ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"Comunicación Ethernet con microprocesador NIOS II."

TESINA DE SEMINARIO

Previa la obtención del Título de:

INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

INGENIERO EN TELEMÁTICA

Presentado por:

Carlos Moisés Díaz Espinoza

Vinicio Ramón Vera Vera

GUAYAQUIL – ECUADOR

AÑO 2014

AGRADECIMIENTO

Ante todo agradezco a Dios por darme la oportunidad de terminar este proyecto, por iluminarme siempre y nunca desampararme en los momentos difíciles, a mi familia en especial a mi madre Miriam por brindarme su apoyo incondicional en todo momento y mi hermano Daniel por ayudarme a superar los obstáculos, a mis amigas Alejandra y Paola que siempre estuvieron pendiente de mi, a mi colega Vinicio por ser un gran apoyo para elaborar el proyecto y a las personas que conocí en el trayecto de mi carrera de las cuales aprendí muchas cosas, a todos ellos gracias.

Un especial agradecimiento a mi tutor el Ing. Ronald Ponguillo por confiar en mí y darme la oportunidad de finalizar este proyecto.

Carlos Moisés Díaz Espinoza

Agradezco a Dios por haberme acompañado y guiado a lo largo de toda mi carrera, a mis padres Ramón y Consuelo por apoyarme en todo momento, por los valores que me han inculcado y por haberme dado la oportunidad de tener una excelente educación, a mis hermanas Melissa y Nicole por ser parte importante en mi vida, representar la unidad familiar y llenar mi vida de alegrías y amor cuando más lo he necesitado.

Debo agradecer de manera especial y sincera al Ing. Ronald Ponguillo por creer en Carlos y en mí y habernos brindado la oportunidad de desarrollar este proyecto.

Vinicio Ramón Vera Vera

DEDICATORIAS

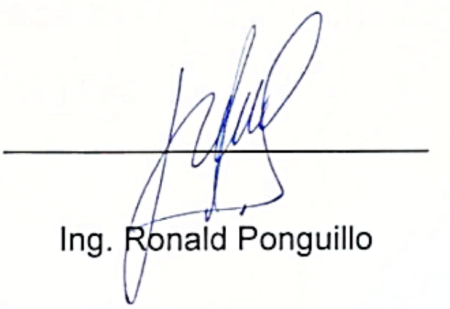
Dedicado a Miriam, mi madre, por creer en mi y apoyarme siempre, a Alejandra, mi amiga, por estar en el momento que mas la necesitaba y a la que guardo un aprecio muy especial y a Mariuxi, mi gata, que por ironías de la vida, siempre me brinda su compañía en las noches de gran esfuerzo.

Carlos Moisés Díaz Espinoza

A Dios, mi familia y personas que me brindaron su apoyo durante el desarrollo de este proyecto.

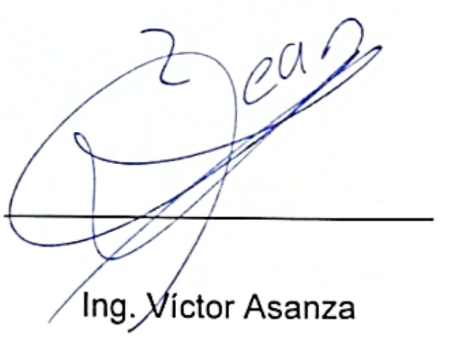
Vinicio Ramón Vera Vera

TRIBUNAL DE SUSTENTACIÓN



Ing. Ronald Ponguillo

PROFESOR DEL SEMINARIO DE GRADUACIÓN



Ing. Víctor Asanza

PROFESOR DELEGADO POR EL DECANO DE LA FACULTAD

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de este trabajo, nos corresponde exclusivamente; y el patrimonio intelectual del mismo a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL". (Reglamento de exámenes y títulos profesionales de la ESPOL).

Carlos Moisés Díaz Espinoza

Vinicio Ramón Vera Vera

RESUMEN

Este documento, contiene información necesaria para desarrollar una aplicación en un sistema embebido configurable, utilizando el protocolo estándar de comunicación ethernet, para enviar y recibir datos, desde tarjetas de desarrollo DE2 con microprocesador NIOS II, hacia una computadora servidor y viceversa, para implementar un sistema de autenticación de usuarios.

Los recursos que se utilizan de las tarjetas DE2 son: conectores GPIO para acoplar un teclado matricial 4x4, LCD, botoneras, Leds, y el chip DM9000 del puerto ethernet. Adicionalmente, los conceptos que se usan como base para desarrollar este proyecto son: algoritmos de cifrado de datos, el modelo de capas TCP/IP, los protocolos Ethernet, ARP, IP y UDP.

Para la configuración del hardware, se utiliza el software Qsys, en Quartus II. La programación del procesador NIOS II, se elabora en lenguaje C++, en Eclipse. La interfaz gráfica en la computadora, se escribe en lenguaje Java, en

NetBeans. La gestión de la base de datos, se trabaja con MySQL. Por último, utilizamos Wireshark para realizar un sniffing de paquetes.

Finalmente, el sistema es probado en un ambiente de laboratorio, creando con un switch D-Link de 5 puertos, las tarjetas DE2 y la computadora una pequeña red de área local (LAN) para realizar la transmisión y recepción de datos.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ABREVIATURAS

ABAP	Programación Avanzada de Aplicaciones de Negocio
AC/DC	Corriente Alterna/Corriente Directa
API	Interfaz de Programación de Aplicaciones
ARES	Software de Edición y Enrutamiento Avanzado
ARP	Protocolo de Resolución de Direcciones
ASCII	Código Americano Estándar para el Intercambio de Información
ATM	Modo de Transferencia Asíncrono
AUTO-MDIX	Interfaz Automática Cruzada Dependiente del Medio
AWT	Kit de herramientas de ventana abstracta
BIOS-ROM	Sistema Básico de Entrada y Salida - Memoria de Solo Lectura
BPTR	Umbral de Presión Posterior
BSP	Paquete de Apoyo de la Tarjeta

CAD	Diseño Asistido por Computadora
CMOS-RAM	Semiconductor Complementario de Óxido Metálico – Memoria de Acceso Aleatorio
COBOL	Lenguaje Común Orientado a Negocios
CRC	Comprobación de Redundancia Cíclica
CSMA/CD	Acceso Múltiple con Escucha de Portadora y Detección de Colisiones
DBMS	Sistema de Gestión de Base de Datos
DE2	Tarjeta de Desarrollo y Educación
DNS	Sistema de Nombres de Dominio
DSC	Llamada Selectiva Digital
EJB	Javabeans Empresarial
E/S	Entrada/Salida
FCTR	Umbral de Control de Flujo
FDDI	Interfaz de Datos Distribuida por Fibra
FIFO	Primero en Entrar, Primero en Salir

FPGA	Arreglo de Compuertas Programable por Campo
FTP	Protocolo de Transferencia de Archivos
GND	Tierra
GNU	Gnu No es Unix
GPIO	Entrada/Salida de Propósito General
GUI	Interfaz Gráfica de Usuarios
HDL	Lenguaje de Descripción de Hardware
HTTP	Protocolo de Transferencia de Hipertexto
ICMP	Protocolo de Mensajes de Control de Internet
ID	Identificación
IDE	Ambiente de Desarrollo Integrado
IEEE	Instituto de Ingeniería Eléctrica y Electrónica
IMR	Registro de Interrupción de Máscara
I /O	Entrada/Salida
IP	Protocolo de Internet

