idénticas, es así que para poder conectar un aparato *MODBUS* a una red Ethernet (por ejemplo) es necesario un puente que interprete los protocolos. Dichos puentes se pueden adquirir en el mercado de aparatos electrónicos orientados al control industrial, pero para una empresa que requiera varios de estos puentes la inversión resulta onerosa, pues cada uno de estos cuesta alrededor de US$700.

Dado que las especificaciones de *MODBUS TCP* son públicas, es posible el desarrollo de un puente de interpretación por parte del usuario. Esa posibilidad es la que hace surgir la idea principal de este proyecto, la cual es diseñar, construir y probar un puente de interpretación de protocolo *MODBUS* a *MODBUS TCP* por parte de un ingeniero en electrónica (o, en este caso, como proyecto de graduación de un estudiante de esta disciplina), que resulte a su vez eficiente y más económico que uno de tipo comercial.

El producto final de este proyecto es un prototipo de puente que le permitirá realizar conectar una UTR perteneciente a un sistema *SCADA* y que maneja el protocolo *MODBUS,* a través de una red que use el protocolo *TCP/IP* (o el protocolo

MODBUS TCP) y cumplan con el estándar internacional *IEEE 802.3*, como, por ejemplo, redes LAN.

1.3 Objetivos

1.3.1 Objetivo General

Implementar en un plazo de 16 semanas un aparato capaz de interpretar el protocolo de comunicación MODBUS y convertirlo a MODBUS TCP basándose en un dispositivo programable, cuyo costo de fabricación para el modelo operacional no sea superior a $700.

1.3.2 Objetivos Específicos

a. Realizar una investigación extensa acerca de los protocolos de comunicación MODBUS y MODBUS TCP.

b. Seleccionar el kit de desarrollo o tarjeta que mejor se adapte a las necesidades del proyecto.

c. Diseñar el programa que requiere el microcontrolador para realizar la traducción de protocolos MODBUS a MODBUS TCP.

d. Diseñar los programas de aplicación necesarios para la prueba de funcionamiento del traductor de protocolos MODBUS a MODBUS TCP.

e. Depurar los programas diseñados, tanto los de aplicación como el de traducción de protocolos.

f. Comprobar el funcionamiento integral del sistema.

g. Realizar un informe detallado de todo lo concerniente a la implementación de todo el sistema.

CAPÍTULO 2
ANTECEDENTES

2.1 Estudio del problema a resolver

En el Centro de Servicio, Investigación y Desarrollo del Instituto Costarricense de Electricidad se desarrollan proyectos para todos los departamentos del Instituto que requieran de soluciones integrales a diversos problemas que puedan resolverse por medio de la electrónica y desarrollar proyectos que contribuyan a la mejora de los servicios de generación y distribución de energía eléctrica y telecomunicaciones.

Uno de los problemas que se presentaron al Centro fue el de diseñar un sistema SCADA para la medición remota de diversas variables físicas en los embalses del ICE. El proyecto en su primera parte consistió en el diseño del sistema, luego se procedió a implementar un prototipo (el que está en fase de pruebas) al que se le adaptó un traductor de *SCADA* a *MODBUS*, protocolo industrial que se ha convertido en un estándar entre los encargados del diseño y la construcción de redes industriales. El nuevo paso para lograr una mayor flexibilidad del sistema es conectarlo a una red LAN, para expandir su capacidad de enviar datos.

Como se había optado por usar el protocolo *MODBUS*, y para no complicar la implementación de una conexión a redes LAN, se pensó inicialmente en la compra de un traductor de protocolos *MODBUS* a *MODBUS TCP*, que es la variante que provee el grupo que desarrolló *MODBUS* y que específicamente es usada para acceder a sistemas *MODBUS* por medio de mensajes TCP/IP. El principal problema de comprar estos dispositivos comerciales es el precio, pues pese a ser un aparato garantizado, su alto precio (alrededor de US$700 aproximadamente, cada uno) es un inconveniente si se piensa en adquirirlo en grandes cantidades.

Dado lo anterior, se pensó en el diseño e implementación de un traductor similar, pues de esta forma se garantizaría no sólo el correcto funcionamiento de este, sino que el desarrollo abarataría los costos de producción de los sistemas que lo requieran.

En el medio industrial siempre se han necesitado sistemas de recolección de datos fiables y rápidos, para guiar las pautas que mejorarán la calidad de los procesos y la eficiencia de cada uno de estos.

Los ingenieros del Instituto Costarricense de Electricidad saben muy bien que la constante recolección de datos importantes en cada proceso en el que se involucra la institución, tanto en el sector energía como en el de telecomunicaciones, y su posterior análisis, garantiza la calidad de sus servicios y la eventual mejora en estos. Es por eso que en el I.C.E. se trata de disponer de esos datos rápidamente y de manera acertada.

Durante la realización de este proyecto se tomó en cuenta una serie de puntos sobre los cuales se trabajó. Uno de los más importantes fue es de escoger de manera adecuada el hardware a usar para el desarrollo del prototipo de traductor.

Para el desarrollo del proyecto, se decidió adquirir una pieza de hardware ya construida, basada en un dispositivo programable de buena capacidad de memoria de programa (suficiente como para guardar un programa de no menos de 5kb), que hiciera más instrucciones por segundo que los usados anteriormente en el Centro, con al menos un puerto RS-232 listo para usar y que integrara tanto al microcontrolador como a una interfase que permitiera conectar el aparato a una red Ethernet. Estos eran los requerimientos de hardware mínimos para el proyecto. Todo esto es necesario porque la interfase física del protocolo *MODBUS* es la RS-232, y para que se cumpliera con el requisito de conectar dispositivos MODBUS a una red de datos tipo *Ethernet*, se requiere de una plataforma adecuada y que cumpla con lo establecido en el estándar *IEEE 802.3*, como se muestra en la figura .2.1.

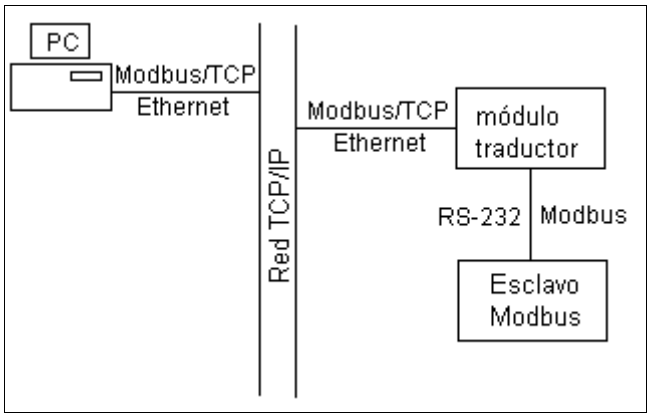


Figura 2.1 Esquema de conexiones de los dispositivos involucrados en el proyecto

Para la parte del software, la estructura principal del programa debió cumplir con aspectos tales como el adecuado manejo de las conexiones TCP/IP (para la comunicación con *MODBUS TCP*) y la conversión, o traducción, de protocolos; además, el programa debió de manejar de buena manera la transmisión de los datos hacia cada una de las interfases físicas (ethernet y serial RS-232).

Los siguientes son los puntos fundamentales que se tomaron en cuenta para el desarrollo del programa de traducción:

a. El protocolo *MODBUS TCP* es simplemente una adaptación del *MODBUS* original, que pone los mensajes de *MODBUS* sobre el segmento de datos de un mensaje TCP/IP normal. *MODBUS TCP* usa la fiabilidad de TCP/IP para garantizar que los datos lleguen a su destino adecuadamente. Por lo anterior, *MODBUS TCP* no hace uso del *CRC*[1], necesario en los mensajes *MODBUS* como método para la verificación de errores en los mensajes durante la comunicación.

[1] Cyclical Redundacy Check (Chequeo cíclico de redundancia). Valor de 2 bytes calculado por el aparato transmisor del mensaje, el que luego es añadido al final del mensaje. El dispositivo receptor recalcula el CRC durante la recepción y lo compara con el valor CRC recibido. Si ambos valores son diferentes, un error ocurrió en la comunicación.

b. *MODBUS TCP* requiere de un encabezado de 6 bytes al comienzo de un mensaje. Este encabezado consta de:

   1. Dos bytes (byte 0 y byte 1) de identificador de transacción (*transaction identifier*): usualmente cero.

   2. Dos bytes (byte 2 y byte 3) de identificador de protocolo (*protocol identifier*): también, ambos con valor cero.

   3. Campo de longitud (*length field*): se usa para saber cuántos bytes, a partir de aquí, hay en el mensaje. Consta de dos bytes (byte 4 y byte 5). El primer byte es la parte alta de la longitud (usualmente con valor cero, pues los mensajes *MODBUS* son de longitud igual o menor a 256 bytes), el segundo es la parte baja del valor.

      Todos estos valores deben de guardarse cuando un mensaje *MODBUS TCP* llega al traductor, para ser repuestos en el mensaje luego de recibir la respuesta del dispositivo que se comunica con *MODBUS.*

c. Para *MODBUS*, la generación del CRC es fundamental, por lo tanto el programa debe hacerlo cuando sea pertinente.

2.2 Requerimientos de la Empresa

El Centro de Servicio, Investigación y Desarrollo buscó con este proyecto el desarrollo e implementación de un “puente” entre dos protocolos de comunicación para redes industriales que, aunque no son muy diferentes entre sí (dado que *MODBUS TCP* es una variante de *MODBUS*), requieren de cierto nivel de interpretación y reestructuración de los mensajes enviados de una estación que use uno de los protocolos, a otra que utilice el otro.

Dicho dispositivo debía ser programable. Además, se debía usar el lenguaje de programación C, para que la interpretación del código final no fuera difícil (como lo es cuando se habla del lenguaje ensamblador, por ejemplo, pues su interpretación y uso dependen en mucho del grado de conocimiento que el programador tenga de la arquitectura de un determinado microcontrolador o microprocesador).

El programa de traducción debía manejar al menos una conexión TCP/IP de manera adecuada (manejo de confirmación de conexión, parada de conexión, transmisión y recepción de datos a través de un puerto designado por el usuario, etc.), además, administrar la transmisión y recepción de datos enviados por una UTR que se comunica con *MODBUS*, a través de un puerto serie.

El costo de producción del aparato, en su versión final, no debía de sobrepasar los US$700, pues sería mejor comprar uno de los tantos que existen hoy en día en el mercado mundial.

Las plataformas físicas para el ingreso de los datos debían ser, para los mensajes *MODBUS*, RS-232; para los mensajes *MODBUS TCP*, 10BaseT (conector RJ-45).

Dado el resultado final del proyecto, el prototipo presentado cumple con todas estas espectativas.

2.3 Solución propuesta

La solución que el I.C.E. propuso fue el desarrollo del hardware y software necesarios para realizar la traducción del protocolo *MODBUS* a *MODBUS TCP* y viceversa. El resultado debía de hacer la traducción adecuadamente y el aparato debía ser programable. Además, el costo final de un traductor no debía sobrepasar los US$700. El diagrama de bloques de la solución propuesta se muestra en la figura 2.2.

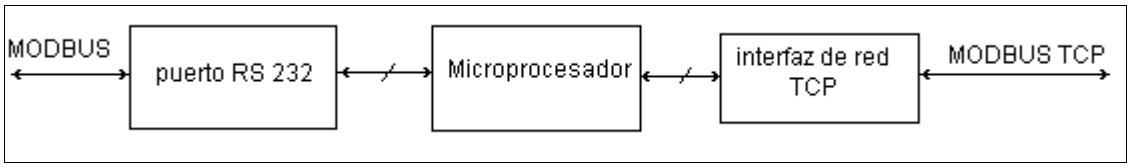


Figura 2.2 Diagrama de bloques de la solución propuesta.

Para cumplir con esto, se realizó un estudio comparativo de varias tarjetas en el mercado, a través de internet. Como resultado de dicha búsqueda, se llegó a la conclusión inicial de adquirir una tarjeta modelo *Adapt11C24ETFX* de la empresa *Technological Arts*[2].

Las características de esta tarjeta parecían, inicialmente, buenas para el proyecto. Luego de que se empezó a programar todo lo concerniente al manejo de la interfase ethernet (en este caso, el chip *CS8900A* de la marca *Cirrus Logic*) y las primeras rutinas para el manejo de TCP/IP, se notó que la cantidad de memoria de programa del microcontrolador en la tarjeta, un *Motorola* modelo *68HC811E2* (256 bytes de RAM y 2kb de EEPROM) no iba a ser suficiente. La empresa *Technological Arts* provee de expansiones de memoria para varios de sus productos, pero para este no, por lo tanto, la única forma de agrandar la memoria de programa era cambiando el microcontrolador por otro de la misma serie que poseyera más memoria.

El microcontrolador de la serie HC11 de Motorola con más memoria de programa es el *68HC711E20*, con 20kb. Este microcontrolador tiene la desventaja de ser del tipo *OTP*[3], entonces, para probar las rutinas programadas se debía usar un simulador o un emulador. Ambas opciones son bastante costosas, además, es prácticamente imposible saber el comportamiento global del sistema, pues, aunque se sepan los valores que se ponen en los registros de salida del puerto serie, o

[2] Dirección en internet: **www.technologicalarts.com**
[3] One Time Programming (Dispositivo programable una vez). Cuando su memoria ROM se quema, no se puede volver a reprogramar.

dentro de los registros de datos para enviar por la interfase Ethernet, no se sabe si por algún otro factor (falta de sincronía, errores de lógica del programa, etc.) el programa final iba a fallar, lo que en términos de la programación en sí de la memoria del microcontrolador significa la pérdida del mismo, al no poder reprogramarse.

El costo de cada microcontrolador es de alrededor de US$18, lo que no da margen para errores en la programación, tan comunes durante la realización de cada programa.

Por todo lo anterior, se decidió hacer un cambio de hardware, para cumplir con los requerimientos del proyecto. Este nuevo hardware elegido es el *kit de desarrollo para TCP/IP Rabbit 2000,* el que incluye, una tarjeta basada en el microprocesador *Rabbit 2000* de la empresa *Rabbit Semiconductors*, que se basa en la arquitectura del famoso Z80; un controlador Ethernet *RTL8019AS* de *Realtek Semiconductor Co.*, además de 128kb de memoria RAM y 512kb de memoria Flash.

El módulo trabaja a 18.432 MHz y se puede programar en el lenguaje C. El software que provee el fabricante, *Dynamic C Special Edition ver. 7.20TSE*, incluye varias librerías, entre ellas una que hace el manejo completo de los protocolos TCP/IP, lo que representa una gran ventaja en lo que respecta al tiempo de desarrollo con respecto a la tarjeta elegida en un principio, pues para esta se debía trabajar en las rutinas para TCP/IP y para el manejo del controlador de Ethernet *CS8900A*, aparte de las rutinas de traducción de los protocolos *MODBUS*. Con esta tarjeta se cubrió lo que corresponde a la elección del hardware, pues cumplió con creces a las espectativas del proyecto.

Una vez elegido el hardware, se pasó directamente a la programación de las rutinas de traducción de protocolos, pues la parte de manejo de TCP/IP y de interfases físicas (Ethernet y RS-232) ya estaba lista. Lo que se necesitaba era un

programa que al iniciarse, alistara una conexión pasiva[4] al puerto 502 (designado para las transacciones de mensajes y datos *MODBUS TCP*), esperara a que una unidad maestra *MODBUS TCP* enviara datos a la dirección IP del traductor, tradujera el mensaje al formato *MODBUS RTU* o *MODBUS ASCII*, esperara luego la respuesta del esclavo al que se le envió el mensaje, recibiera el mensaje de respuesta, tradujera de vuelta la respuesta y la reenviara a la estación maestra. El formato habitual de las comunicaciones *MODBUS* es el bidireccional *half duplex*[5].

Este programa se almacenó en la memoria Flash de la tarjeta *Rabbit 2000*.

[4] Su característica principal es que, una vez que se echa a andar la máquina de estados que maneja el estado de una conexión TCP/IP, esperar á que algún usuario externo escriba datos en el puerto apuntado por la conexión. Por eso, a los puertos asignados a este tipo de conexiones se les llama “puertos de escucha” (listen ports).