ISIS	Sistema Inteligente para Ingreso de Esquemáticos
ISR	Registro de Estatus de Interrupción
JDK	Herramientas de Desarrollo de Java
JTAG	Grupo de Acción Comprobación
LAN	Red de Área Local
LCD	Pantalla de Cristal Líquido
LED	Diodo Emisor de Luz
M4K	Bloque de Memoria de 4096 bits
MAC	Control de Acceso al Medio
MRCMD	Comando para Leer Datos de la Memoria Incrementando la Dirección de Registro
MRCMDX	Comando para Leer Datos de la Memoria Sin Incrementar la Dirección de Registro
MTU	Unidad Máxima de Transferencia
MWCMD	Comando para Escribir Datos en la Memoria sin Incrementar

	la Dirección de Registro
NCR	Registro de Control de Red
NSR	Registro de Estatus de Red
OSI	Interconexión de Sistemas Abiertos
PC	Computadora Personal
PCB	Placa de Circuito Impreso
PHP	Pre procesador de Hipertexto Php
PHY	Capa Física
PIO	Puerto Entrada/Salida
POSIX	Interfaz de Sistema Operativo Portátil Unix
RAM	Memoria de Acceso Aleatorio
RCR	Registro de Control de Recepción
RTFCR	Registro de Control de Flujo Tx/Rx
SDRAM	Memoria Síncrona de Acceso Dinámico Aleatorio
SoC	Sistema Sobre un Chip

SQL	Lenguaje de Consulta Estructurado
SRAM	Memoria Estática de Acceso Aleatorio
TCP	Protocolo de Control de Transmisión
TCR	Registro de Control de Transmisión
TDS	Flujo de Datos Tabular
TELNET	Red de Telecomunicación
TTL	Tiempo de Vida
TXREQ	Solicitud de Transmisión
TX / RX	Transmisión/Recepción
UART	Transmisor – Receptor Asíncrono Universal
UDP	Protocolo de Datagramas de Usuario
USB	Bus Serial Universal
UTP	Cable de Par Trenzado
VHDL	Lenguaje de Descripción de Hardware Vhsic
VHSIC	Circuitos Integrados de Muy Alta Velocidad

INTRODUCCIÓN

En la actualidad, ethernet es el estándar más popular para la transmisión y recepción de datos en una red de área local (LAN), se encuentra implementado en casi todos lados debido a la rápida expansión del internet. Como consecuencia, cuando se desarrollan nuevas aplicaciones con dispositivos electrónicos, surge la necesidad de hacer uso de esta tecnología existente para comunicarse.

Los sistemas embebidos configurables es otra tecnología que hoy en día, no solo se encuentra en una amplia gama de aplicaciones de ingeniería, si no también, está avanzando vertiginosamente en capacidad de procesamiento y en tamaño de empaquetamiento, logrando integrar más componentes, haciendo que se puedan manejar más recursos en una misma tarjeta base.

Por este motivo, usamos estas dos tecnologías para diseñar un sistema de autenticación de usuarios, utilizando como plataforma de desarrollo la tarjeta DE2 y empleando el protocolo de comunicación ethernet para enviar y recibir datos.

Con la intención de explicar paso a paso cómo se elaboró este proyecto, la tesina se lo ha dividido en 4 capítulos:

En el capítulo 1, detallamos los objetivos generales y específicos, además los alcances y limitaciones de nuestro proyecto.

En el capítulo 2, damos a conocer los conceptos teóricos básicos del hardware, software y protocolos usados, necesarios para implementar el proyecto.

En el capítulo 3, explicamos el diseño del proyecto con un esquemático y un diagrama de bloques, además de ir detallando los pasos necesarios para su configuración, implementación y su respectivo funcionamiento en un ambiente de laboratorio.

En el capítulo 4, realizaremos una comparación de los resultados obtenidos en las pruebas del prototipo con los objetivos propuestos, además de elaborar las conclusiones y recomendaciones pertinentes que se debe tomar en cuenta al momento de desarrollar este proyecto.

Finalmente en los anexos agregaremos información acerca del código utilizado para la programación de la tarjeta DE2 y la aplicación desarrollada en Java.

CAPÍTULO 1

1. GENERALIDADES

En este capítulo detallamos los precedentes que tomamos en cuenta para el diseño del proyecto, así como sus alcances y limitaciones, además de los resultados esperados y su respectiva aplicación.

1.1. ANTECEDENTES

Hoy en día, gracias a las nuevas tecnologías que tenemos a disposición, los sistemas de autenticación de usuarios están

incorporando modernos y sofisticados equipos electrónicos de autenticación, que se comunican entre sí, utilizando el protocolo IP, a través de una red ethernet, dando una mayor flexibilidad de implementación ya que no es necesario agregar nuevas conexiones al instalarlos y comunicar los equipos.

Sin embargo, el precio para incorporar estos equipos en las empresas que desean ejercer un control más exhaustivo del uso de sus instalaciones, se vuelve más costoso y el proceso de rediseño de esos sistemas, más complejo, al momento de agregar nuevas tecnologías en ellas, provocando que tengan que comprar nuevos productos para poder incorporar esas tendencias, desechando los equipos anteriores.

Por esa razón, decidimos usar la tecnología de los microprocesadores embebidos para implementar un sistema de autenticación que sea eficaz, de bajo costo, versátil, sencillo de implementar y fácil de reconfigurar.

La tarjeta de desarrollo DE2 que incorpora esta tecnología, nos permite programar directamente en el lenguaje C++ para elaborar la aplicación, logrando facilidad de desarrollo en el diseño.

Además, gracias a que esta tarjeta utiliza tecnología digital flexible y reprogramable de los FPGA (Arreglo de Puertas Programables por Campos) integrada en su procesador NIOS II, nos permite reprogramarla cuando se lo requiera, bajando los costos de rediseño.

También, la tarjeta agrega otros recursos de hardware permitiendo hacer uso de ellos dependiendo de las necesidades del desarrollador, agregando versatilidad al diseño.

Por último, la tarjeta DE2 contiene la tecnología suficiente para elaborar un sistema de autenticación eficaz y de bajo costo.

1.2. OBJETIVOS

Se realiza un planteamiento de conceptos, tanto en software como en hardware, que desarrolla a lo largo de la elaboración del proyecto.

1.2.1. OBJETIVOS GENERALES

- ✓ Construir un sistema embebido basado en el microprocesador NIOS II y bloques de lógicos configurable que permita establecer una comunicación Ethernet.

- ✓ Implementar el modelo de referencia TCP/IP en la tarjeta DE2 para enviar y recibir paquetes en una red.

- ✓ Entender la versatilidad que brinda en la actualidad los sistemas embebidos configurables para desarrollar distintas aplicaciones de ingeniería.

- ✓ Constatar las ventajas de utilizar la tarjeta de desarrollo DE2 con tecnología FPGA integrada en su procesador NIOS II en el diseño, depuración e implementación de sistemas digitales.

- ✓ Desarrollar destrezas en el diseño de soluciones de problemas de ingeniería que afectan a la sociedad.

- ✓ Probar en un ambiente de laboratorio prototipos de productos que se puedan implementar en la vida real.

1.2.2. OBJETIVOS ESPECÍFICOS

- ✓ Implementar un sistema de autenticación de usuarios seguro, versátil y escalable dentro de un edificio.