[5] Con dos dispositivos que pueden transmitir y recibir datos. Durante la transmisión por parte de uno de los dos, el otro sólo puede recibir los datos y esperar, hasta que pueda mandar datos y se invierten los papeles.

CAPÍTULO 3
PROCEDIMIENTO METODOLÓGICO

3.1 Metodología

Esta es la metodología seguida durante el desarrollo del proyecto de graduación.

a. Investigación acerca de los protocolos de comunicación MODBUS y MODBUS TCP: esta investigación es importante, ya que con base en la comprensión del funcionamiento de estos protocolos es que se realizará el resto del proyecto. El equipo que se usará es una computadora con acceso a internet y manuales varios. Tiempo estimado de duración: una semana.

b. Selección del microprocesador y otros componentes electrónicos importantes de manera idónea: esta etapa es importante para así determinar los componentes, o módulos existentes en el mercado, que de mejor manera se adaptan al proyecto. Se tomará en cuenta tanto la experiencia del ingeniero a cargo del proyecto como los parámetros principales de los protocolos, así como los criterios aprendidos durante la carrera y el consejo del profesor asesor. Tiempo estimado de duración: tres semanas.

c. Diseño del programa de traducción que se incorporará al microcontrolador: esta es la parte más importante, después de la de hardware. En esta se diseñará el código necesario para que el sistema sea capaz de la correcta traducción de un protocolo a otro. Se utilizará una computadora con programas especializados para la programación, tanto del microprocesador como del código en sí. Tiempo estimado de duración: cuatro semanas.

d. Diseño de los programas de aplicación que se utilizarán para las pruebas de funcionamiento: estos programas se encargarán de simular comandos de comunicación entre dispositivos que utilicen los protocolos MODBUS y MODBUS TCP, para así comprobar que el funcionamiento de todas las partes anteriores sea el correcto. Para crear los programas es necesario tener una computadora con software especializado (Delphi). Tiempo estimado de duración: tres semanas.

e. Depuración de software de aplicación y de traducción: para garantizar la veracidad de las pruebas del sistema y el funcionamiento óptimo de los programas de aplicación. Tiempo estimado de duración: una semana.

f. Comprobación del funcionamiento del sistema completo: parte en que se comprueba definitivamente el funcionamiento del sistema completo. Se utilizarán todas las partes del proyecto realizadas hasta el momento, además de equipo de medición (tester, ORC) por si se presentan contratiempos de tipo físico del circuito, una computadora con programas especializados. Tiempo estimado de duración: dos semanas.

g. Elaboración del informe final: en esta sección se elaborará el informe final en el que se especifican todos los detalles de tipo técnico y de desarrolllo del proyecto, esquemas de diseño del circuito, diagramas de flujo y código final de los programas usados, un manual de usuario y, en fin, cualquier dato de importancia para la empresa y que la haga capaz de reproducir en el momento en el que crea conveniente el prototipo final, para que entre en su período operacional. Para esto se usará una computadora con software de levantado de texto y graficación de diagramas de flujo. Tiempo estimado de duración: una semana.

CAPÍTULO 4
DESCRIPCIÓN DEL HARDWARE UTILIZADO

A continuación, se describe en detalle el hardware usado en este proyecto de graduación.

La pieza más importante del hardware usado es la tarjeta de marca *Rabbit 2000*, incluida en el kit de desarrollo *Rabbit 2000 TCP/IP*. La figura 4.1 muestra el kit de desarrollo completo, que incluye:

a. Una tarjeta de desarrollo *Rabbit 2000 TCP/IP*. Incluye el microprocesador Rabbit 2000, memoria Flash, memoria SRAM, hardware para Ethernet, puertos serie, entradas/salidas digitales.

b. Un disco compacto que contiene el programa usado para programar el microcontrolador, llamado *Dynamic C Special Edition ver. 7.20TSE*. También, se incluyen en el disco las librerías necesarias para el manejo del hardware, el *TCP/IP stack*[6], hojas de datos del kit y documentación varia relacionada con el producto.

c. Una tarjeta para demostraciones.

d. Una fuente de poder.

e. Un cable serial para programación.

[6] Así se le llama al conjunto de librerías y rutinas usadas para el manejo de todo lo que corresponde a los protocolos del TCP/IP.

Figura 4.1 Kit de desarrollo Rabbit 2000 TCP/IP

Las características mecánicas y eléctricas de la tarjeta de desarrollo se incluyen en las tablas 4.1 y 4.2.

Tabla 4.1 Especificaciones de la tarjeta de desarrollo Rabbit 2000 TCP/IP

Parámetro	Especificación
Tamaño de la tarjeta (con batería de apoyo opcional)	4.30" × 4.71" × 0.79" (109 mm × 120 mm × 20 mm)
Conectores	15 terminales atornillables, 1 RJ-12 y 1 RJ-45
Temperatura de operación	-20^0C a $+70^0$C
Humedad	5% a 95%, no condensada
Voltaje de entrada	+5 VCD a +40 VCD
Corriente	100mA @ 12 VCD
Interfase Ethernet	Conexión directa a redes Ethernet 10BaseT vía conector RJ-45
Entradas digitales	4 protegidas, 0 a 5 VCD (protección de –36 V a +36 VCD máximo)
Salidas digitales	4 de colector abierto (200mA, 40 VCD máximo)
Microprocesador	Rabbit 2000™
Reloj	18.432 MHz
SRAM	128kb, montaje de superficie (soporta 32k – 128k)
Flash EPROM	256kb para programa, mas 256kb para guardar archivos (soporta 128k – 512k)
Temporizadores	7 temporizadores de 8 bits disponibles

Tabla 4.2 Continuación de tabla 4.1

Parámetro	Especificación
Puertos seriales	♦ 1 RS-232 (tres líneas), 1 RS-485 y 1 RS-232 para programación ♦ RS-232 y RS-485 pueden ser configurados como 1 RS-232 (5 líneas) o 2 RS-232 (3 líneas)
Tasa de transmisión serial	Máximo de transmisión asíncrona de 115 200 bps para ambos puertos serie.
Watchdog/Supervisor	Sí
Reloj de tiempo real	Sí
Batería de respaldo	En tarjeta de batería de respaldo (no incluida)

La figura 4.2 muestra en detalle las dimensiones mecánicas y la posición de los componentes en la tarjeta de desarrollo.

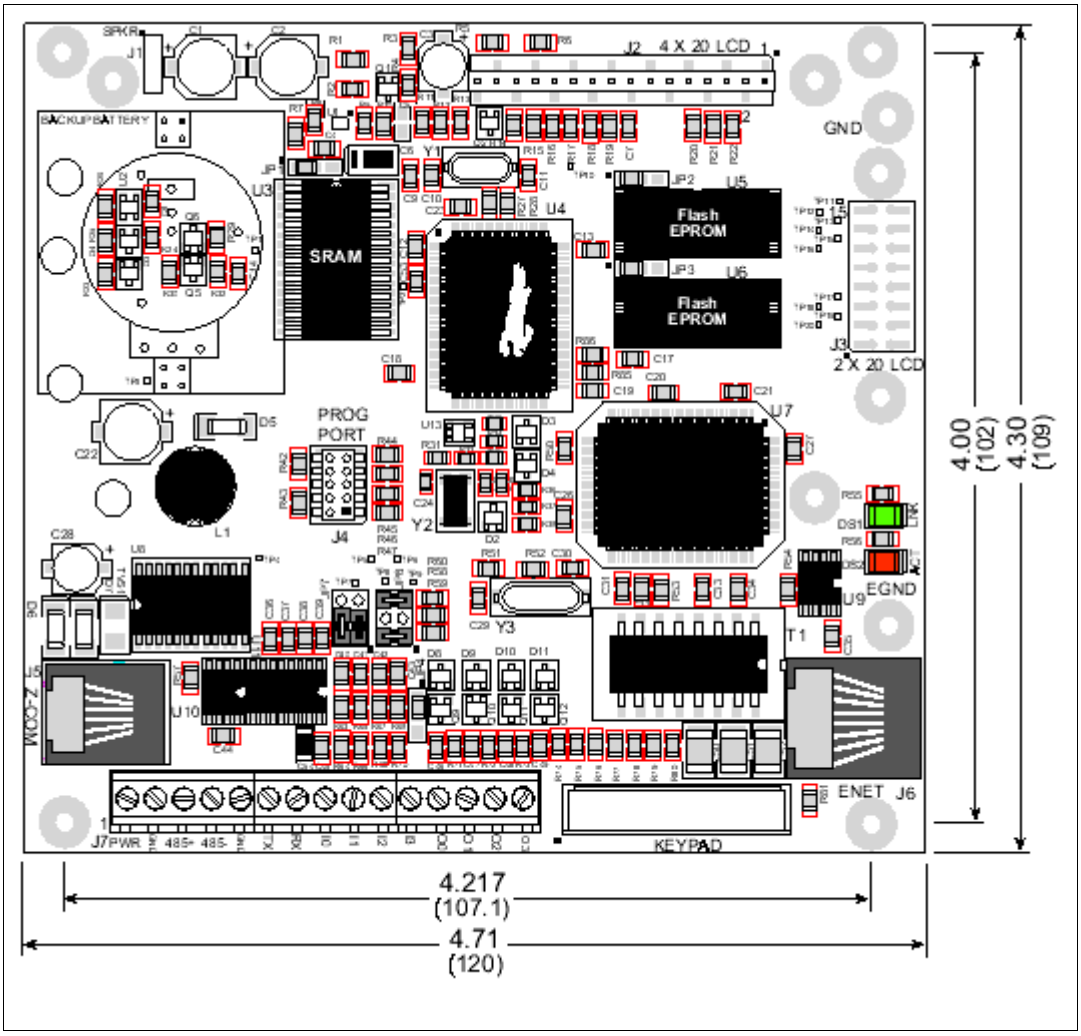


Figura 4.2 Dimensiones mecánicas de la tarjeta de desarrollo Rabbit 2000 TCP/IP

La tarjeta de desarrollo cuenta con 15 pines de tornillo J7, un conector RJ-12 para comunicación serial por RS-232 o RS-485 y un conector RJ-45 para Ethernet. La figura 4.3 muestra un esquema de los conectores de la tarjeta.

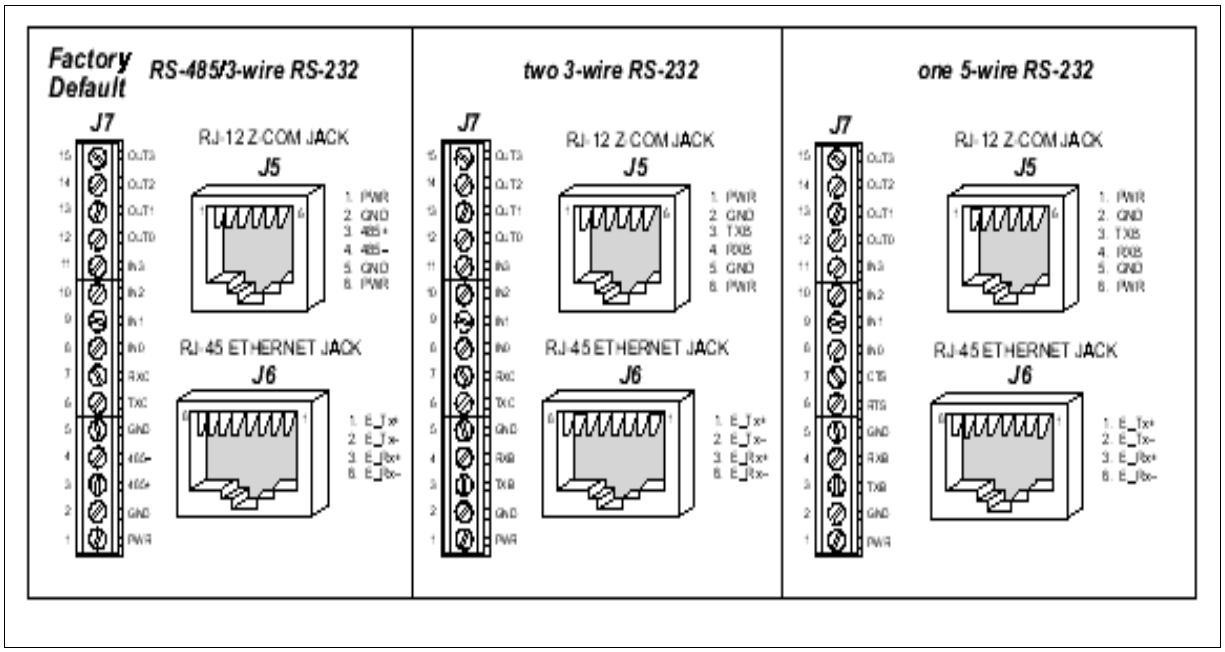


Figura 4.3 Esquema de los conectores de la tarjeta de desarrollo Rabbit 2000 TCP/IP.

Para la configuración de los puertos seriales, en la tarjeta existen una serie de puentes (o jumpers) que sirven al usuario para elegir una de las opciones que esta tarjeta ofrece. La figura 4.4 muestra la ubicación de dichos puentes en la tarjeta.

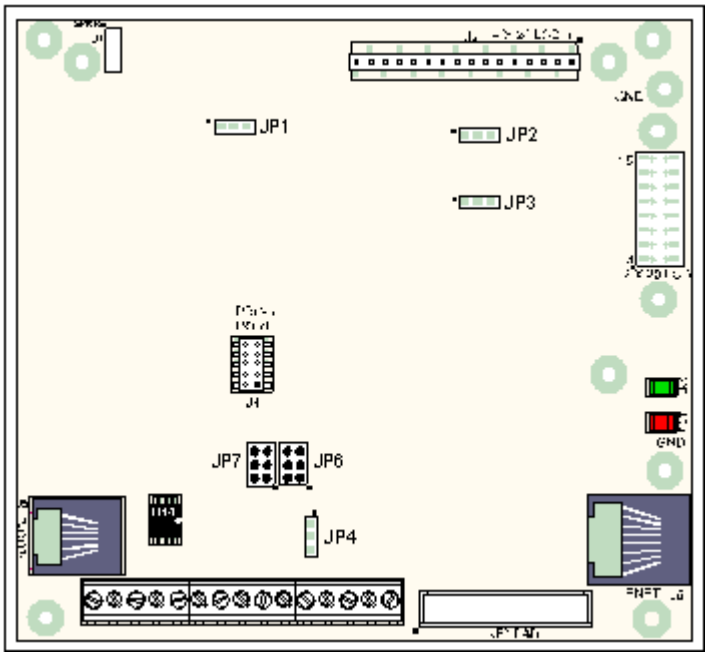


Figura 4.4 Ubicación de los puentes (jumpers) de configuración en la tarjeta de desarrollo Rabbit 2000 TCP/IP

Las tablas 4.3 y 4.4 muestran la forma en que los puentes se relacionan con los componentes de la tarjeta, y da una idea de las posibles configuraciones de hardware.

Tabla 4.3 Configuración de puentes (jumpers) para la tarjeta de desarrollo Rabbit 2000 TCP/IP

Nombre	Descripción	Pines conectados		Conectados en fábrica
JP1	Tamaño de SRAM	1-2	128k	X
		2-3	512k	
		Ninguno	32k	
JP2	Tamaño de mem. Flash 1	1-2	128k/256k	X
		2-3	512k	
JP3	Tamaño de mem. Flash 2	1-2	128k/256k	X
		2-3	512k	
JP4	Resistencias de pull up/pulldown digitales	1-2	Pulled up	X
		2-3	Pulled down	
JP5	Selección de banco, memoria Flash	1-2	Modo normal	X
		2-3	Modo de banco	

Tabla 4.4 Continuación de Tabla 4.3

Nombre	Descripción	Pines conectados		Conectados en fábrica
JP6	RS-485 Bias and termination resistors	1-2 5-6	Bias and termination resistors connected	X
		-	Bias and termination resistors not connected	
JP7	Selección RS-232/RS-485	3-5 4-6	RS-232 TxC/RxC y señales RS-485 en J7	X
		1-5 2-6	RS-232 TxB/RxB en J5 (chip U10 debe ser removido)	

Nota: sólo los jumpers JP6 y JP7 son removibles. Las otras conexiones son hechas usando resistencias de soldadura de superficie de 0Ω.

Debido a la forma en que están conectados los puentes JP6 y JP7 (los únicos removibles), si el usuario quiere cambiar las opciones de uso para los puertos serie, debe referirse a la información que se muestra en la tabla 4.5. Nótese que de elegir la opción de dos puertos RS-232 de tres líneas, el puerto RS-485 se pierde. Además, en la tabla se muestran los valores que adquieren los pines del conector JP7. Estas

configuraciones deben ir acompañadas de ciertas configuraciones de software, pues sin dichas configuraciones, no es posible que los puertos funcionen adecuadamente.

Tabla 4.5 Configuraciones para puertos serie (tarjeta de desarrollo versión 175-206)

Item	Por defecto	Un RS –232 (3 líneas) y RS-485	Dos RS-232 (3 líneas)	Un RS-232 (5 líneas)	RS-232 en J5
Terminal JP7		3-5 4-6	1-3 2-4	1-3 2-4	1-5 2-6
Terminal JP6		1-2 5-6	--	--	Sin puentes instalados
U10		Salida	Salida	Salida	Entrada
J7-3		RS-485+	TxB	TxB	--
J7-4		RS-485-	RxB	RxB	--
J7-6		TxC	TxC	RTS	TxC o RTS
J7-7		RxC	RxC	CTS	RxC o CTS
J5-3		RS-485+	--	--	TxB
J5-4		RS-485-	--	--	RxB

Las áreas cercanas al cristal de oscilación y al circuito de la batería de apoyo, están cubiertas por una capa protectora con base de silicón (protector Down Corning 1-2620). Esta capa de protección protege a estos circuitos de alta impedancia de la humedad y los contaminantes a lo largo del tiempo. Cualquier componente de esta

área protegida puede ser reemplazado usando las técnicas comunes de soldadura superficial. Después del cambio se debe aplicar otra capa de protector, con el fin de que los elementos queden de nuevo aislados de la humedad y los contaminantes.

El puerto de programación de la tarjeta de desarrollo (puerto serie A), puede ser usado tanto para cargar los programas en la tarjeta, como para un puerto serie normal, pudiéndosele dar un gran uso como puerto de diagnóstico o de configuración en el campo. La figura 4.5 muestra la asignación de pines de este puerto especial.

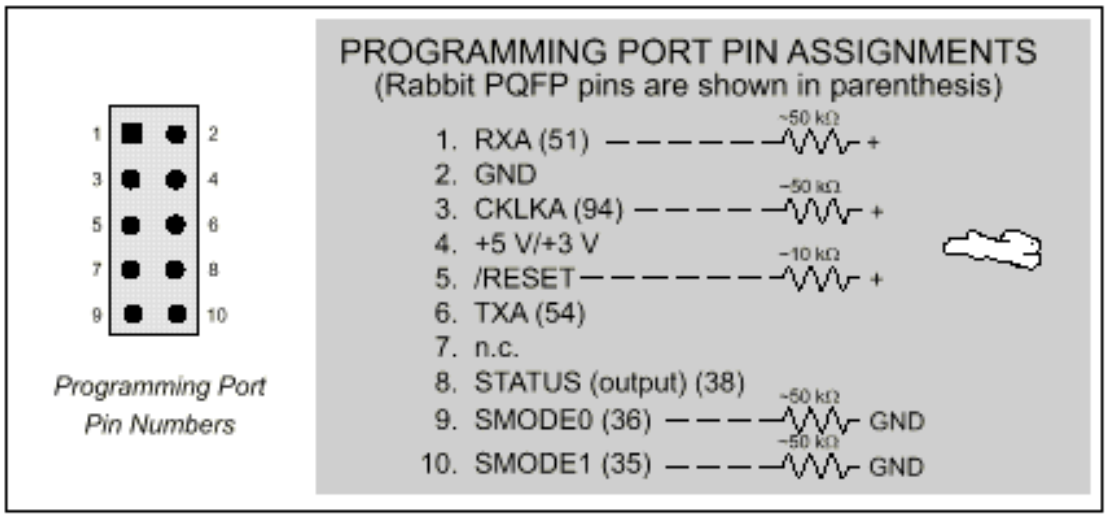


Figura 4.5 Asignación de pines del puerto de programación, tarjeta de desarrollo Rabbit 2000 TCP/IP

El cable de programación de la tarjeta es especial, no sólo por el tipo de conector, sino porque se puede usar en dos modos de funcionamiento: modo *DIAG* y modo *PROG*. Cada uno de los modos tiene un conector aparte.

El conector PROG es usado sólo cuando el cable está conectado a la tarjeta durante el desarrollo de nuevas aplicaciones. Por otra parte, el conector DIAG del cable de programación permite que dicho cable sea usado como un convertidor de niveles RS-232 a CMOS para comunicación serie, que es apropiado para el monitoreo y la depuración de aplicaciones en sistemas que todavía estén corriendo.

Una vez que se establezca que el puerto de programación no va a ser utilizado más para este fin, los pines del puerto pueden usarse como líneas de estrada y salida adicionales, según indica la tabla 4.6.

Tabla 4.6 Configuraciones de pines para el puerto de programación de la tarjeta de desarrollo Rabbit 2000 TCP/IP

Pin	Nombre de pin	Uso normal	Uso alternativo	Notas
1	RxA	Puerto serial A	PC6—Entrada	
2	GND			
3	CLKA		PB1—Entrada programable por bit o paralela.	
4	VCC			
5	RESET			Conectado al generador de reset U4
6	TxA	Puerto serial A	PC7—Salida	
8	STATUS		Salida	
9	SMODE0		Entrada	Debe estar en bajo cuando se reinicia un BL200
10	SMODE1		Entrada	Debe estar en bajo cuando se reinicia un BL200

CAPÍTULO 5
DESCRIPCIÓN DEL SOFTWARE DEL SISTEMA

En este apartado se explica todo lo referente al programa desarrollado para la traducción de los protocolos y el manejo de las conexiones TCP/IP.

Según estaba planteado el problema, se requería de un programa que, básicamente, hiciera de puente entre dos interfases físicas, RS-232 y Ethernet 10BaseT. Entre los programas de ejemplo que se incluyen en el disco compacto de *Dynamic C*, existe un programa llamado *serialexa.c*, que direcciona todos los datos que llegan a un puerto serie hacia un puerto TCP y viceversa. Basándose en este programa, se analizó la estructura del software de traducción, así como el manejo de las rutinas de TCP/IP y el uso del puerto serie.

El programa desarrollado, denominado *mod_bridge.c*, logra con éxito la implementación, tanto del manejo de la interfase de conexión como de la traducción de los protocolos. Las estructuras de datos utilizadas en el programa son las siguientes:

a. VsState: es una estructura común en C que almacena la información de los parámetros de configuración de cada *socket*[7]. Esta es la estructura *TCB*[8] del programa. En esta se encuentra el socket activo, el estado de la conexión y algunas variables que contienen datos, así como una constante que fija el tiempo de espera de la conexión.

[7] Socket (conector): Identificador de una conexión. Está formado por la dirección IP más el número de puerto que corresponden a cierta conexión TCP/IP.
[8] TCB (Transmition Control Block o Bloque de Control de Transmisión): sirve para la introducción en un programa de datos que correspondan a una conexión TCP/IP en particular.

b. VsInfo: es otra estructura que inicializa la anterior, poniéndola en el estado inicial una vez que arranca el programa. En ella se especifica el número de puerto que se usó (en el caso de *MODBUS TCP* es el 502), el valor de *timeout*[9] entre caracteres recibidos, la velocidad de transmisión serial, el modo en que inicia una conexión (activa o pasiva; en el caso del proyecto, las conexiones siempre son pasivas), la dirección IP de destino (para conexiones activas), el puerto de destino (también para conexiones activas) y una variable que inicializa el *stack TCP/IP*[10] de la tarjeta de desarrollo en uno de los modos de manejo de datos posible, que son binario y ASCII. También se encuentra un valor de open_timeout, para avisar cuando una conexión no se abre satisfactoriamente.

c. Const VsInfo factory_defaults: tiene los datos iniciales para cargar al momento del inicio del programa dentro de la estructura VsInfo usada. En el caso del proyecto los valores fueron:

    1. Puerto: 502
    2. Timeout: 1200ms
    3. Velocidad serial: 9600 baudios
    4. Modo: pasivo
    5. Dirección de destino: 0
    6. Puerto destino: 0
    7. Modo TCP: binario (porque se usan estructuras *MODBUS TCP tipo RTU*[11])
    8. open_timeout: 1000ms

[9] Tiempo de espera.
[10] Se le llama stack (pila) TCP/IP al conjunto de rutinas usadas para el manejo de los protocolos TCP/IP en un sistema dado. Ver Anexo B.2.
[11] MODBUS RTU: uno de los dos tipos posibles de codificación que existen para el protocolo MODBUS. En esta, los datos en los mensajes se interpretan como grupos de 8 bits, representando números hexagesimales.

d. vs_init(): rutina para inicializar las estructuras VsState, para cuando pasen por primera vez por la rutina principal.

e. CRC: rutina para calcular el CRC de los mensajes *MODBUS*. Se hizo con base en la información obtenida de las especificaciones del protocolo.

f. truncate_msg: es una rutina para darle formato a los mensajes *MODBUS TCP*. Elimina los 6 primeros bytes de los mensajes recibidos a través de la conexión TCP/IP, los que conforman el encabezado del mensaje, el que no es necesario en los mensajes *MODBUS*.

g. format_slave_msg: es la rutina que le da formato a los mensajes enviados por el esclavo, a través del puerto serie. Lo que hace esta rutina es retomar el mensaje recibido, incluir el encabezado especial para *MODBUS TCP* (que fue cortado por la rutina truncate_msg) y quitar los bytes de CRC, que no necesita el mensaje *MODBUS TCP.*

h. vs_handler: es la rutina que hace todo el trabajo. Se encarga de manejar la máquina de estados que rige una conexión TCP/IP, además, es en esta donde se recogen los mensajes, se traducen y se reenvían a sus respectivos destinos.

i. main(): rutina principal, necesaria en cada programa C. Es ella la que inicializa las variables a usar, así como al puerto serie (en este caso, el puerto C) y a la conexión TCP/IP. En un ciclo infinito llama a vs_handler, para realizar la traducción de los mensajes, si se da una conexión TCP/IP. Mientras pasa eso, se queda en estado de espera.

Los mensajes *MODBUS* tienen un encabezado especial, según las especificaciones del protocolo. Este encabezado debió almacenarse en memoria durante la recepción de los mensajes para realizar con éxito la posterior conversión de protocolos. La siguiente es una lista de las principales variables del programa. Las

variables de la e a la j corresponden a las que se usaron para almacenar el encabezado especial.

a. net_read_buf[260]: arreglo de variables tipo char usado como buffer de entrada para los datos provenientes de la conexión TCP/IP.

b. net_write_buf[260]: arreglo de variables tipo char usado como buffer de salida para los datos que van hacia la conexión TCP/IP. Almacena el mensaje ya traducido que va hacia la conexión.

c. ser_read_buf[260]: arreglo de variables tipo char usado como buffer de entrada de los mensajes que vienen del puerto serial.

d. ser_write_buf[260]: arreglo de variables tipo char que es usado como buffer de salida de los mensajes que se envían por el puerto serial. Almacena los mensajes convertidos que van hacia el puerto serial.

e. tr_idb0: variable tipo byte que guarda el primer byte de la identificación de transacción de los mensajes *MODBUS TCP.*

f. tr_idb1: variable tipo byte que guarda el segundo byte de la identificación de transacción de los mensajes *MODBUS TCP.*

g. proto_trb2: variable tipo byte que guarda el primer byte de la identificación de protocolo de los mensajes *MODBUS TCP.*

h. proto_trb3: variable tipo byte que guarda el segundo byte de la identificación de protocolo de los mensajes *MODBUS TCP.*

i. len_upb4: variable tipo byte que almacena el primer byte del campo de longitud de los mensajes *MODBUS TCP*.

j. len_upb4: variable tipo byte que almacena el segundo byte del campo de longitud de los mensajes *MODBUS TCP*.

k. net_bytes_read: variable que guarda el número de bytes leídos del buffer de entrada de la conexión TCP/IP. Es de tipo integer (int).

l. ser_bytes_read: variable que guarda el número de bytes leídos del buffer de entrada del puerto serie C. Es de tipo integer (int).

m. vs_info: del tipo VsInfo. Guarda la inforrmación que inicializa la conexión TCP/IP.

n. vs_state: tipo VsState. Guarda los estados de la conexión TCP/IP.

Dentro del programa usado, existen una serie de estados de conexión, definidos con la directiva *#define* de C, que realiza una sustitución de las apariciones de alguna palabra en el programa por otra, que es determinada por el programador. Los estados de conexión son los siguientes:

a. VS_INIT: estado inicial de la rutina vs_handler(VsState* state). Su valor es 0 y en este estado, una conexión se considera cerrada o nunca abierta.

b. VS_LISTEN: en este estado, el programa espera que una conexión se realice. Su valor es 1.

c. VS_OPENING: de valor 2, espera mientras el programa abre una conexión.

d. VS_OPEN: se da cuando hay una conexión abierta. Su valor es 3.

e. VS_WAITCLOSE: en este estado, se espera a que una conexión termine. Su valor es 4.

El siguiente diagrama de flujo (figura 5.1) corresponde al programa mod_bridge.c. En este diagrama se muestran tres instrucciones que se incluyen dentro de la lista de funciones del software *Dynamic C*. Estas funciones son sock_init(), usada para dar inicio a una conexión TCP/IP; serCopen(vs_info.baud), que abre el puerto serial C, con una velocidad de 9600 baudios (este dato se

almacena en la variable vs_info.baud); tcp_tick(NULL), función usada en este caso para atender solicitudes *Ping*[12] de otros aparatos conectados a la misma red que el traductor.

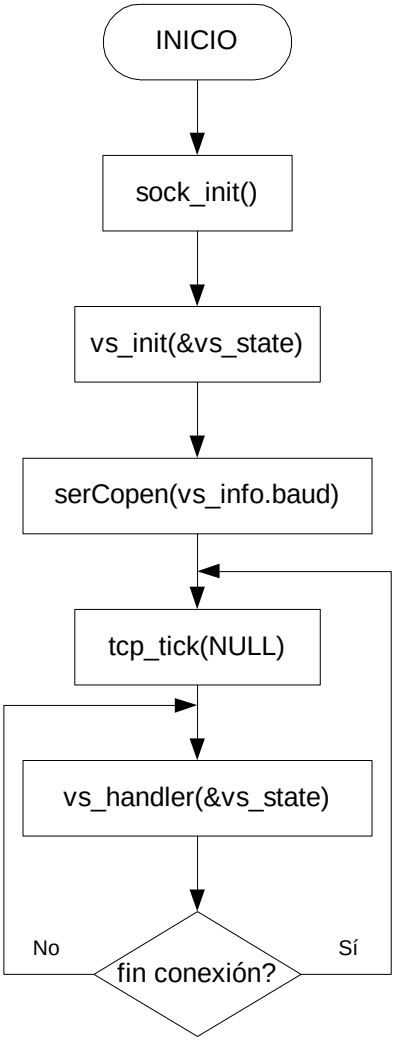


Figura 5.1 Diagrama de flujo del programa principal

[12] Ping: es un mensaje en el que un aparato en red verifica que otro esté activo, enviándole un mensaje. La respuesta consiste en devolverle el mismo mensaje a la fuente.

La figura 5.2 muestra el diagrama de flujo de la rutina vs_handler(VsState* state). Esta rutina, como ya se mencionó, es la que se encarga del trabajo de recepción, traducción y envío de los mensajes.