- ✓ Emplear la tarjeta de desarrollo DE2 como módulo de control en el diseño de un sistema digital.

- ✓ Adquirir conocimientos necesarios para poder utilizar, agregar y configurar los recursos de la tarjeta DE2 en el software Qsys.

- ✓ Utilizar VHDL para realizar la descripción de señales en la tarjeta DE2 utilizando Quartus.

- ✓ Aplicar conocimientos de programación en C++ para realizar la aplicación de la tarjeta DE2 utilizando como entorno de desarrollo Eclipse

- ✓ Usar el Stack de funciones en Java para realizar el envío y recepción de paquetes IP desde la computadora.

- ✓ Elaborar una interfaz gráfica en lenguaje de Java para la administración de un sistema de autenticación de usuarios desde una computadora

- ✓ Gestionar mediante MySQL una base de datos para guardar información básica de los usuarios y el historial de ingreso de las oficinas de un edificio.

- ✓ Emplear el LCD de la tarjeta de desarrollo DE2 como interfaz visual de usuario.

- ✓ Usar los puertos GPIO de la tarjeta DE2 para recibir datos desde un teclado matricial.

- ✓ Habilitar el puerto ethernet de la tarjeta DE2 para transmitir y recibir datos en una red LAN.

- ✓ Implementar una pequeña red LAN utilizando un switch D-Link para conectar equipos.

- ✓ Comprender el funcionamiento del protocolo ethernet.

- ✓ Disponer del Stack de funciones del chip DM9000 integrado en la tarjeta DE2 para realizar el envío y recepción de paquetes IP.

- ✓ Entender los conceptos básicos de cableado para conectar equipos en una red LAN.

- ✓ Usar el protocolo ARP para enlazar equipos electrónicos mediante la resolución de direcciones lógicas en direcciones físicas.

- ✓ Estudiar conceptos básicos para la asignación de direcciones IP en equipos conectados a una red LAN.

- ✓ Entender cómo se realiza el proceso de encapsulamiento para transmitir datos usando el protocolo IP.

- ✓ Utilizar el protocolo UDP para transportar datos.

- ✓ Agregar seguridad a los datos enviados, usando un algoritmo de encriptación simétrico.

- ✓ Comprobar la seguridad de los datos enviados utilizando el software Wireshark para realizar el sniffing de paquetes.

- ✓ Gestionar un base de datos mediante MySQL.

1.3. ALCANCE Y LIMITACIONES

A continuación, detallamos las limitantes y los alcances del proyecto.

1.3.1. ALCANCE

Este proyecto permite:

- ❖ Autenticación de usuarios utilizando una computadora servidor.

- ❖ Desbloquear puertas de un edificio usando la tarjeta DE2 como dispositivo actuador.

- ❖ Monitorear la seguridad de las oficinas de un edificio desde una computadora.

- ❖ Guardar un registro del historial de ingreso de los usuarios en las instalaciones de un edificio.

- Gestionar con una base de datos la información de los usuarios para realizar el proceso de autenticación.

- Comunicar al sistema de autenticación utilizando una red ethernet existente.

- Agregar más tarjetas DE2 al sistema dándole escalabilidad al proyecto.

- Posibilidad de implementación en un ambiente real.

1.3.2. LIMITACIONES

Las limitaciones del proyecto son:

- Los elementos actuadores para bloquear y desbloquear las puertas del sistema no son reales. Estos son simulados mediante Leds y botoneras de la tarjeta DE2.

- El sistema está diseñado explícitamente para ser usado en una red de área local.

- ❖ Las pruebas son realizadas en un ambiente de laboratorio, en una red LAN donde no existe un tráfico de paquetes real.

- ❖ La funcionalidad del teclado matricial 4x4 usado, no permite el ingreso de datos con letras del alfabeto.

1.4. RESULTADOS ESPERADOS

Las expectativas que se espera son:

- ➢ Obtener un sistema de autenticación de usuarios seguro, escalable y versátil.

- ➢ Elaborar un producto con una Interfaz de fácil uso y que sea amigable para el usuario.

- ➢ Controlar y monitorizar en tiempo real el acceso a las oficinas de un edificio.

- Manejar una base de datos, que sea auditable y permita ver con facilidad el historial de ingreso de usuarios a las instalaciones de un edificio.

- Implementar el sistema en ambiente real luego de ser probado en un ambiente de laboratorio.

1.5. APLICACIONES

La aplicación de este proyecto está orientada a atender las necesidades de seguridad de empresas, para permitirles ejercer un mejor control del uso de sus instalaciones y llevar un registro de ingresos del personal en una base de datos.

CAPÍTULO 2

2. MARCO TEÓRICO

Para una fácil lectura de la información que se necesita repasar y una mejor comprensión de los conceptos básicos que se debe entender antes de la implementación del proyecto, el capítulo se dividió en 4 secciones:

- Tecnologías de apoyo del sistema.
- Especificaciones del hardware implementado.
- Protocolos y conceptos de comunicación usados.
- Herramientas de diseño manejados.

2.1. TECNOLOGÍAS DE APOYO DEL SISTEMA.

En esta sección se detallan los conceptos básicos, características y beneficios de las tecnologías utilizada en nuestro proyecto.

2.1.1. SISTEMAS EMBEBIDOS

Los sistemas embebidos son sistemas de computación diseñados para realizar funciones dedicadas y cubrir necesidades específicas, integrando sus componentes en una misma tarjeta base. Figura 2.1.

Figura 2.1 – Sistema embebido [1]

Estos sistemas se pueden programar directamente en lenguaje ensamblador o usando compiladores específicos, que emplean lenguajes C o C++.

Existen muchos ejemplos en donde los sistemas embebidos se encuentran inmersos, la gran mayoría se localizan en dispositivos que realizan tareas dedicadas como: un sistema de control de acceso, la electrónica que controla una máquina expendedora o el sistema de control de una fotocopiadora, entre un sin número de aplicaciones más.

Las características que diferencia a los sistemas embebidos de otros sistemas de cómputo son:

- Ejecuta un mismo programa de forma repetitiva.
- Realiza cálculos en tiempo real de forma precisa y acelerada.
- Son más baratos y poseen un tamaño reducido.
- Consumen menos energía.

Las componentes básicas son:

- Microprocesador.
- Memoria.
- Caché.
- Disco duro.
- BIOS-ROM.
- CMOS-RAM.
- Chip Set.
- Entradas y salidas del sistema.
- Ranuras de expansión para tareas específicas.

2.1.2. FPGA

La tecnología FPGA por sus siglas en ingles (Field Programable Gate Array) - Arreglo de Puertas Programables por Campos es una técnica de diseño de dispositivos semiconductores que contienen bloques de lógica digital cuya interconexión y funcionalidad permite construir circuitos digitales integrados Figura 2.2, que pueden ser reconfigurada a nivel de hardware aun después de su fabricación.