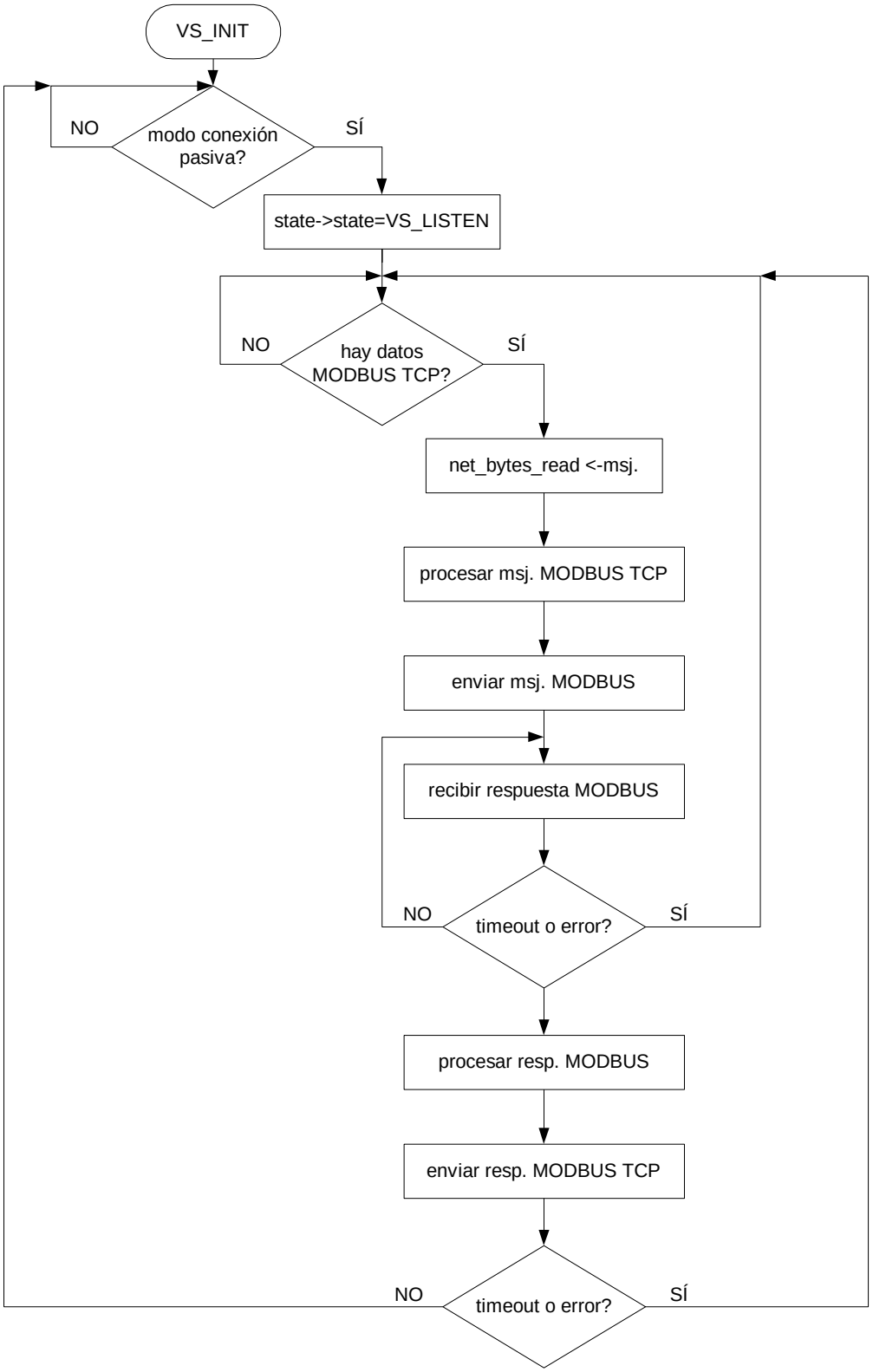


Figura 5.2 Diagrama de flujo para la rutina principal del programa traductor

CAPÍTULO 6
ANÁLISIS Y RESULTADOS

6.1 Explicación del diseño

6.1.2 Hardware utilizado en el proyecto

En este proyecto, se pensó inicialmente en diseñar y construir un prototipo que cumpliera con los requerimientos del proyecto. Como, después de una investigación de opciones, se encontraron piezas de hardware listas que podían adquirirse y empezar a usarse inmediatamente, se decidió comprar una de estas. La tarjeta elegida forma parte del kit de desarrollo TCP/IP *Rabbit 2000*, y sus características principales se detallan en el capítulo 5 de este informe.

La tarjeta de desarrollo *Rabbit 2000 TCP/IP* cumple con todos los requerimientos del proyecto, pues provee puertos seriales y una conexión Ethernet 10BaseT, cuenta con suficiente memoria para almacenar tanto las rutinas de TCP/IP como las de traducción de los protocolos y su precio es menor a los US$700. Los esquemáticos de la tarjeta de desarrollo y el cable de programación usado se presentan en el Anexo B1.

6.1.2 Software utilizado en el proyecto

Para la programación de las rutinas de traducción y el manejo de los protocolos TCP/IP, se usó el software *Dynamic C Special Edition ver. 7.20TSE*, que provee el fabricante del kit de desarrollo. Este software es un sistema integrado de desarrollo de aplicaciones para sistemas propios. Está

diseñado para usarse con microcontroladores de marca *Z-World* y otros controladores basados en el microprocesador *Rabbit*. Un microprocesador *Rabbit* puede manejar aproximadamente 50 000 sentencias C o un megabyte.

El software *Dynamic C* integra las funciones de edición, compilación, enlace, carga de aplicaciones y depuración en un programa. De hecho, el compilado, el enlace y la carga son una función. Los programas pueden ser ejecutados y depurados interactivamente, a nivel de código fuente o a nivel de código máquina. Además, las instrucciones en C pueden combinarse con código ensamblador.

Ahora se mencionarán el procedimiento usado para programar el traductor, las características del programa desarrollado y los conceptos que se tomaron en cuenta para desarrollarlo.

Primero que nada, se realizó una investigación acerca de los protocolos a usar, así como de los usos de las rutinas TCP/IP, *MODBUS* y *MODBUS TCP*.

Según el modelo de referencia OSI[13], se pudo enmarcar la aplicación hecha para este proyecto en dicho modelo. De acuerdo con el protocolo *MODBUS TCP*, el no es necesario usar todas las capas del modelo OSI. La figura 6.1 muestra la relación entre el modelo OSI y el programa de aplicación.

[13] Ver Anexo B2.

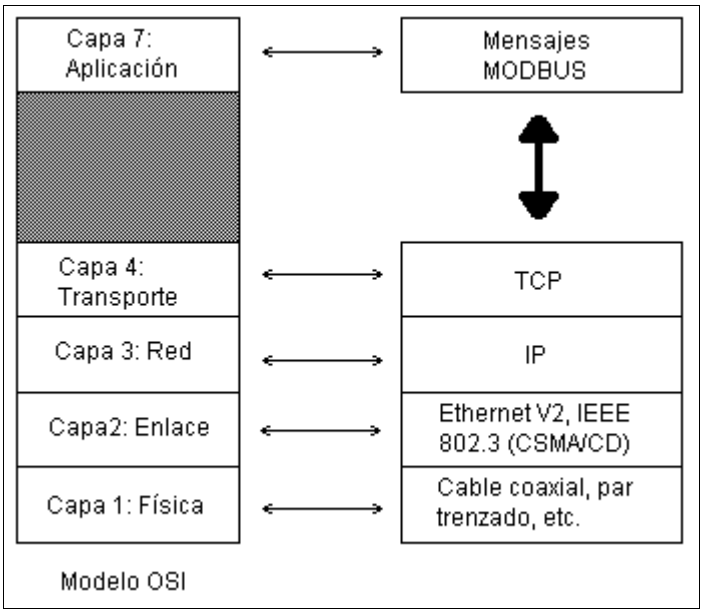


Figura 6.1 Capas de protocolos de acuerdo al modelo OSI y su relación con el programa traductor del proyecto.

En primer lugar, fue necesario tomar en consideración los protocolos de TCP/IP que debían utilizarse para crear un sistema mínimamente funcional. Esto porque para el caso de este proyecto, no es necesario usar todos los protocolos que se incluyen en el TCP/IP.

La estructura de funcionamiento de TCP/IP hace obligatorio el uso de algunos protocolos, tal es el caso del *ICMP*[14], que en este caso se usa solamente para contestar a solicitudes *Echo ICMP*[15]. También, se debió hacer uso del protocolo *ARP*[16], con el fin de que los otros equipos conectados a una red sepan la dirección física de la tarjeta traductora.

Los programas que usan TCP tienen un servicio de conexión entre los programas que llaman y los llamados, chequeo de errores, control de flujo y capacidad de interrupción. Estas ventajas garantizan la fiabilidad en la entrega

[14] ICMP: Internet Control Message Protocol. Ver Anexo B2.
[15] Una solicitud de este tipo será usada por un host (cliente) para saber si otro host es operativo. El receptor de la solicitud la devuelve a su origen. Esta aplicación recibe el nombre de *Ping.* Ver Anexo B2.
[16] ARP: Address Resolution Protocol. Ver Anexo B2.

sin errores de los mensajes a traducir, y es por esto que *MODBUS TCP* deja que dicha etapa de verificación de errores la realice la capa TCP/IP.

Para el caso de este proyecto de graduación, se decidió usar como interfase entre la conexión TCP/IP y otros programas la pila (stack) de programas de la capa de red. Como en esta capa sólo está el IP, entonces la interfase la determina este protocolo. Esta interfase tiene las siguientes características.

La interfase envuelve el programa de usuario llamando a una rutina que introduce entradas en una estructura de datos llamada el bloque de control de transmisión (TCB). Las entradas se realizan inicialmente en la pila de *hardware* y son transferidas al TCB por medio de una rutina de sistema. Estas entradas permiten al TCP asociar un usuario con una conexión particular, de modo que pueda aceptar comandos de un usuario y mandarlos a otro usuario en la otra parte de la conexión. TCP utiliza unos identificadores únicos para cada parte de la conexión. Esto se utiliza para recordar la asociación entre dos usuarios. Al usuario se le asigna un nombre de conexión para utilizarlo en futuras entradas del TCB. Los identificadores para cada extremo de la conexión se llaman *sockets*. El *socket* local se construye concatenando la dirección IP de origen y el número de puerto de origen. El socket remoto se obtiene concatenando la dirección IP de destino y el número de puerto de destino. Todo esto se hace porque TCP es un protocolo orientado a la conexión.

El TCP recuerda el estado de cada conexión por medio del TCB. Cuando se abre una conexión, se efectúa una entrada única en el TCB. Un nombre de conexión se le asigna al usuario para activar los comandos de la conexión. Cuando se cierra una conexión se elimina su entrada del TCB.

Para el caso del programa desarrollado, las estructuras VsState son usadas como TCB. Se debe recordar que la estructura VsState guarda la información del socket en uso por la rutina de manejo (vs_handler).

En cuanto a la parte de *MODBUS*, se debe mencionar que, para realizar la traducción de los protocolos se debió tomar en cuenta los siguientes puntos.

Los mensajes *MODBUS* tienen la siguiente estructura, mostrada en la figura 6.2. El significado de cada elemento es el siguiente:

a. Additional address (dirección adicional): es el número, o dirección, del dispositivo al que se le envía el mensaje.

b. Function code (código de función): indica la función que el destinatario del mensaje debe cumplir, por ejemplo, enviar datos de señales digitales, forzar un valor en un registro, etc.

c. Data (datos): campo variable, en el que, de ser necesario, se incluyen datos.

d. Error check (chequeo de error): es donde se inserta el CRC.

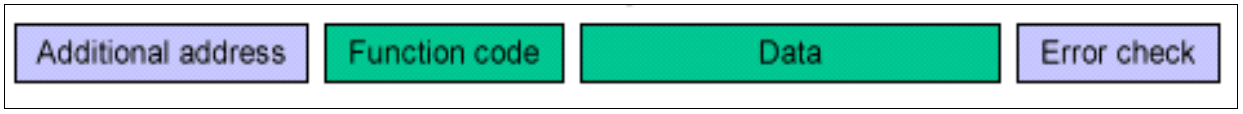


Figura 6.2 Estructura de mensaje MODBUS

Los mensajes *MODBUS TCP* tienen la siguiente forma, según se ve en la figura 6.3. Los campos de Function code y Data son iguales que en el caso

anterior, pero en esta estructura se incluye un encabezado (*MBAP*[17] header) adicional, el cual está formado por los siguientes elementos:

a. Transaction identifier (identificador de transacción): su longitud es de 2 bytes. Identificación de una transacción de petición o respuesta *MODBUS.*

b. Protocol identifier (identificador de protocolo): de 2 bytes. 0 = protocolo *MODBUS.*

c. Length field (campo de longitud): de 2 bytes de longitud. Número de bytes siguientes, a partir de aquí, dentro del mensaje.

d. Unit identifier (identificador de unidad): de 1 byte. Es el equivalente del Additional Address.

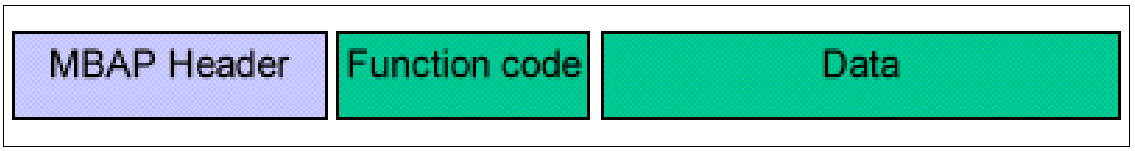


Figura 6.3 Estructura de mensaje MODBUS TCP

Como es claro en las dos figuras anteriores, ambas estructuras de mensaje tienen elementos en común y diferencias entre sí. La estructura *MODBUS* requiere de campos para el CRC, mientras que los mensajes*; MODBUS TCP* dejan el chequeo de errores al protocolo TCP/IP y no los requiere; pero un mensaje *MODBUS TCP* necesita el encabezado MBAP, el que no está presente en los mensajes *MODBUS.* Estas diferencias y similitudes se aprecian en la figura 6.4.

[17] MBAP: MODBUS Application Protocol Header (encapezado de protocolo de aplicación MODBUS).

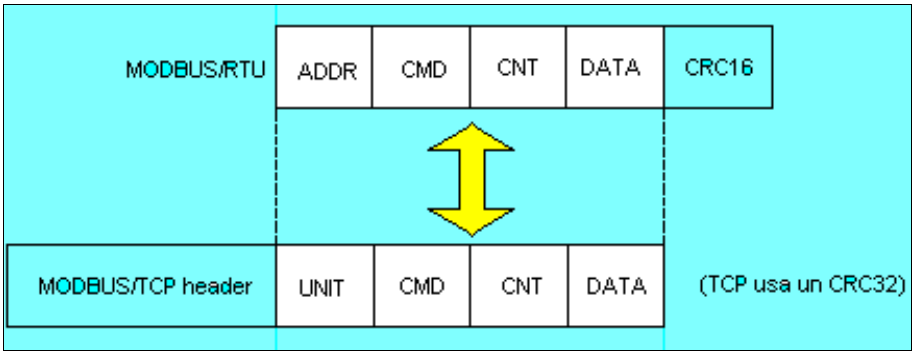


Figura 6.4 Esquema comparativo de las diferencias entre mensajes MODBUS RTU y MODBUS TCP

Entonces, para realizar la traducción de los mensajes de manera acertada, se siguieron los siguientes pasos:

a. Obtener un mensaje de *MODBUS TCP.*

b. Truncar y guardar (se necesitan para después) los primeros 6 bytes de dicho mensaje.

c. Añadir el CRC al mensaje.

d. Enviarlo por el puerto serie.

e. Esperar por el mensaje de respuesta.

f. Una vez recibida la respuesta, formar el mensaje de respuesta en formato *MODBUS TCP*, insertando los bytes del encabezado MBAP en el punto 2 (pero cuidando que el campo de longitud sea el adecuado para el nuevo mensaje) y truncando los dos últimos bytes de la respuesta (el CRC en la respuesta *MODBUS*).

g. Enviar la respuesta al dispositivo *MODBUS TCP.*

Calcular adecuadamente el CRC de los mensajes *MODBUS* es importante para garantizar la buena transmisión de estos. La rutina que se

encarga de esto se programó según los parámetros del protocolo *MODBUS*. El valor CRC es de 16 bits, es binario y debe ser generado por el dispositivo transmisor. Se calcula de la siguiente manera:

a. Primero se carga un registro de 16 bits con FFFFh, este será el registro CRC.

b. Se pasan el primer byte del mensaje y la parte baja del registro CRC por una operación XOR, el resultado se guarda en el registro CRC.

c. Se desplaza el registro CRC un bit a la derecha (hacia el bit menos significativo), se reemplaza el bit más significativo con un cero, se extrae y se examina el bit menos significativo.

d. Si el bit menos significativo es cero, se repite el paso 3. Si es igual a uno, el registro CRC se pasa por una operación XOR junto con un valor de A001h (1010 0000 0000 0001).

e. Se repiten los pasos 3 y 4 hasta completar ocho desplazamientos. Cuando esto ocurre, se ha procesado un byte completo.

f. Se repiten los pasos 2-5 hasta completar todo el mensaje, exceptuando los bits de parada, de inicio y de paridad. El contenido final del registro CRC es el valor CRC.

g. Cuando se coloca el valor CRC en el mensaje que se está construyendo, se intercambian las partes baja y alta del registro.

Todas las acciones anteriormente descritas son realizadas por el programa que se implementó para el proyecto.

Para las pruebas del sistema, se usaron dos programas de simulación de estaciones *MODBUS*. En el Centro de Servicio, Investigación y Desarrollo se había conseguido un simulador para Windows de estación esclava

MODBUS, el que fue utilizado. Para simular la estación maestra *MODBUS TCP*, se descargó de internet un programa llamado *Mdbus,* que simula una estación maestra o una esclava en modo RTU y TCP. En un principio se pensó en programar las aplicaciones de prueba para comprobar el funcionamiento del traductor, pero al conseguir software de simulación de las estaciones maestra y esclava, esto no se realizó.

Las pruebas consistieron en enviar comandos *MODBUS* comunes, como forzar el valor de una entrada digital, leer una entrada digital, escribir un valor en un registro, leer un registro y leer una entrada analógica.

Durante las primeras pruebas, en la pantalla del simulador de estación maestra se desplegó siempre un mensaje de error en la comunicación. Al revisar el programa, se encontró que el valor asignado al tiempo de espera de respuesta para los mensajes *MODBUS* era muy pequeño (de 5 ms). Dado que los dispositivos *MODBUS* transmiten sus datos a bajas velocidades y que el indicador de fin de mensaje es una pausa de aproximadamente 3.5 caracteres, este valor debió ser cambiado por uno mucho más grande (de 1200 ms), con lo que se solucionó el inconveniente.

A la hora de tratar de escribir un valor en un registro, se presentó en la pantalla de la estación maestra un mensaje de error de comunicación. Según la ayuda del programa, este mensaje se da cuando el simulador envía tres mensajes seguidos y no recibe una respuesta satisfactoria, o la respuesta está mala. Sin embargo, en la pantalla del esclavo, se desplegaba el nuevo valor de manera correcta.

6.2 Alcances y limitaciones del proyecto

6.2.1 Alcances

El proyecto cumplió con todos los alcances propuestos al principio. El prototipo final realiza la traducción de un protocolo a otro, además, maneja de manera adecuada la pila de programas TCP/IP.

La conexión del prototipo en una red local pudo verificarse satisfactoriamente, pues respondió bien a los llamados *ping* hechos desde cualquier computadora que estaba en su misma red.

Las pruebas del sistema fueron positivas en todos los casos. En las ventanas de los simuladores se mostraron los valores que se esperaban en cada una de las pruebas hechas.

El valor del kit de desarrollo es de US$199 más el envío, precio que es menor que los US$700 que en promedio se pagaría por un dispositivo comercial de similares funciones.

6.2.2 Limitaciones

El prototipo parece no responder bien al comando *MODBUS* de escritura de un valor en registro, porque en la pantalla del simulador de unidad maestra *MODBUS TCP* se despliega un mensaje de error de comunicación cuando se intenta realizar el comando. Pese a este resultado, el valor se escribió correctamente en el registro en todas las pruebas y el enlace TCP/IP no se pierde. Este mensaje de error se desplegó por un par de segundos en todas las ocasiones que se usó el comando mencionado.

Este prototipo puede manejar solamente una conexión TCP/IP a la vez. Por tanto, si se quiere que la versión final pueda manejar más de una conexión, hay que modificar el programa principal implementado.

La dirección IP que necesita el prototipo para su conexión a red, no se puede modificar antes ni durante el funcionamiento de este. La única forma de hacer el cambio, es re escribiendo el valor de la dirección directamente en el programa fuente en C de traducción/comunicación (llamado mod_bridge.c) y volviendo a compilar el programa y a grabar la memoria Flash EPROM de la tarjeta. Esto no permitiría hacer este tipo de modificaciones en el sitio en donde esté un prototipo, si es que no se cuenta con una computadora con el software *Dynamic C* instalado y un cable de programación.

CAPÍTULO 7
CONCLUSIONES Y RECOMENDACIONES

7.1 Conclusiones

La sencillez y las opciones que presentan los protocolos de comunicación *MODBUS* y su variante *MODBUS TCP*, los hacen convertirse en una herramienta muy útil para transmitir datos en redes industriales. Como el *MODBUS TCP* permite al usuario implementar sistemas que se pueden conectar a redes locales o a la internet, el monitoreo, tanto como el envío y la recepción de datos se puede hacer remotamente, lo que representa una ventaja adicional de esos sistemas.

Se decidió que el objetivo de desarrollar los programas de aplicación que se requerían para las pruebas de sistema se cumplía con la utilización de dos programas ya listos, uno proporcionado por la empresa (simulador de esclavo *MODBUS*) y el otro conseguido durante el proyecto (simulador de maestro *MODBUS TCP*). El propósito de desarrollar las aplicaciones fue el de hacer las pruebas del sistema y con los programas mencionados se hizo satisfactoriamente.

El programa de traducción desarrollado se programó en el lenguaje C, que es una ventaja para la interpretación de la lógica La elección de este lenguaje fue correcta, pues el número de líneas programadas se incrementaría demasiado si se hubiera usado el lenguaje ensamblador, lo que dificultaría en el futuro la adaptación rápida del programa.

El programa de traducción desarrollado realiza satisfactoriamente el trabajo para el que fue creado. Los mensajes, tanto *MODBUS* como *MODBUS TCP*, son leídos, traducidos y reenviados de manera confiable y rápida. La tasa de errores en los mensajes no pudo ser determinada, puesto que en el caso de las transmisiones

del comando que presentó problemas, en la pantalla del esclavo *MODBUS* se desplegó el resultado esperado y la conexión no se perdió.

La elección del kit de desarrollo *Rabbit 2000 TCP/IP* resultó ser la mejor, tanto por su precio (menor a los US$700 de un traductor comercial) como por sus prestaciones. Las mayores ventajas de este kit, con respecto a otros productos similares, son: que posee las interfases físicas que se necesitaron para el proyecto; se programa en C; las rutinas para manejar TCP/IP son provistas por el fabricante, lo que ahorra gran cantidad de tiempo al usuario, que puede empezar a hacer aplicaciones casi inmediatamente después de recibir el kit; tiene una cantidad de memoria de programa suficiente para manejar, tanto los protocolos como las aplicaciones del usuario.

7.2 Recomendaciones

Cuando se implementen aplicaciones que trabajen con los protocolos TCP/IP, se debe hacer un estudio extenso acerca de los mismos. Tanto en gran cantidad de libros como en internet, se encuentra la documentación necesaria para conocer los protocolos a fondo.

Al trabajar con *MODBUS*, se debe prestar atención en la cantidad de tiempo de espera que se le da a las aplicaciones para recibir las respuestas a los mensajes, ya que los dispositivos que manejan este protocolo son lentos, en cuanto a su tiempo de respuesta. Es mejor poner desde un principio tiempos mayores o iguales a un segundo (1000 ms).

Cuando se trabaja con sistemas que tienen una memoria pequeña, como los sistemas que usan microcontroladores o microprocesadores para aplicaciones propias (no procesadores de uso general), se debe vigilar el tamaño del programa o aplicación que se hace, más cuando se trabaja con el lenguaje C, ya que no se tiene un control real del tamaño, como se podría hacer con ensamblador.

El tamaño final de los programas o aplicaciones que usen rutinas para el manejo de TCP/IP es bastante grande para sistemas con limitaciones de memoria (como los basados en microcontroladores). Si se piensa en hacer un sistema similar al implementado en este proyecto, se debe buscar la mejor opción de hardware, que posea, además de las características que se necesitan para el proyecto, una cantidad de memoria mayor a los 4 kb. Esta recomendación vale también para sistemas que sean programados con el lenguaje C.

A la hora de hacer un proyecto de graduación, cuando se manden comprar partes de cualquier tipo en el extranjero, se debe tener paciencia, pues las entregas muy a menudo se retrasan. Tanto la persona que realiza el proyecto como sus asesores deben coordinar muy bien estas compras, realizando una lista detallada de

los objetos a comprar, los precios y las formas de pago disponibles. También, el realizador del proyecto y los asesores debe estar pendientes del estado de la orden, para saber si ya fue despachada, si está en la aduana, si no ha salido del país de origen, etc. Los retrasos en las entregas determinan la forma en la que, a partir de estos, se llevará a cabo el proyecto; por esto, el realizador debe saber exactamente cuándo llegarán sus pedidos, para hacer los ajustes pertinentes en su cronograma.

BIBLIOGRAFÍA

1. Swales, Andy. **Open MODBUS/TCP Specification: Release 1.0**. Schneider Electric, 1999.

2. **MODICON MODBUS Protocol Reference Guide: PI-MBUS-300 Rev. J**. MODICON Inc., 1996.

3. Miranet Bonet, Juan Salvador. **Protocolos TCP/IP**.

   <http://fredimerino.8k.com/Tcpip.htm > (Marzo 2002)

4. Moreno, Luciano. **El Modelo OSI**.

   <http://www.htmweb.net/redes/osi/osi_1.htm> (Abril 2002)

5. **TCP/IP Development Kit Getting Started Manual**. E.E.U.U.: Rabbit Semiconductors, 2000.

6. **An Introduction to TCP/IP for Embedded Systems Designers**. E.E.U.U.: Z-World Inc., 2001.

7. **Dynamic C Premier for Rabbit Semiconductors Microprocessors: Integrated C Development System User´s Manual**. E.E.U.U.: Z-World Inc., 1999.

8. **Dynamic C TCP/IP User´s Manual**. E.E.U.U.: Z-World Inc., 2001.

APÉNDICES

Apéndice A1: Abreviaturas y acrónimos utilizados en este documento

ARP: Address Resolution Protocol (Protocolo de Resolución de Dirección).

ASCII: American Standard Code for Information Interchange (Código Standard Americano para el Intercambio de Información).

CMOS: Metal Oxid Silicon Complementary (Silicio Complementario de Oxido de Metal).

CRC: Cyclical Redundancy Check (Chequeo Cíclico de Redundancia).

ICMP: Internet Control Message Protocol (Protocolo de Control de Mensajes Internet).

IEEE: Institute of Electrical and Electronics Engineers (Instituto de Ingenieros Eléctricos y Electrónicos).

IP: Internet Protocol (Protocolo de interred).

LAN: Local Area Network (Red de Área Local).

ORC: Osciloscopio de Rayos Catódicos.

OSI: Open Systems Interconection (Interconexión de Sistemas Abiertos)

ROM: Read Only Memory (Memoria de Sólo Lectura).

SCADA: Supervisory control and data acquisition (Control Supervisorio y adquisición de datos).

SRAM: Static Random Access Memory (Memoria de Acceso Aleatorio).

UTR: Unidad Terminal Remota (en inglés RTU, Remote Terminal Unit).

XOR: Exclusive OR (OR Exclusiva).

Apéndice A2: Glosario de términos

Esclavo MODBUS: unidad de recolección de datos y control de procesos, que se comunica con un sistema usando el protocolo MODBUS.

Maestro MODBUS: unidad que controla a uno o más esclavos MODBUS. Esta unidad se encarga de enviar comandos a los esclavos, pedirles información o configurarlos.

Microprocesador: unidad que ejecuta instrucciones de programas. Esta unidad se comunica con otros subsistemas (dispositivos) y a menudo controla su operación. Debido al papel central de tal unidad se conoce como unidad central de procesamiento, o CPU (Central processing unit).

MODBUS: Protocolo de Comunicación abierto, de Modicon, utilizado para PLCs y sistemas SCADA.

MODBUS TCP: Modificación del protocolo MODBUS. Se usa para conectar dispositivos MODBUS a redes que utilicen los protocolos TCP/IP.

PING: Envía una Llamada a una estación de trabajo e informa si se puede establecer conexión o no con ella.

Sistema SCADA Sistema de Control Supervisorio, administración y adquisición de datos. Se utiliza para monitorear y controlar uno o varios procesos físicos, generalmente distribuidos sobre un área geográfica relativamente grande.

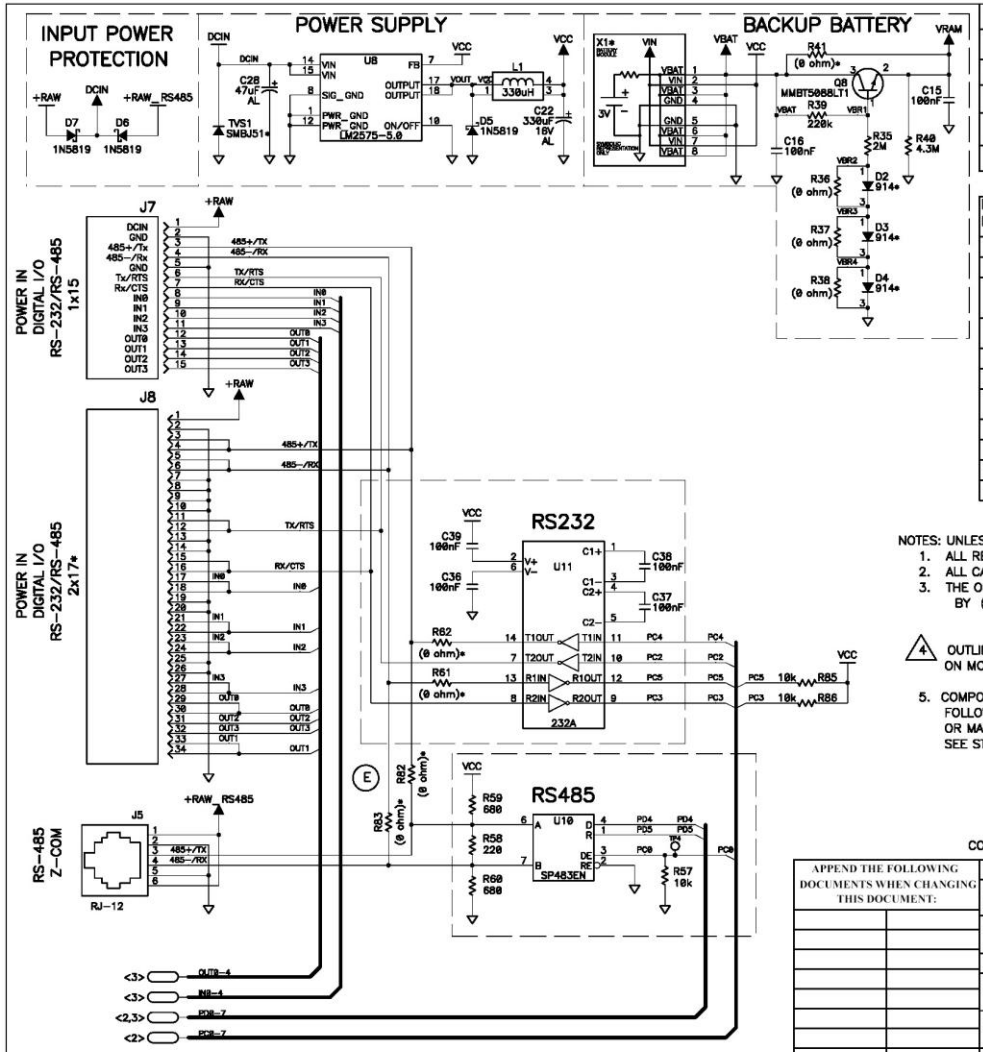
TCP/IP: Es un protocolo que proporciona transmisión fiable de paquetes de datos sobre redes. El nombre TCP / IP Proviene de dos protocolos importantes de la familia, el Transmission Contorl Protocol (TCP) y el Internet Protocol (IP). Todos juntos llegan a ser más de 100 protocolos diferentes definidos en este conjunto.