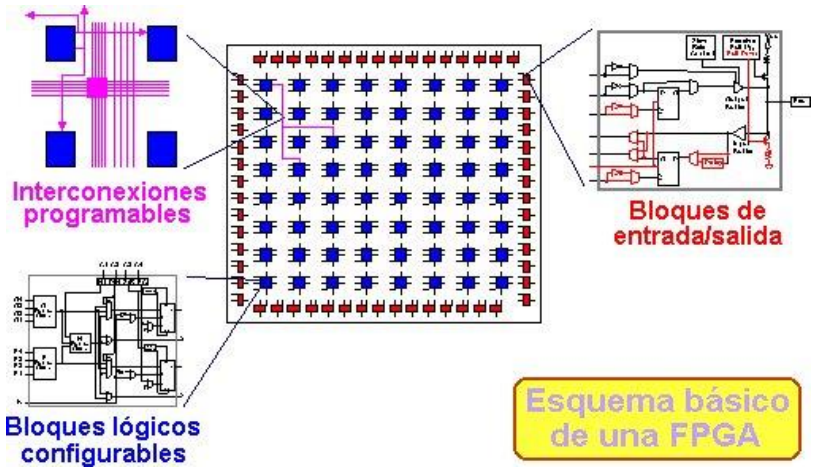


Figura 2.2 – Estructura de una FPGA [2]

La configuración de las FPGAs se realiza usando lenguaje de descripción de hardware (HDL) generalmente, VHDL y Verilog utilizando el software del fabricante para describirla y programarla.

Los beneficios de utilizar la tecnología FPGA son:

- Alto rendimiento.
- Poco consumo de energía.
- Excelente confiabilidad.
- Bajo precio.

- Mantenimiento a largo plazo.
- Existe una gama amplia de productos en el mercado.
- Mejoras continuas en su tecnología.
- Generación rápida de prototipos de sistemas embebidos.
- Implementación personalizada.
- Tiempos de respuesta de E/S rápidos.

Los fabricantes de los FPGAs para desarrollar sistemas embebidos son:

- Xilinx.
- Altera.
- Atmel.
- Cypress.
- Lattice Semiconductor.
- Quicklogic.
- Achronix Semiconducto.

2.1.2.1. CYCLONE II

La familia Cyclone II de la Figura 2.3 es un FPGA de bajo costo anunciado por Altera el 28 de junio de 2004.

Figura 2.3 – FPGA Cyclone II [3]

La familia Cyclone II tiene las siguientes características:

- Arquitectura con 4,608 a 68,416 elementos lógicos.
- Bloques embebidos de Memoria M4K.
- Desde 0.1 a 1.1 Mbits de memoria RAM útil.
- 4,096 bits de memoria por bloque.

Multiplicadores Embebidos

- Desde 13 a 150 multiplicadores de 18x18 bits. Cada uno puede ser configurado como dos independiente de 9x9 bits.
- Registros de entrada y salida opcionales.

Circuito de administración flexible de Reloj

- Red de jerarquía del reloj con rendimiento de hasta 402.5 MHz.

Configuración de Dispositivo

- Configuración serial con tiempos de configuración inferiores a 100ms.
- Se pueden realizar diferentes modos de configuración: Serial Activo, Serial Pasivo y JTAG.
- La configuración del dispositivo tolera múltiples voltajes (3.3, 2.5 o 1.8 V).

En la figura 2.4 se detallan las características de todas las series:

Table 1–1. Cyclone II FPGA Family Features

Feature	EP2C5	EP2C8 *(2)*	EP2C15 *(1)*	EP2C20 *(2)*	EP2C35	EP2C50	EP2C70
LEs	4,608	8,256	14,448	18,752	33,216	50,528	68,416
M4K RAM blocks (4 Kbits plus 512 parity bits	26	36	52	52	105	129	250
Total RAM bits	119,808	165,888	239,616	239,616	483,840	594,432	1,152,000
Embedded multipliers *(3)*	13	18	26	26	35	86	150
PLLs	2	2	4	4	4	4	4
Maximum user I/O pins	158	182	315	315	475	450	622

Figura 2.4 – FPGA Cyclone II [4]

2.1.3. SISTEMA SOBRE UN CHIP

SoC por sus siglas en inglés (System on Chip) es una plataforma que integra el procesador, la memoria, la red de conexión en un solo chip, lo que provoca una menor demanda de potencia del sistema.

Un SoC típico posee un núcleo procesador de 32 bits y funciones seleccionables por el desarrollador como drivers E/S, decoders, memorias, interfaces de bus y soportes de red, que permiten diseñar sistemas a la medida, dando como resultado, un sistema

embebido que puede realizar funciones más complejas desde un mismo chip sin necesidad de agregar otros chip a la tarjeta madre, reduciendo espacios y consumos de potencia a lo estrictamente necesario. Figura 2.5.

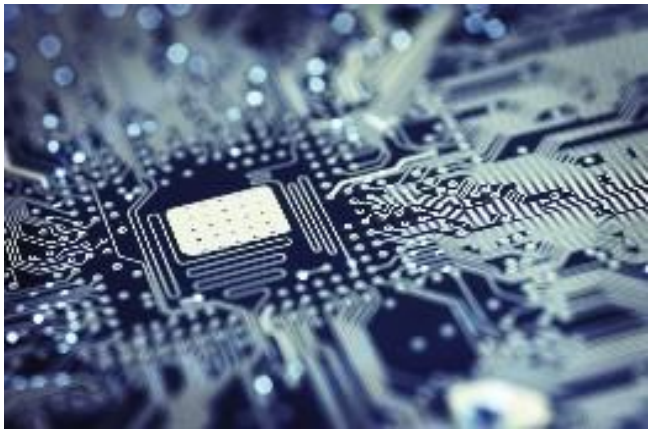


Figura 2.5 – Sistema en un chip [5]

2.1.3.1. PROCESADOR NIOS II

Es un sistema SoC que agrega la tecnología FPGA en el mismo chip, permitiéndole al sistema ser reconfigurable.

Esta interesante propuesta, se está viendo actualmente en el mercado de los sistemas embebidos. Conforme las FPGA han ido mejorando su capacidad, velocidad, coste, han

posibilitado la implantación de soft processor en el chip, permitiéndole añadir solo aquello que es necesario en el diseño. Figura 2.6.

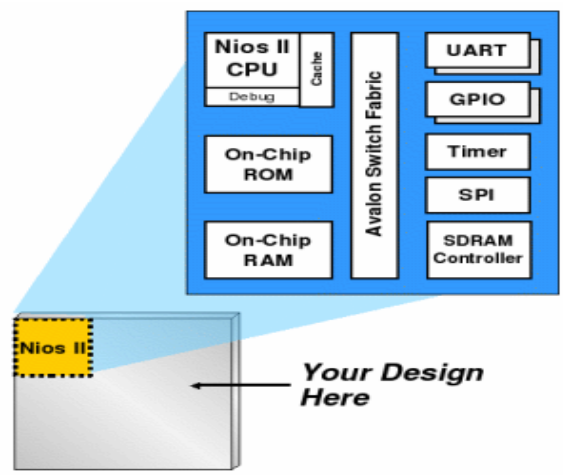


Figura 2.6 – Procesador embebido NIOS II [6]

En el mercado existen tres versiones del procesador Nios II:

- NIOS II /f (fast): Cuenta con pipeline de 6 etapas, aumenta su desempeño con opciones específicas como: memoria caché de instrucciones, datos y una unidad de manejo de memoria.

- NIOS II /s (standard): Contiene un pipeline de 5 etapas que combina rendimiento y consumo de recursos, mediante una unidad aritmética lógica.

- NIOS II /e (economic): Se remueve el pipeline, requiere menos recursos de la FPGA y carece de operaciones de multiplicación y división.

Un sistema NIOS II se compone de las siguientes partes como se muestra en la figura 2.7:

- Procesador NIOS II de 32 bits.
- AVALON (Bus del sistema).
- Memoria para programas y datos.
- Periféricos integrados.
- Interfaces de entrada y salida

Figura 2.7 – Sistema NIOS II [7]

2.2. ESPECIFICACIONES DEL HARDWARE.

En esta sección se detallan las características básicas de cada una de las componentes que se utilizaron para implementar el proyecto.

2.2.1. TARJETA DE DESARROLLO DE2

La tarjeta de desarrollo DE2 de la figura 2.8 contiene un FPGA Cyclone II EP2C35 de Altera, diseñada con propósito académico, para mostrar los conceptos fundamentales de diseño sobre FPGA,

lógica digital y organización de computadores en laboratorios de las universidades de ingeniera.

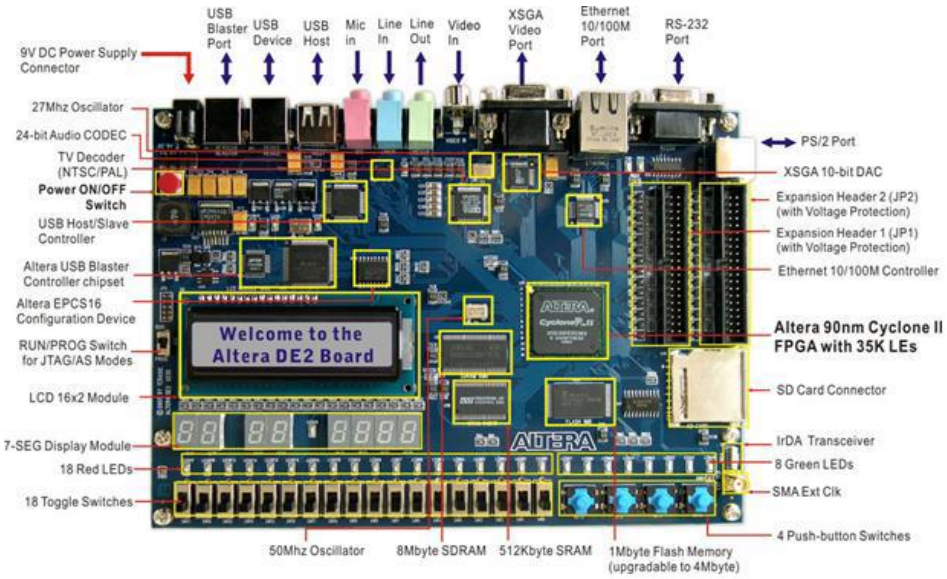


Figura 2.8 – Tarjeta de desarrollo DE2 [8]

El kit de desarrollo de la tarjeta DE2 contiene lo siguiente:

- Una tarjeta DE2 de 8'' x 6'' con un FPGA Cyclone II EP2C35.
- Adaptador de 9V AC/DC.
- Cable USB para programar y controlar la tarjeta DE2.

- Cobertor para la tarjeta.
- Guía de Instalación.
- CD con documentación detallada de la DE2, material de apoyo e instaladores de los programas Quartus II y NIOS II.

En la figura 2.9 se detalla las componentes de la Tarjeta DE2.

Información de la Tarjeta DE2	
FPGA	• Cyclone II EP2C35F672C6 con EPCS16 16-Mbit • configuración serial del dispositivo
Interfaces E/S	• USB-Blaster para la configuración FPGA • Linea In/Out, Micrófono (24-bit Audio CODEC) • Video Out (VGA 10-bit DAC) • Video In (NTSC/PAL/Multi-format) • RS232 • Puerto Infrarrojo • PS/2 puerto para el mouse o el teclado • 10/100 Ethernet • USB 2.0 (tipo A y tipo B) • Cabecera de Expansión (Dos de 40 pines)
Memoria	• 8 MB SDRAM, 512 KB SRAM, 4 MB Flash • Puerto para la tarjeta SD
Displays	• Ocho displays de 7-segmentos • Display LCD de 16 x 2
Switches y LEDs	• 18 switches • 18 LEDs rojos • 9 LEDs verdes • 4 botoneras
Relojes	• 50 MHz clock • 27 MHz clock • Entrada SMA de clock

Figura 2.9 – Componentes de la tarjeta DE2 [9]

A continuación se detallan los módulos de la tarjeta DE2 que se utilizaron:

2.2.1.1. MÓDULO LCD

El módulo LCD de la figura 2.10 es una pantalla de cristal líquido agregada en la tarjeta DE2, que se puede utilizar como dispositivo de visualización gráfica, para mostrar caracteres y símbolos, mediante el envío adecuado de comandos al controlador HD44780 de la pantalla.

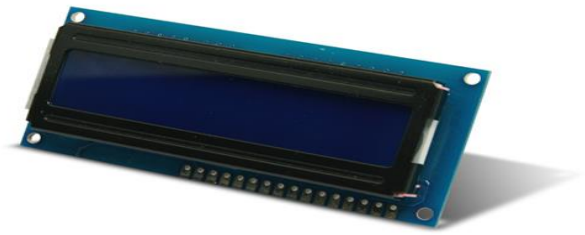


Figura 2.10 – Módulo LCD 16X2 [10]

Las características de la pantalla LCD son:

- Consta de 2 filas, en donde se pueden escribir hasta 16 caracteres por fila, con una resolución de 5x7 pixeles por símbolo.

- Se puede escribir cualquier carácter del código ASCII.
- Permite el desplazamiento del texto que se imprime, de izquierda a derecha y viceversa.
- Posibilita el movimiento del cursor.

2.2.1.2. MÓDULO ETHERNET

El módulo ethernet de la tarjeta DE2 proporciona soporte a nivel de hardware, para efectuar una comunicación Ethernet, a través del chip Davicom DM9000A.

2.2.1.2.1. CHIP DAVICOM DM9000A

El chip Davicom DM9000A de la figura 2.11 es un controlador Fast Ethernet completamente integrado, de bajo consumo de energía y de gran eficiencia, que incluye: una interfaz de procesador general, 16 Kbytes de SRAM, una unidad de control de acceso a medios (MAC), y un transceptor PHY 10/100M.

Figura 2.11 – Chip DAVICOM DM9000A [11]

El DM9000A es totalmente compatible con el estándar Ethernet IEEE 802.3.

Los drivers que controlan el DM9000A se pueden programar en lenguaje C, utilizando el Stack de funciones ya hechos de Microchip.

El DM9000A tiene una interfaz de apoyo de 8 bits y de 16 bits para acceso a la memoria interna. La PHY del DM9000A puede interactuar con UTP categoría 3, 4, 5 en 10Base-Tx y UTP categoría 5 en 100Base-Tx con AUTO-MDIX. Además, tiene un control de flujo full-dúplex y su función de auto-negociación se configura automáticamente.