Unidad terminal remota: Computador de aplicación especial orientado a la toma de datos y control de procesos físicos, subordinado a un computador central.

ANEXOS

Anexo B.1: Esquemáticos del hardware usado en el proyecto

Anexo B.1.1 Esquemáticos de la tarjeta de desarrollo Rabbit 2000 TCP/IP



REVISION HISTORY			REVISION APPROVAL			
REV	ECO	DESCRIPTION	PROJECT ENGINEER	APPROVAL DATE	DOCUMENT CONTROL	APPROVAL DATE
C	E11145	SEE ECO FOR HISTORY	XT	14AUG00	KAH	18AUG00
D	E11180	CHG TOLERANCE OF R35 & R39 FROM 1% TO 5%	LSW	30AUG00	KAH	29AUG00
E	E11320	CORRECT NOTE ON PAGE 3. REFERENCED WRONG COMPONENTS.	RJH	2JAN01	KIS	2JAN01
F	E11342	ADD R82 & R83 TO KEEP RS485 WITH 5-WIRE RS232	EP	5JAN01	KIS	9JAN01

TABLE A

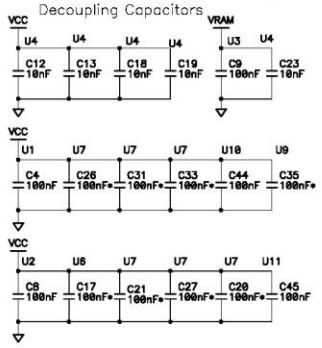
REF DES	DEVICE	DEVICE VOLTAGE INFORMATION					DEVICE: FILTER CAP
		AGND	GND	VCC	VRAM	NO CONNECTS	REF DES(s)
U1	LM4872		2	6			C4
U2	ETC811L						C8
U3	SRAM 512K X 8		15		32		C9
	SRAM 32K X 8		14		28		
U4	RABBIT 2000		2,27,39 52,77,89	3,26,53, 78,92	42		C12,13,18,19 C23 - PIN42 (VRAM)
U5	FLASH		24	8			C17
U6	FLASH		24	8			
U7	RLT8819AS		14,28,44 52,83,86	6,17,47 57,70,89			C20,21,26,27,31,33
U8	LM2575-5.0					2-6.9.11.13.16.19-24	
U9	93C46		5	8		7	C35
U10	SP483E		5	8			C44
U11	232A		15	16			C45

NOTES: UNLESS OTHERWISE SPECIFIED;
1. ALL RESISTOR VALUES ARE IN OHMS, 1/10W, 5%
2. ALL CAPACITORS ARE 50VDC OR HIGHER.
3. THE ORIGINATION SOURCE OF A VOLTAGE IS REPRESENTED BY (VCC), AND ALL REFERENCES TO THAT VOLTAGE ARE REPRESENTED BY (VCC).

4. OUTLINED CIRCUIT MAY NOT BE STUFFED DEPENDING ON MODEL, SEE STUFFING CHART FOR CLARIFICATION.

5. COMPONENT VALUES SHOWN WITH AN ASTERISK (*) FOLLOWING THE VALUE, MAY HAVE DIFFERENT VALUES, OR MAY NOT BE STUFFED DEPENDING ON MODEL. SEE STUFFING CHART FOR CLARIFICATION..

Decoupling Capacitors



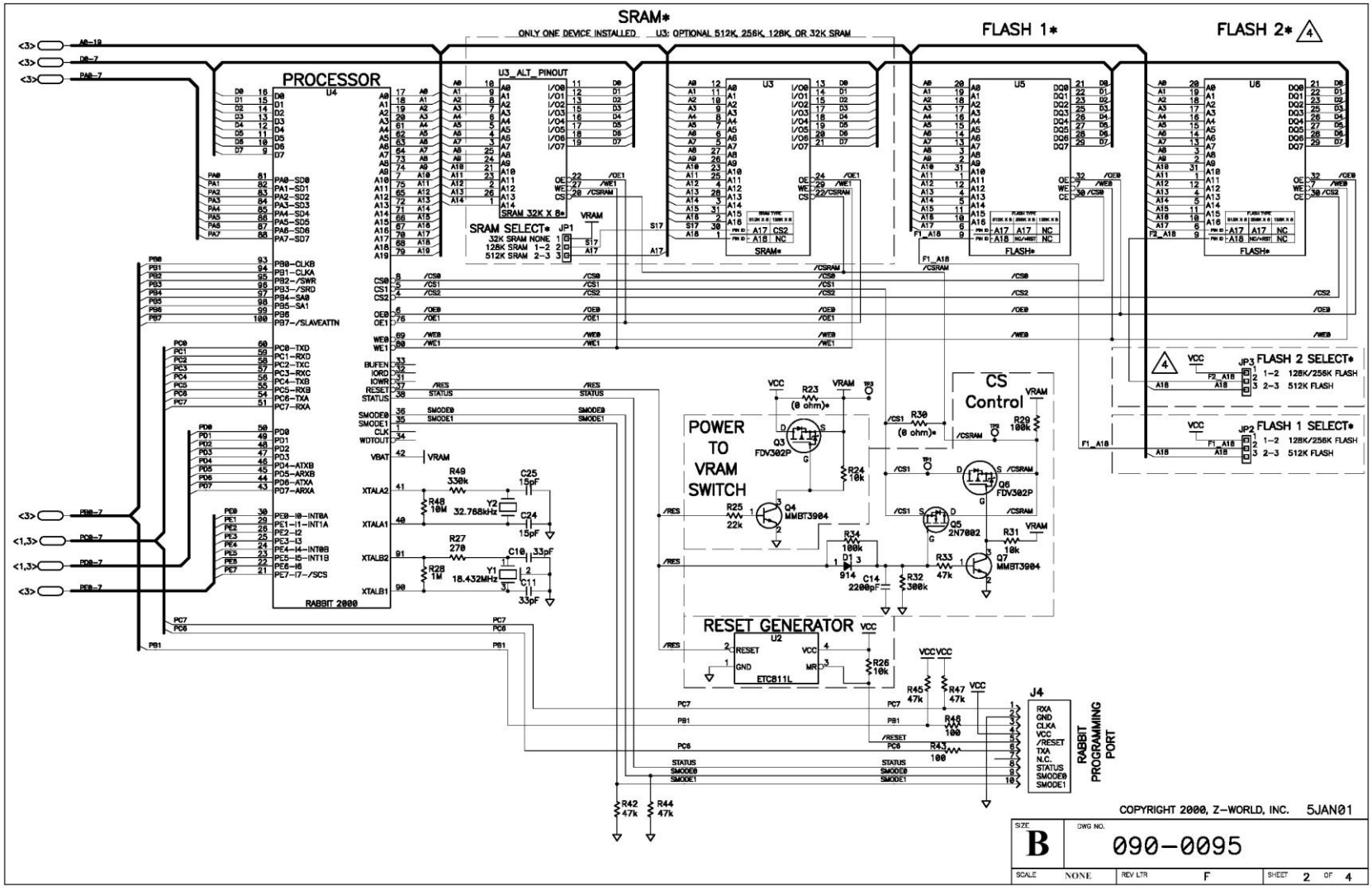
APPEND THE FOLLOWING DOCUMENTS WHEN CHANGING THIS DOCUMENT:		DRAWING CONTENT:	
		DRAWN BY: (INITIAL RELEASE)	
		GEY	14SEP99
		REVISED BY:	
		E.PEAK	5JAN01
		APPROVALS: INITIAL RELEASE	
		PROJECT ENGINEER:	
		RJH	24MAY00
		ENGINEERING MANAGER:	
		SIGNATURES	DATE

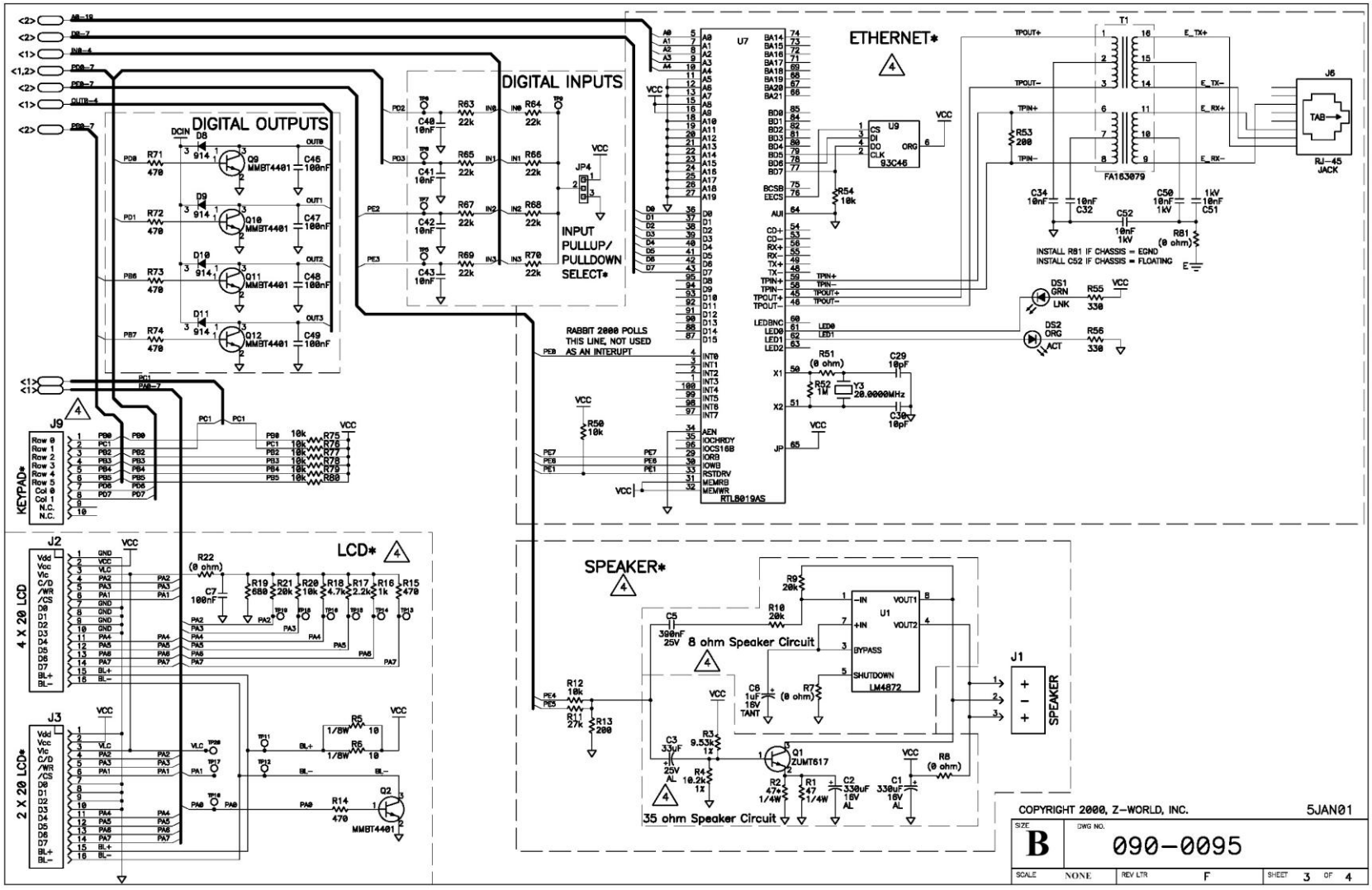
TITLE: SCHEMATIC DIAGRAM OP6600/6700 SERIES & RABBIT 2000 TCP/IP DEVELOPMENT KIT

Z-WORLD
2900 SPAFFORD ST.
DAVIS, CA 95616
530 - 757 - 4616

SIZE: B | DWG NO.: 090-0095

SCALE: NONE | RELEASE DATE: 24MAY00 | SHEET 1 OF 4


SRAM*
ONLY ONE DEVICE INSTALLED U3: OPTIONAL 512K, 256K, 128K, OR 32K SRAM
FLASH 1*
FLASH 2*
PROCESSOR
U4
RABBIT 2000
U3_ALT_PINOUT
SRAM 32K X 8*
SRAM SELECT* JP1
32K SRAM NONE 1
128K SRAM 1-2 2
512K SRAM 2-3 3
U3
SRAM*
U5
FLASH*
U6
FLASH*
FLASH 2 SELECT*
FLASH 1 SELECT*
1-2 128K/256K FLASH
2-3 512K FLASH
CS Control
POWER TO VRAM SWITCH
RESET GENERATOR
U2
ETC811L
J4
RABBIT PROGRAMMING PORT
RXA
GND
CLKA
VCC
/RESET
TXA
N.C.
STATUS
SMODE0
SMODE1
COPYRIGHT 2000, Z-WORLD, INC. 5JAN01
SIZE B
DWG NO. 090-0095
SCALE NONE
REV LTR F
SHEET 2 OF 4


DIGITAL OUTPUTS
DIGITAL INPUTS
ETHERNET*
LCD*
SPEAKER*
KEYPAD*
4 X 20 LCD
2 X 20 LCD*
8 ohm Speaker Circuit
35 ohm Speaker Circuit
INPUT PULLUP/ PULLDOWN SELECT*
RABBIT 2000 POLLS THIS LINE, NOT USED AS AN INTERUPT
INSTALL R81 IF CHASSIS = EGND
INSTALL C52 IF CHASSIS = FLOATING
RTL8019AS
LM4872
93C46
FA163079
RJ-45 JACK
SPEAKER
COPYRIGHT 2000, Z-WORLD, INC.
5JAN01
090-0095
SIZE B
SCALE NONE
REV LTR F
SHEET 3 OF 4