La figura 2.12 muestra el diagrama de bloques del chip DM9000A

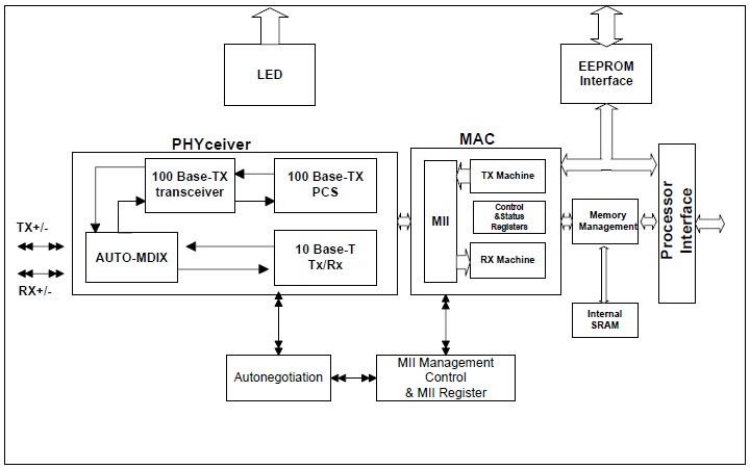


Figura 2.12 – Diagrama de bloques del DM9000A [12]

2.2.1.3. CONECTORES GPIO

GPIO por sus siglas en inglés (General Purpose Input/Output) – conectores de entrada/salida de propósito general, es un pin genérico de un chip que se puede configurar por el usuario mediante programación, como entrada o salida para interactuar con el medio externo cuando se ejecuta un programa en el chip. Figura 2.13.

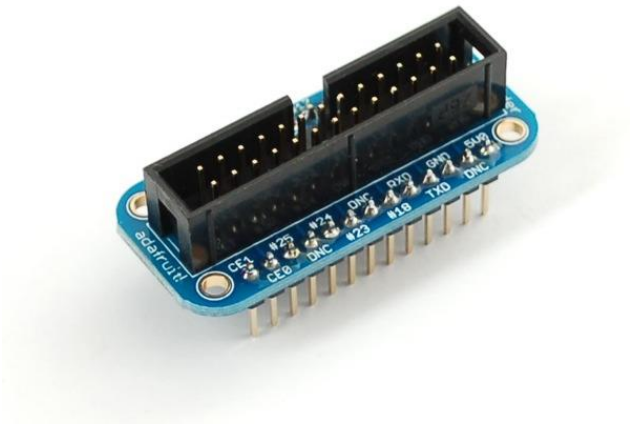


Figura 2.13 – Conectores GPIO [13]

La tarjeta DE2 proporciona dos cabeceras de expansión de 40 pines. Cada una de las cabecera se conecta directamente a 36 pines en del FPGA Cyclone II. Los 4 pines restantes sirven para proporcionar niveles DC de 5 V en un pin, 3 V en otro y los dos restantes para GND de las fuentes respectivamente. Cada pin en las cabeceras de expansión, están conectadas a dos diodos y una resistencia que ofrece protección contra altas y bajas tensiones.

2.2.2. SWITCH D-LINK

Este conmutador Plug and Play de la figura 2.14 se ha diseñado para implementarse en pequeñas empresas y oficinas, garantizando una rápida conexión. Permite que los usuarios conecten un puerto de cualquier tipo a un nodo a 10Mbps o 100Mbps para multiplicar el ancho de banda, mejorar los tiempos de respuesta y realizar pesadas cargas de trabajo.

Figura 2.14 – Switch D-LINK [14]

Proporciona 5 puertos, que pueden negociar tanto la velocidad de conexión en entornos de red 10BASE-T y 100BASE-TX como el modo de transmisión full-dúplex o half-dúplex.

El control de flujo que posee, reduce al mínimo la pérdida de paquetes, enviando señales de colisión cuando el buffer de

recepción del puerto está lleno. El control de flujo está disponible solo en modo full-dúplex.

2.2.3. CABLE UTP

El cable UTP por sus siglas en inglés (Unshielded Twisted Pair) - par trenzado sin blindaje mostrado en la figura 2.15, consiste en 4 pares de conductores de cobre, aislados y entrelazados entre sí, para anular interferencias de fuentes externas y diafonía de los conductores opuestos, envueltos en mismo cable pero sin agregar un blindaje exterior.

Figura 2.15 – Cable UTP [15]

Este tipo de cable es usado en comunicaciones para transmitir señales eléctricas, generalmente se los utiliza en redes de área local (LAN), son de bajo costo y fácil implementación pero produce más errores en las señales que se transmiten debido al ruido, comparado con otros tipos de cables que tienen una protección contra interferencias externas. Su impedancia es de 100 Ohmios. Además, tiene limitaciones para trabajar a distancias mayores a los 100 metros, sin la utilización de un repetidor o regenerador de señal.

2.2.4. TECLADO MATRICIAL

Un teclado matricial 4x4 como se observa en la figura 2.16, es un pequeño circuito ensamblado en forma de matriz configurada en 4 columnas y 4 renglones, cuando no se oprime ninguna tecla, no existe ninguna conexión entre las columnas y renglones, cuando se oprime una tecla se hace una conexión entre la columna y el renglón de la tecla.

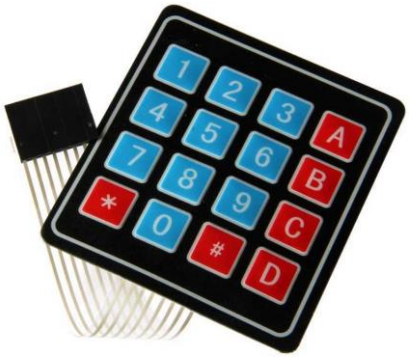


Figura 2.16 – Teclado matricial 4x4 [16]

El teclado matricial 4x4 posee:

- 16 teclas
- Bus de datos de 8 Pines.

2.3. PROTOCOLOS Y CONCEPTOS DE COMUNICACIÓN USADOS.

En esta sección se detallan los conceptos de comunicaciones necesarios que se deben conocer antes de elaborar el proyecto.

2.3.1. MODELO DE CAPAS TCP/IP

TCP/IP (Protocolo de Control de Transmisión/Protocolo de Internet) es una arquitectura de capas que determina los protocolos que se debe implementar para efectuar una comunicación entre sistemas. Actualmente es la arquitectura estándar más adoptada para la interconexión de equipos.

El modelo TCP/IP como se muestra en la figura 2.17 es la versión resumida del modelo OSI y se divide en cuatro capas, cada una con su conjunto de protocolos.

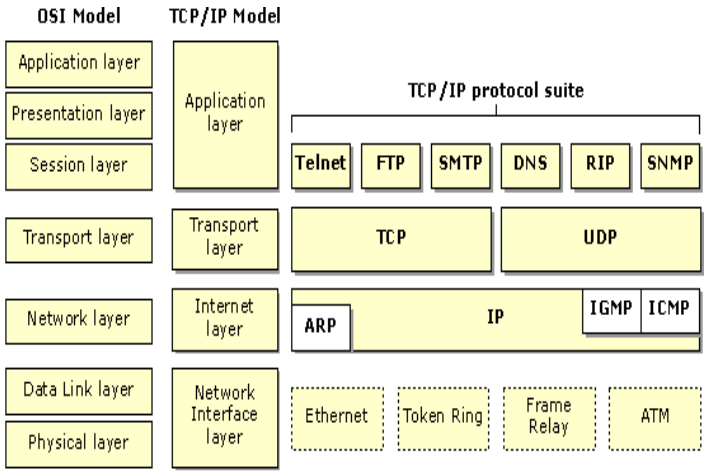


Figura 2.17 – Modelo TCP/IP [17]

Un sistema implementado con el modelo TCP/IP se basa en manejar los paquetes de datos que se quieren enviar a través de la red por etapas, agregándoles información necesaria al paquete (encabezado) dependiendo del protocolo usado para que pueda continuar a la siguiente etapa (encapsulación), hasta completar las 4 capas, para que el dato llegue confiablemente a su destino.