STUFFING TABLE

CIRCUIT		PART	MODEL		
			RABBIT 2000 TCP/IP	OP6700 SERIES	OP6600 SERIES
BACKUP BATTERY	BATTERY MODULE	X1	NOT INSTALLED	INSTALLED	INSTALLED
	MAIN	R40	4.3M	4.3M	4.3M
		R41	NOT INSTALLED	NOT INSTALLED	NOT INSTALLED
		C15	100nF	100nF	100nF
		C16	100nF	100nF	100nF
	REGULATOR	Q8	5088 npn	5088 npn	5088 npn
		R35	2M	2M	2M
		R39	220k	220k	220k
	REGULATOR TEMPERATURE COMPENSATION ADJUST	D2	NOT INSTALLED	NOT INSTALLED	NOT INSTALLED
		D3	NOT INSTALLED	NOT INSTALLED	NOT INSTALLED
		D4	NOT INSTALLED	NOT INSTALLED	NOT INSTALLED
		R36	ZERO ohm	ZERO ohm	ZERO ohm
		R37	ZERO ohm	ZERO ohm	ZERO ohm
		R38	ZERO ohm	ZERO ohm	ZERO ohm
CS CONTROL BYPASS		R30	NOT INSTALLED	NOT INSTALLED	NOT INSTALLED
PWR- VRAM SW BYPASS		R23	NOT INSTALLED	NOT INSTALLED	NOT INSTALLED
PWR SPLY TRANSZORB		TVS1	NOT INSTALLED	SMBJ51	SMBJ51
SRAM	MAIN	U3	128K SRAM	128K SRAM	128K SRAM
		C9	100nF	100nF	100nF
	SRAM SELECT	JP1	ZERO ohm ACROSS PINS 1-2	ZERO ohm ACROSS PINS 1-2	ZERO ohm ACROSS PINS 1-2
FLASH 1	MAIN	U5	256K FLASH	256K FLASH	256K FLASH
	FLASH 1 SELECT	JP2	ZERO ohm ACROSS PINS 1-2	ZERO ohm ACROSS PINS 1-2	ZERO ohm ACROSS PINS 1-2
FLASH 2	MAIN	U6	256K FLASH	256K FLASH	NOT INSTALLED
		C17	100nF	100nF	NOT INSTALLED
	FLASH 2 SELECT	JP3	ZERO ohm ACROSS PINS 1-2	ZERO ohm ACROSS PINS 1-2	NOT INSTALLED
RS485		R58	NOT INSTALLED	220 ohm	220 ohm
RS232		R61	NOT INSTALLED	NOT INSTALLED	NOT INSTALLED
		R62	NOT INSTALLED	NOT INSTALLED	NOT INSTALLED
POWER IN CONN 2 x 17		J8	NOT INSTALLED	NOT INSTALLED	NOT INSTALLED
KEYPAD CONNECTOR		J9	NOT INSTALLED	INSTALLED	INSTALLED
LCD	4 X 20 CONN	J2	NOT INSTALLED	INSTALLED	INSTALLED
	2 X 20 CONN	J3	NOT INSTALLED	NOT INSTALLED	NOT INSTALLED
	MAIN	C7	NOT INSTALLED	100nF	100nF
		Q2	NOT INSTALLED	MMBT4401	MMBT4401
		R5	NOT INSTALLED	10 ohm 1/8W	10 ohm 1/8W
		R6	NOT INSTALLED	10 ohm 1/8W	10 ohm 1/8W
		R14	NOT INSTALLED	470 ohm	470 ohm
		R15	NOT INSTALLED	470 ohm	470 ohm
		R16	NOT INSTALLED	1k	1k
		R17	NOT INSTALLED	2.2k	2.2k
		R18	NOT INSTALLED	4.7k	4.7k
		R19	NOT INSTALLED	680 ohm	680 ohm
		R20	NOT INSTALLED	10k	10k
		R21	NOT INSTALLED	20k	20k
		R22	NOT INSTALLED	ZERO ohm	ZERO ohm

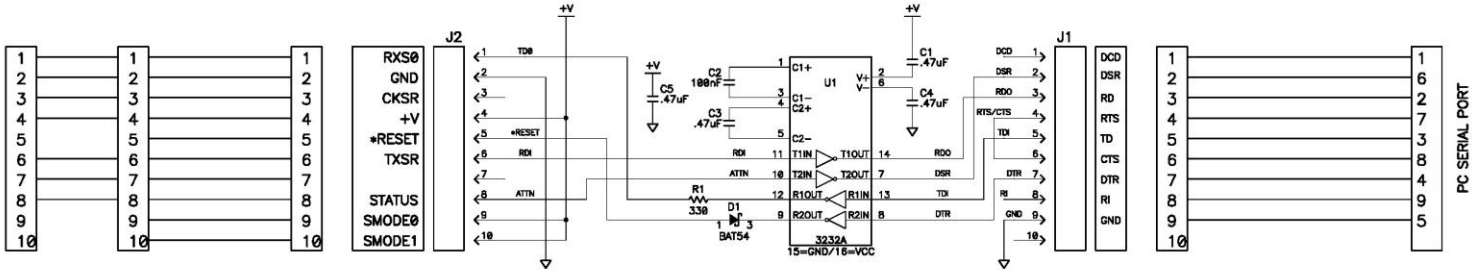
STUFFING TABLE

CIRCUIT		PART	MODEL		
			RABBIT 2000 TCP/IP	OP6700 SERIES	OP6600 SERIES
DIGITAL INPUT PULLUP/ PULLDOWN SELECT		JP4	ZERO ohm ACROSS PINS 1-2	ZERO ohm ACROSS PINS 1-2	ZERO ohm ACROSS PINS 1-2
ETHERNET	MAIN	U7	RLT8019AS	RLT8019AS	NOT INSTALLED
		C20	100nF	100nF	NOT INSTALLED
		C21	100nF	100nF	NOT INSTALLED
		C26	100nF	100nF	NOT INSTALLED
		C27	100nF	100nF	NOT INSTALLED
		C31	100nF	100nF	NOT INSTALLED
		C33	100nF	100nF	NOT INSTALLED
		R50	10k	10k	NOT INSTALLED
	ETHERNET CRYSTAL	Y3	20.0000MHz	20.0000MHz	NOT INSTALLED
		C29	10pF	10pF	NOT INSTALLED
		C30	10pF	10pF	NOT INSTALLED
		R51	ZERO ohm	ZERO ohm	NOT INSTALLED
		R52	1 MEG	1 MEG	NOT INSTALLED
	EEPROM	U9	93C46	93C46	NOT INSTALLED
		C35	100nF	100nF	NOT INSTALLED
		R54	10k	10k	NOT INSTALLED
	LEDS	DS1	GRN	GRN	NOT INSTALLED
		DS2	ORG	ORG	NOT INSTALLED
		R55	330 ohm	330 ohm	NOT INSTALLED
		R56	330 ohm	330 ohm	NOT INSTALLED
	TRANSFORMER	T1	FA163079	FA163079	NOT INSTALLED
		C32	10nf	10nf	NOT INSTALLED
		C34	10nf	10nf	NOT INSTALLED
		C50	10nf 1KV	10nf 1KV	NOT INSTALLED
		C51	10nf 1KV	10nf 1KV	NOT INSTALLED
		C52	10nf 1KV	10nf 1KV	NOT INSTALLED
		J6	RJ-45	RJ-45	NOT INSTALLED
		R53	200 ohm	200 ohm	NOT INSTALLED
		R81	ZERO ohm	ZERO ohm	NOT INSTALLED
SPEAKER	MAIN	R11	NOT INSTALLED	27k	27k
		R12	NOT INSTALLED	10k	10k
		R13	NOT INSTALLED	200 ohm	200 ohm
	35 ohm CIRCUIT	C1	NOT INSTALLED	330uF 16V	330uF 16V
		C2	NOT INSTALLED	330uF 16V	330uF 16V
		C3	NOT INSTALLED	33uF 25V	33uF 25V
		J1	NOT INSTALLED	INSTALLED	INSTALLED
		Q1	NOT INSTALLED	ZUMT617	ZUMT617
		R1	NOT INSTALLED	47 ohm 1/4W	47 ohm 1/4W
		R2	NOT INSTALLED	NOT INSTALLED	NOT INSTALLED
		R3	NOT INSTALLED	9.53k 1%	9.53k 1%
		R4	NOT INSTALLED	10.2k 1%	10.2k 1%
		R8	NOT INSTALLED	ZERO ohm	ZERO ohm
	8 ohm CIRCUIT		NOT INSTALLED	NOT INSTALLED	NOT INSTALLED

SIZE	DWG NO.		
B	090-0095		
SCALE NONE	REV LTR F	SHEET 4 OF 4	

Anexo B.1.2 Esquemático del cable de programación serial usado en el proyecto

REVISION HISTORY			REVISION APPROVAL			
REV	ECO	DESCRIPTION	PROJECT ENGINEER	APPROVAL DATE	DOCUMENT CONTROL	APPROVAL DATE
A	E11523	INITIAL RELEASE OF SCHEMATIC	EP	5/14/01	KIS	5/14/01
B	E11691	CORRECT DE9 PINOUT	EP	10/5/01	KIS	10/4/01
C	E11816	ADD 3.3V CAPABILITY AND RED TUBING				



NOTES: UNLESS OTHERWISE SPECIFIED;
1. ALL RESISTOR VALUES ARE IN OHMS, 1/10W, 5%
2. ALL CAPACITORS ARE 50VDC OR HIGHER.
3. THE ORIGINATION SOURCE OF A VOLTAGE IS REPRESENTED BY (V_{CC}), AND ALL REFERENCES TO THAT VOLTAGE ARE REPRESENTED BY (V_{CC}).

APPEND THE FOLLOWING DOCUMENTS WHEN CHANGING THIS DOCUMENT:		DRAWING CONTENT:		TITLE	
		DRAWN BY: (INITIAL RELEASE) E. PEAK	14MAY01	SCHEMATIC DIAGRAM RABBIT PROG. CABLE 3.3V – 5.0V	2900 SPAFFORD ST. DAVIS, CA 95616 530 - 757 - 4616
		REVISED BY: D MUSGROVE	01/21/02		
		APPROVALS: INITIAL RELEASE			
		PROJECT ENGINEER: ERIC PEAK	5/14/01	SIZE B	DWG NO. 090–0128
		ENGINEERING MANAGER: R MATTHEWS	5/14/01		
		SIGNATURES	DATE	SCALE NONE	RELEASE DATE SHEET 1 OF 1

Anexo B.2 Protocolos de red: Protocolos TCP/IP

Una *red* es una configuración de computadora que intercambia información. Pueden proceder de una variedad de fabricantes y es probable que tenga diferencias tanto en hardware como en software, para posibilitar la comunicación entre estas es necesario un conjunto de reglas formales para su interacción. A estas reglas se les denominan protocolos.

Un *protocolo* es un conjunto de reglas establecidas entre dos dispositivos para permitir la comunicación entre ambos.

B.2.1 DEFINICION TCP / IP

Se han desarrollado diferentes familias de protocolos para comunicación por red de datos para los sistemas UNIX. El más ampliamente utilizado es el Internet Protocol Suite, comúnmente conocido como TCP / IP.

Es un protocolo DARPA que proporciona transmisión fiable de paquetes de datos sobre redes. El nombre TCP / IP Proviene de dos protocolos importantes de la familia, el Transmission Contorl Protocol (TCP) y el Internet Protocol (IP). Todos juntos llegan a ser más de 100 protocolos diferentes definidos en este conjunto.

El TCP / IP es la base del Internet que sirve para enlazar computadoras que utilizan diferentes sistemas operativos, incluyendo PC, minicomputadoras y computadoras centrales sobre redes de área local y área extensa. TCP / IP fue desarrollado y demostrado por primera vez en 1972 por el departamento de defensa de los Estados Unidos, ejecutándolo en el ARPANET una red de área extensa del departamento de defensa.

B.2.2 LAS CAPAS CONCEPTUALES DEL SOFTWARE DE PROTOCOLOS

Pensemos los módulos del software de protocolos en una pila vertical constituida por capas. Cada capa tiene la responsabilidad de manejar una parte del problema.

B.2.3 RED

Conceptualmente, enviar un mensaje desde un programa de aplicación en una máquina hacia un programa de aplicaciones en otra, significa transferir el mensaje hacia abajo, por las capas sucesivas del software de protocolo en la maquina emisora, transferir un mensaje a través de la red y luego, transferir el mensaje hacia arriba, a través de las capas sucesivas del software de protocolo en la maquina receptora.

En la práctica, el software es mucho más complejo de lo que se muestra en el modelo. Cada capa toma decisiones acerca de lo correcto del mensaje y selecciona una acción apropiada con base en el tipo de mensaje o la dirección de destino. Por ejemplo, una capa en la maquina de recepción debe decidir cuándo tomar un mensaje o enviarlo a otra maquina. Otra capa debe decidir que programa de aplicación deberá recibir el mensaje.

B.2.4 FUNCIONALIDAD DE LAS CAPAS

Una vez que se toma la decisión de subdividir los problemas de comunicación en cuatro subproblemas y organizar el software de protocolo en módulos, de manera que cada uno maneja un problema, surge la pregunta. "¿Qué tipo de funciones debe instalar en cada modulo?". La pregunta no es fácil de responder por varias razones. En primer lugar, un grupo de objetivos y condiciones determinan un problema de comunicación en particular, es posible elegir una organización que optimice un software de protocolos para ese problema. Segundo, incluso cuando se consideran los servicios generales al nivel de red, como un transporte confiable es posible

seleccionar entre distintas maneras de resolver el problema. Tercero, el diseño de una arquitectura de red y la organización del software de protocolo esta interrelacionado; no se puede diseñar a uno sin considera al otro.

B.2.5 MODELO DE REFERENCIA ISO (OSI) DE 7 CAPAS

Existen dos modelos dominantes sobre la estratificación por capas de protocolo. La primera, basada en el trabajo realizado por la International Organization for Standardization (Organización para la Estandarización o ISO, por sus siglas en inglés), conocida como Referencia Model of Open System Interconnection Modelo de referencia de interconexión de sistemas abiertos) de ISO, denominada frecuentemente modelo ISO.El modelo ISO, elaborado para describir protocolos para una sola red, no contiene un nivel especifico para el ruteo en el enlace de redes, como sucede con el protocolo TCP/IP.

B.2.6 EL MODELO DE ESTRATIFICACIÓN POR CAPAS DE TCP/IP DE INTERNET

El modelo de estratificación por capas no se origina de un comité de estándares, sino que proviene de las investigaciones que se realizan respecto al conjunto de protocolos de TCP/IP. Con un poco de esfuerzo, el modelo ISO puede ampliarse y describir el esquema de estratificación por capas del TCP/IP, pero los presupuestos subyacentes son lo suficientemente distintos para distinguirlos como dos diferentes.

- Capa de aplicación. Es el nivel mas alto, los usuarios llaman a una aplicación que acceda servicios disponibles a través de la red de redes TCP/IP. Una aplicación interactúa con uno de los protocolos de nivel de transporte para enviar o recibir datos. Cada programa de aplicación selecciona el tipo de transporte necesario, el cual puede ser una secuencia de mensajes individuales o un flujo contínuo de octetos. El programa de aplicación pasa los datos en la forma requerida hacia el nivel de transporte para su entrega.

- Capa de transporte. La principal tarea de la capa de transporte es proporcionar la comunicación entre un programa de aplicación y otro. Este tipo de comunicación se conoce frecuentemente como comunicación punto a punto. La capa de transporte regula el flujo de información. Puede también proporcionar un transporte confiable, asegurando que los datos lleguen sin errores y en secuencia. Para hacer esto, el software de protocolo de transporte tiene el lado de recepción enviando acuses de recibo de retorno y la parte de envío retransmitiendo los paquetes perdidos. El software de transporte divide el flujo de datos que se está enviando en pequeños fragmentos (por lo general conocidos como paquetes) y pasa cada paquete, con una dirección de destino, hacia la siguiente capa de transmisión. Aun cuando en el esquema anterior se utiliza un solo bloque para representar la capa de aplicación, una computadora de propósito general puede tener varios programas de aplicación accesando la red de redes al mismo tiempo. La capa de transporte debe aceptar datos desde varios programas de usuario y enviarlos a la capa del siguiente nivel. Para hacer esto, se añade información adicional a cada paquete, incluyendo códigos que identifican qué programa de aplicación envía y qué programa debe recibir, así como una suma de verificación para verificar que el paquete ha llegado intacto y utiliza el código de destino para identificar el programa de aplicación en el que se debe entregar.

- Capa Internet. La capa Internet maneja la comunicación de una máquina a otra. Ésta acepta una solicitud para enviar un paquete desde la capa de transporte, junto con una identificación de la máquina, hacia la que se debe enviar el paquete. La capa Internet también maneja la entrada de datagramas, verifica su validez y utiliza un algoritmo de ruteo para decidir si el datagrama debe procesarse de manera local o debe ser transmitido. Para el caso de los datagramas direccionados hacia la máquina local, el software de la capa de red de redes borra el encabezado del datagrama y selecciona, de entre varios

protocolos de transporte, un protocolo con el que manejará el paquete. Por último, la capa Internet envía los mensajes ICMP de error y control necesarios y maneja todos los mensajes ICMP entrantes.

- Capa de interfaz de red. El software TCP/IP de nivel inferior consta de una capa de interfaz de red responsable de aceptar los datagramas IP y transmitirlos hacia una red específica. Una interfaz de red puede consistir en un dispositivo controlador (por ejemplo, cuando la red es una red de área local a la que las máquinas están conectadas directamente) o un complejo subsistema que utiliza un protocolo de enlace de datos propios (por ejemplo, cuando la red consiste de conmutadores de paquetes que se comunican con anfitriones utilizando HDLC).