En la figura 2.18 se muestra el proceso de encapsulación de paquetes.

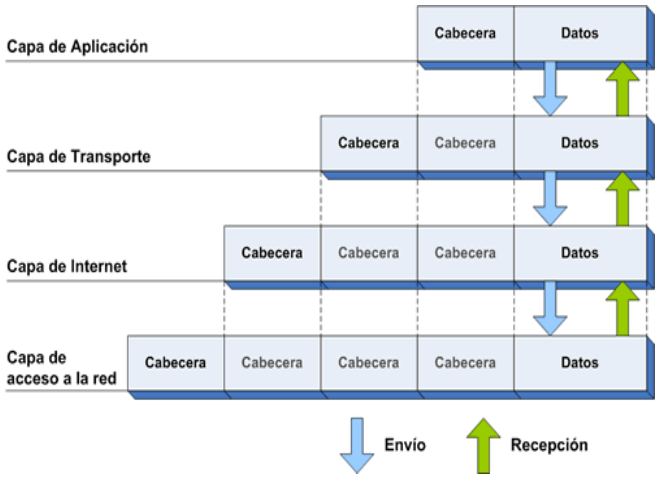


Figura 2.18 – Proceso de encapsulación [18]

Un aspecto importante al manejar un modelo de capas es que se puede solucionar problemas de comunicación de red revisando en una o varias capas específicas el inconveniente, de acuerdo con el nivel de abstracción que se tenga del problema.

A continuación se detalla cada una de las capas del modelo TCP/IP:

Capa de Acceso a la red:

Esta etapa muestra las especificaciones y los recursos que se deben implementar para transmitir datos a través de una red física.

Los protocolos que maneja esta capa encontramos Ethernet, FDDI, Token Ring, ATM, entre otros.

Capa de Internet:

Esta capa tiene como propósito hacer llegar los paquetes de datos desde cualquier equipo de la red a su destino.

En esta etapa es donde se determina la mejor ruta y la conmutación de paquetes que se deben realizar a través de la red, a este proceso se lo conoce como enrutamiento y utiliza direcciones IP asignadas a cada equipo para poder identificar el dispositivo destino correcto.

Los protocolos de esta capa son IP (Protocolo de Internet), ARP, ICMP, entre otros.

Capa de Transporte:

Los protocolos de la capa transporte se encargan de solucionar problemas de confiabilidad de un enlace dado a través del control de flujo, segmentación y control de errores, ya que es posible surjan problemas de congestión de red, los paquetes de datos

lleguen en diferente orden a la aplicación destino o se pierdan paquetes a través de la red.

Estos protocolos de transporte se utilizan además para especificar a qué aplicación del equipo destino van dirigido los datos usando identificadores llamados puertos que se colocan en el encabezado de cada paquete de datos.

Los protocolos se utilizan en esta capa son UDP y TCP.

Capa de Aplicación:

Esta capa se comunica con los protocolos de la capa inferior (capa de transporte) para mostrar los datos con la lógica necesaria en la aplicación destino.

Existen diferentes tipos de aplicaciones en esta capa, pero la mayoría son servicios de red o aplicaciones brindadas al usuario para proporcionarle una interfaz con el sistema operativo.

Entre los protocolos más usados en esta capa encontramos HTTP, FTP, DNS, TELNET, POP3, entre otros.

2.3.2. PROTOCOLO ETHERNET

Ethernet al que también se conoce como IEEE 802.3 es el estándar más popular para comunicar redes LAN, usando el método de transmisión de datos llamado Acceso múltiple con detección de portadora y detección de colisiones (CSMA/CD).

El protocolo Ethernet define las características del cableado, señalización de nivel físico y los formatos de tramas de datos del nivel de acceso a la red que se implementan en el modelo TCP/IP.

Las redes Ethernet tienen un esquema de direccionamiento de 48 bits, asignándole a cada computadora un número único conocido como dirección Ethernet o Mac address representado en hexadecimal.

Las velocidades manejadas en el estándar Ethernet son de 10Mbps (Ethernet estándar), 100Mbps (Fast Ethernet – 100BASEX) y de 1000Mbps utilizando el Gigabit Ethernet cuya especificación se encuentra respaldada por la IEEE 802.3z.

En la figura 2.19 se muestra el esquema de una trama Ethernet.

Figura 2.19 – Trama Ethernet [19]

A continuación se detalla cada campo de la trama Ethernet:

- Preámbulo: Indica el inicio de la trama y tiene como finalidad de que el dispositivo que lo reciba sepa cuando inicia una nueva trama.

- Inicio: Indica que la trama inicia a partir de ese byte.

- Dirección Mac Origen Y Destino: Indican las direcciones físicas de los dispositivos al que van dirigidos los datos y del que se originaron los datos.

- Tipo: Indica con que protocolo están encapsulados los datos en el campo datos.

- Datos: Es donde se encuentran los datos, dentro de estos pueden ir cabeceras de otros protocolos de capas superiores.

- CRC: La secuencia de comprobación es un campo de 4 bytes que contiene un valor de verificación CRC (Control de Redundancia cíclica). El emisor calcula el CRC de toda la trama, y cuando llega al receptor este lo recalcula y debe ser igual para que se considere una trama válida.

2.3.2.1. CSMA/CD

CSMA/CD es el mecanismo de control para el acceso al medio físico utilizado por las redes Ethernet que determina cómo y cuándo un paquete de dato es ubicado en el cable.

Antes de que un dispositivo Ethernet empiece a transmitir datos, primero tiene que escuchar para asegurarse de que el medio está "libre", es decir, ningún dispositivo este transmitiendo.

Cuando la red está libre, los dispositivos pueden iniciar la transmisión. Durante el proceso de transmisión, el dispositivo continuará escuchando la red para ver si otro dispositivo también está trasmitiendo. Si no hay ninguno, entonces la data es considerada enviada.

Sin embargo, si durante la transmisión detecta que otro dispositivo también está transmitiendo (conocida como una colisión), ambos detendrán sus transmisiones y realizarán un proceso conocido como back-off en el que esperaran un tiempo aleatorio antes de intentar volver a transmitir nuevamente.

Una de las ventajas de este protocolo es que realiza esta detección de colisiones relativamente fácil en redes LAN, además el tiempo medio necesario para detectar una colisión es relativamente bajo.

2.3.3. PROTOCOLO ARP

El protocolo ARP (protocolo de resolución de direcciones) es el responsable de mapear las direcciones de protocolo de alto nivel (Direcciones IP) con direcciones de red físicas (Mac address).

Si una aplicación desea enviar datos a una determinada dirección IP destino, el mecanismo de enrutamiento IP determina cual es la dirección IP del siguiente salto para encaminar al paquete a su destino final. El módulo ARP busca en su cache e intenta asociar la dirección IP con una dirección física ligada al dispositivo del siguiente salto.

Este proceso se vuelve a realizar en el nuevo dispositivo hasta que el paquete llegue hasta su destino final.

En caso de que el módulo ARP no encuentre una asociación, se descarta el paquete.

Este protocolo siempre mantiene actualizada el cache del módulo ARP generando solicitudes para obtener las direcciones físicas asociadas a direcciones IP solicitadas con anterioridad.