B.2.7 EL PRINCIPIO DE LA ESTRATIFICACION POR CAPAS DE PROTOCOLOS

Independientemente del esquema de estratificación por capas que se utilice o de las funciones de las capas, la operación de los protocolos estratificados por capas se basa en una idea fundamental. Los protocolos estratificados por capas están diseñados de modo que una capa n en el receptor de destino reciba exactamente el mismo objeto enviado por la correspondiente capa n de la fuente.

El principio de estratificación por capas explica por que la estratificación por capas es una idea poderosa. Esta permite que el diseñador de protocolos enfoque su atención hacia una capa a la vez, sin preocuparse acerca del desempeño de las capas inferiores. Por ejemplo, cuando se construye una aplicación para transferencia de archivos, el diseñador piensa solo en dos copias del programa de aplicación que se correrá en dos máquinas y se concentrará en los mensajes que se necesitan intercambiar para la transferencia de archivos. El diseñador asume que la aplicación en el anfitrión receptor es exactamente la misma que en el anfitrión emisor.

B.2.8 LA DESVENTAJA DE LA ESTRATIFICACIÓN POR CAPAS

La estratificación por capas es una idea fundamental que proporciona las bases para el diseño de protocolos. Permite al diseñador dividir un problema complicado en subproblemas y resolver cada parte de manera independiente. Por desgracia, el software resultante de una estratificación por capas estrictas puede ser muy ineficaz. Si se considera el trabajo de la capa de transporte, debe aceptar un flujo de octetos desde un programa de aplicación, dividir el flujo en paquetes y enviar cada paquete a través de la red de redes. Para optimizar la transferencia, la capa de transporte debe seleccionar el tamaño de paquete más grande posible que le permita a un paquete viajar en una trama de red. En particular, si la máquina de destino está conectada a una máquina de la misma red de la fuente, solo la red física se verá involucrada en la transferencia, así, el emisor puede optimizar el tamaño del paquete para esta red. Si el software preserva una estricta estratificación por capas, sin embargo, la capa de transporte no podrá saber como ruteará el módulo de Internet el tráfico o que redes están conectadas directamente. Más aún, la capa de transporte no comprenderá el datagrama o el formato de trama ni será capaz de determinar como deben ser añadidos muchos octetos de encabezado a un paquete. Así, una estratificación por capas estricta impedirá que la capa de transporte optimice la transferencia.

Por lo general, las implantaciones atenúan el esquema estricto de la estratificación por capas cuando construyen software de protocolo. Permiten que información como la selección de ruta y la MTU de red se propaguen hacia arriba. Cuando los buffers realizan el proceso de asignación, generalmente dejan espacio para encabezados que serán añadidos por los protocolos de las capas de bajo nivel y pueden retener encabezados de las tramas entrantes cuando pasan hacia protocolos de capas superiores. Tal optimización puede producir mejoras notables en la eficiencia siempre y cuando conserve la estructura básica en capas.

B.2.9 COMANDOS TCP/IP

TCP/IP incluye dos grupos de comandos utilizados para suministrar servicios de red:

- Los comandos remotos BERKELEY
- Los comandos DARPA

Los comandos remotos BERKELEY, que fueron desarrollados en la Universidad Berkeley (California), incluyen órdenes para comunicaciones entre sistemas operativos UNIX, como copia remota de archivos, conexión remota, ejecución de shell remoto, etc.

Permiten utilizar recursos con otros hosts, pudiendo tratar distintas redes como si fueran una sola.

En la versión 4 para UNIX Sistema V, se pueden distinguir los siguientes comandos más comunes:

- RCP Realiza una copia de archivos al mismo o a otro servidor
- RLOGINGL-RLOGINVT Se utiliza para hacer una conexión al mismo o a otro servidor
- REXEC-RSH Permite ejecutar comandos del sistema operativo en el mismo o en otro servidor.

Los comandos DARPA incluyen facilidades para emulación de terminales, transferencia de archivos, correo y obtención de información sobre usuarios. Pueden ser utilizadas kpara comunicación con computadores que ejecutan distintos sistemas operativos.

En la versión 2.05 para DOS, dependiendo de las funciones que realizan, se pueden distinguir los siguientes grupos de comandos:

- Kernel PC/TCP y herramientas asociadas

  Se utilizan para cargar el núcleo TCP/IP en la memoria del computador.

  BOOTP Asigna la dirección IP de la estación de trabajo

  INET Descarga el núcleo PC/TCP de la memoria y/o realiza estadísticas de red

  KERNEL Carga el núcleo TCP/IP en la memoria y lo deja residente

- Configuración de la red

  Permiten configurar TCP/IP con determinados parámetros.

  IFCONFIG Configura el hardware para TCP/IP

  IPCONFIG Configura el software TCP/IP y la dirección IP

- Transferencia de archivos

  Se utilizan para transferir archivos entre distintos computadoras.

  DDAT'ES Muestra las fechas y horas guardadas en un archivo creado con el comando TAR

  FTP Transfiere archivos entre una estación de trabajo y un servidor

  FRPSRV Convierte una estación de trabajo en un servidor FTP

  PASSWD Se utiliza para poner contraseñas en las estaciones de trabajo a los usuarios para poder utilizar el comando FTPSRV

RMT Permite realizar copia de archivos en una unidad de cinta

TAR Realiza una copia de archivos creando un único archivo de BACKUP

TFTP Transfiere archivos entre una estación de trabajo un servidor o a otra estación de trabajo sin necesidad de validar al usuario

- Impresión

  Permiten el control de la impresión en las impresoras conectadas al servidor.

  DOPREDIR Imprime un trabajo de impresión que aún no ha sido impreso

  IPRINT Envía un texto o un archivo a un servidor de impresoras de imagen

  LPQ Indica el estado de la cola de impresión indicada

  LPR Envía un texto o un archivo a una impresora local o de red.

  LPRM Elimina trabajos pendientes de la cola de impresión

  ONPREDIR Realiza tareas de configuración para el comando PREDIR

  PREDIR Carga o descarga el programa que permite la impresión remota y lo deja residente.

  PRINIT Se usa con los comandos PREDIR y ONPREDIR

  PRSTART Indica a la estación de trabajo remota que imprima un archivo usando la configuración por defecto

- Conexión a servidores

  Permiten la conexión de los computadores a servidores de nuestra red.

  SUPDUP Permite conectarse a otro servidor de la red

TELNET - TN Es el método normal de conectarse a un servidor de la red

- Información sobre los usuarios

  Muestran información sobre los usuarios conectados a la red.

  FINGER Muestra información sobre un usuario conectado a otra estación de trabajo

  NICKNAME Muestra información sobre un usuario o sobre un servidor solicitada al centro de información de redes

  WHOIS Muestra información sobre un usuario registrado que esté conectado a otra estación de trabajo

- Envío y recepción de correo

  Estos comandos permiten el envío y/o recepción de correo entre los usuarios de la red.

  MAIL Permite enviar y recibir correo en la red

  PCMAIL Permite leer correo. Se ha de usar con el comando VMAIL

  POP2 - POP3 Se utiliza para leer correo. Se han de usar con VMAIL Y SMTP

  SMTP Se utiliza para enviar correo en la red

  SMTPSRV Permite leer el correo recibido

  VMAIL Es un comando que muestra una pantalla preparada para leer el correo recibido. Se utiliza en conjunción con los comandos PCMAIL, POP2 0 POP3

- Chequeo de la red

  Permiten chequear la red cuando aparecen problemas de comunicaciones.

HOST Indica el nombre y la dirección IP de una estación de trabajo determinada

PING Envía una Llamada a una estación de trabajo e informa si se puede establecer conexión o no con ella

SETCLOCK Muestra la fecha y la hora que tiene la red

B.2.10 COMO FUNCIONA TCP/IP

Una red TCP/IP transfiere datos mediante el ensamblaje de bloques de datos en paquetes, cada paquete comienza con una cabecera que contiene información de control; tal como la dirección del destino, seguido de los datos. Cuando se envía un archivo por la red TCP/IP, su contenido se envía utilizando una serie de paquetes diferentes. El Internet protocol (IP), un protocolo de la capa de red, permite a las aplicaciones ejecutarse transparentemente sobre redes interconectadas. Cuando se utiliza IP, no es necesario conocer que hardware se utiliza, por tanto ésta corre en una red de área local.

El Transmissión Control Protocol (TCP); un protocolo de la capa de transporte, asegura que los datos sean entregados, que lo que se recibe, sea lo que se pretendía enviar y que los paquetes que sean recibidos en el orden en que fueron enviados. TCP terminará una conexión si ocurre un error que haga la transmisión fiable imposible.

B.2.11 ADMINISTRACION TCP/IP

TCP/IP es una de las redes más comunes utilizadas para conectar computadoras con sistema UNIX. Las utilidades de red TCP/IP forman parte de la versión 4, muchas facilidades de red como un sistema UUCP, el sistema de correo, RFS y NFS, pueden utilizar una red TCP/CP para comunicarse con otras máquinas.

Para que la red TCP/IP esté activa y funcionando será necesario:

- Obtener una dirección Internet.
- Instalar las utilidades Internet en el sistema
- Configurar la red para TCP/IP
- Configurar los guiones de arranque TCP/IP
- Identificar otras máquinas ante el sistema
- Configurar la base de datos del o y ente de STREAMS
- Comenzar a ejecutar TCP/IP.

B.2.12 ¿QUÉ ES INTERNET?

Internet es una red de computadoras que utiliza convenciones comunes a la hora de nombrar y direccionar sistemas. Es una colecciona de redes independientes interconectadas; no hay nadie que sea dueño o active Internet al completo.

Las computadoras que componen Internet trabajan en UNIX, el sistema operativo Macintosh, Windows 95 y muchos otros. Utilizando TCP/IP y los protocolos veremos dos servicios de red:

- Servicios de Internet a nivel de aplicación
- Servicios de Internet a nivel de red

B.2.13 SERVICIOS DE INTERNET A NIVEL DE APLICACIÓN:

Desde el punto de vista de un usuario, una red de redes TCP/IP aparece como un grupo de programas de aplicación que utilizan la red para llevar a cabo tareas útiles de comunicación. Utilizamos el término interoperabilidad para referirnos a la habilidad que tienen diversos sistemas de computación para cooperar en la resolución de problemas computacionales. Los programas de aplicación de Internet

muestran un alto grado de interoperabilidad. La mayoría de usuarios que accesan a Internet lo hacen al correr programas de aplicación sin entender la tecnología TCP/IP, la estructura de la red de redes subyacente o incluso sin entender el camino que siguen los datos hacia su destino. Sólo los programadores que crean los programas de aplicación de red necesitan ver a la red de redes como una red, así como entender parte de la tecnología. Los servicios de aplicación de Internet más populares y difundidos incluyen:

- Correo electrónico. El correo electrónico permite que un usuario componga memorandos y los envíe a individuos o grupos. Otra parte de la aplicación de correo permite que un usuario lea los memorandos que ha recibido. El correo electrónico ha sido tan exitoso que muchos usuarios de Internet depende de él para su correspondencia normal de negocios. Aunque existen muchos sistemas de correo electrónico, al utilizar TCP/IP se logra que la entrega sea más confiable debido a que non se basa en compradoras intermedias para distribuir los mensajes de correo. Un sistema de entrega de correo TCP/IP opera al hacer que la máquina del transmisor contacte directamente la máquina del receptor. Por lo tanto, el transmisor sabe que, una vez que el mensaje salga de su máquina local, se habrá recibido de manera exitosa en el sitio de destino.

- Transferencia de archivos. Aunque los usuarios algunas veces transfieren archivos por medio del correo electrónico, el correo está diseñado principalmente para mensajes cortos de texto. Los protocolos TCP/IP incluyen un programa de aplicación para transferencia de archivos, el cual permite que lo usuarios envíen o reciban archivos arbitrariamente grandes de programas o de datos. Por ejemplo, al utilizar el programa de transferencia de archivos, se puede copiar de una máquina a otra una gran base de datos que contenga imágenes de satélite, un programa escrito en Pascal o C++, o un diccionario del idioma inglés. El sistema proporciona una manera de verificar que los usuarios cuenten con autorización o, incluso, de impedir el acceso. Como el

correo, la transferencia de archivos a través de una red de redes TCP/IP es confiable debido a que las dos máquinas comprendidas se comunican de manera directa, sin tener que confiar en máquinas intermedias para hacer copias del archivo a lo largo del camino.

- Acceso remoto. El acceso remoto permite que un usuario que esté frente a una computadora se conecte a una máquina remota y establezca una sesión interactiva. El acceso remoto hace aparecer una ventana en la pantalla del usuario, la cual se conecta directamente con la máquina remota al enviar cada golpe de tecla desde el teclado del usuario a una máquina remota y muestra en la ventana del usuario cada carácter que la computadora remota lo genere. Cuando termina la sesión de acceso remoto, la aplicación regresa al usuario a su sistema local.

B.2.14 SERVICIOS DE INTERNET A NIVEL DE RED

Un programador que crea programas de aplicación que utilizan protocolos TCP/IP tiene una visión totalmente diferente de una red de redes, con respecto a la visión que tiene un usuario que únicamente ejecuta aplicaciones como el correo electrónico. En el nivel de red, una red de redes proporciona dos grandes tipos de servicios que todos los programas de aplicación utilizan. Aunque no es importante en este momento entender los detalles de estos servicios, no se deben omitir del panorama general del TCP/IP:

- Servicio sin conexión de entrega de paquetes. La entrega sin conexión es una abstracción del servicio que la mayoría de las redes de conmutación de paquetes ofrece. Simplemente significa que una red de redes TCP/IP rutea mensajes pequeños de una máquina a otra, basándose en la información de dirección que contiene cada mensaje. Debido a que el servicio sin conexión rutea cada paquete por separado, no garantiza una entrega confiable y en

orden. Como por lo general se introduce directamente en el hardware subyacente, el servicio sin conexión es muy eficiente.

- Servicio de transporte de flujo confiable. La mayor parte de las aplicaciones necesitan mucho más que sólo la entrega de paquetes, debido a que requieren que el software de comunicaciones se recupere de manera automática de los errores de transmisión, paquetes perdidos o fallas de conmutadores intermedios a lo largo del camino entre el transmisor y el receptor. El servicio de transporte confiable resuelve dichos problemas. Permite que una aplicación en una computadora establezca una "conexión" con una aplicación en otra computadora, para después enviar un gran volumen de datos a través de la conexión como si fuera perramente y directa del hardware.

Muchas redes proporcionan servicios básicos similares a los servicios TCP/IP, pero existen unas características principales que los distingue de los otros servicios:

- Independencia de la tecnología de red. Ya que el TCP/IP está basado en una tecnología convencional de conmutación de paquetes, es independiente de cualquier marca de hardware en particular. La Internet global incluye una variedad de tecnologías de red que van de redes diseñadas para operar dentro de un solo edificio a las diseñadas para abarcar grandes distancias. Los protocolos TCP/IP definen la unidad de transmisión de datos, llamada datagrama, y especifican cómo transmitir los datagramas en una red en particular.

- Interconexión universal. Una red de redes TCP/IP permite que se comunique cualquier par de computadoras conectadas a ella. Cada computadora tiene asignada una dirección reconocida de manera universal dentro de la red de

redes. Cada datagrama lleva en su interior las direcciones de destino para tomar decisiones de ruteo.

- Acuses de recibo punto-a-punto. Los protocolos TCP/IP de una red de redes proporcionan acuses de recibo entre la fuente y el último destino en vez de proporcionarlos entre máquinas sucesivas a lo largo del camino, aún cuando las dos máquinas no estén conectadas a la misma red física.

- Estándares de protocolo de aplicación. Además de los servicios básicos de nivel de transporte (como las conexiones de flujo confiable), los protocolos TCP/IP incluyen estándares para muchas aplicaciones comunes, incluyendo correo electrónico, transferencia de archivos y acceso remoto. Por lo tanto, cuando se diseñan programas de aplicación que utilizan el TCP/IP, los programadores a menudo se encuentran con que el software ya existente proporciona los servicios de comunicación que necesitan.