En la figura 2.20 se muestra una imagen del paquete de solicitud y respuesta ARP.

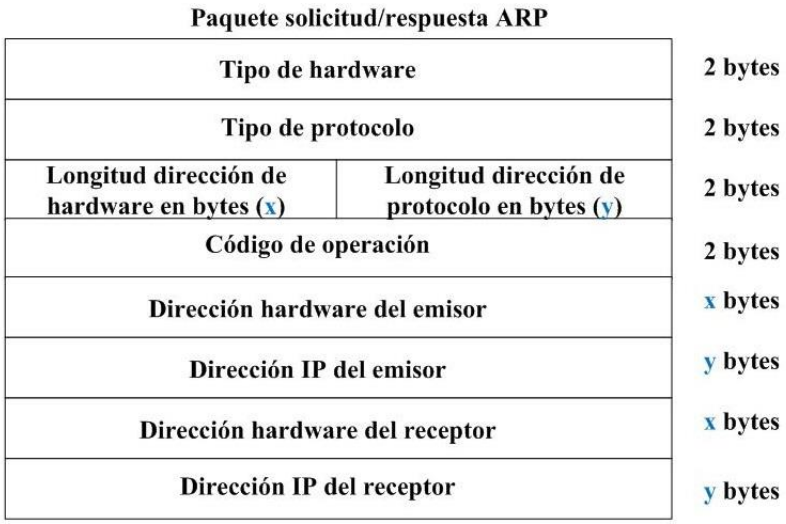


Figura 2.20 – Paquete de solicitud/respuesta ARP [20]

A continuación se detalla cada campo:

- Tipo de Hardware: Especifica el tipo de hardware, estos podrían ser Ethernet o Packet Radio Net.

- Tipo de Protocolo: Especifica el tipo de protocolo, es el mismo que en el campo de tipo de EtherType en la cabecera de IEEE 802.

- Longitud de las direcciones de hardware: Especifica la longitud (en byte) de las direcciones de hardware del paquete. Para IEEE 802.3 será de 6.
- Longitud de las direcciones de protocolo: Especifica la longitud (en bytes) de las direcciones de protocolo del paquete. Para el protocolo IP será de 4.

- Código de operación: Especifica si se trata de una petición (1) o una solicitud (2) ARP.

- Dirección de hardware del receptor y emisor: Contiene las direcciones físicas. En IEEE 802.3 son de 48bits.

- Dirección IP del receptor y emisor: Contiene las direcciones del protocolo. En TCP/IP son direcciones IP de 32bits.

2.3.4. PROTOCOLO IP

El protocolo IP (Protocolo de Internet) es uno de los protocolos más importantes de Internet ya que permite el transporte de paquetes de datos. Este Protocolo provee un servicio de

datagramas no fiable (también llamado del mejor esfuerzo) que hará lo mejor posible para que el paquete llegue a su destino.

El protocolo IP no provee ningún mecanismo de confianza para determinar si un paquete alcanza o no su destino, únicamente proporciona seguridad a sus cabeceras usando checksums para verificar que su contenido sea correcto.

Al no proporcionar seguridad para los datos transmitidos estos podrían llegar dañados, duplicados, desordenados o simplemente no llegar. Estos problemas de fiabilidad se solucionan con protocolos de capa superiores.

Una de las características del protocolo IP es que cuando la información a transmitir supera el MTU (Unidad Máxima de Transferencia), en el tramo de red donde va a circular, éste puede ser dividido en partes más pequeñas (fragmentación), y luego ser reensamblado en el destino. Estas pequeñas partes del paquete pueden viajar por diferentes caminos, debido a que IP es un protocolo no orientado a conexión.

En resumen las características relevantes del protocolo IP son:

- Protocolo no orientado a conexión.

- Fragmenta paquetes si es necesario.
- Direccionamiento mediante direcciones lógicas IP de 32 bits.
- Si un paquete no es recibido, este permanecerá en la red durante un tiempo finito, valiéndose del campo TTL en la cabecera IP.
- Realiza el “mejor esfuerzo” para la distribución de paquetes.
- Tamaño máximo del paquete de 65365 bytes.
- Solo se realiza verificación por suma al encabezado del paquete, no a los datos que este contiene.

En la figura 2.21 se muestra el formato de un paquete IP.

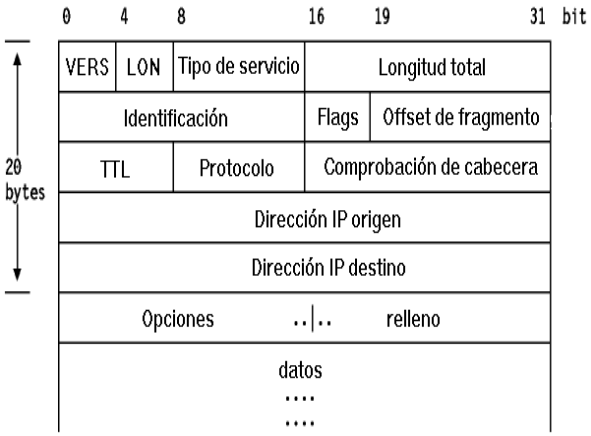


Figura 2.21 – Formato del paquete IP [21]

A continuación de detalla cada campo del paquete IP:

- VERS: Es la versión del protocolo IP.

- LON: Es la longitud de la cabecera IP, de 32 en 32 bits. No incluye el campo de datos.

- Tipo de Servicio: Es un indicador de la calidad de servicio solicitado por este datagrama IP. Donde los 3 primeros bits se llaman precedencia, los siguientes 4 TDS y el último

DSC. Los bits de precedencia son una medida de la naturaleza y prioridad de este datagrama. Los bits de TDS especifican el valor de tipo de servicio. El último bit reservado para uso futuro “Debe ser Cero”.

- Longitud Total: Longitud total del datagrama, cabecera y datos.

- Identificación: Un número único que asigna el emisor para ayudar a reensamblar un datagrama fragmentado. Los fragmentos de un datagrama tendrán el mismo número de identificación.

- Flags: Hay varios flags de control. El primer bit siempre debe ser 0. El segundo bit indica si se permite o no la fragmentación. El tercer bit indica si este es el último fragmento de un paquete.

- Offset de Fragmento: Se usa para ayudar al reensamblado de paquetes. Nos indica la ubicación de este paquete en uno fragmentado.

- TTL: Es un contador que sirve para limitar la vida de un paquete.
- Protocolo: Indica el protocolo de alto nivel al que IP debería transportar los datos.
- Comprobación de cabecera: Comprueba si la cabecera fue modificada, en ese caso se considera un paquete dañado.
- Dirección IP origen y destino: Dirección lógica de 32 bits.
- Opciones: Información usada por administración: Longitud variable.
- Relleno: Ajusta las opciones a 32 bits.

2.3.5. PROTOCOLO UDP

El protocolo UDP (Protocolo de Datagrama de Usuario) es un protocolo que ofrece a las aplicaciones un mecanismo para enviar datagramas IP encapsulados sin tener que establecer una conexión.

Es denominado también un protocolo del “máximo esfuerzo”, porque hace lo que puede para que el datagrama llegue a la aplicación destino, pero no garantiza que la aplicación lo reciba.

UDP no admite numeración de los datagramas y tampoco utiliza señales de confirmación de entrega, provocando que los datagramas lleguen desordenados, duplicados o no lleguen a su destino, haciendo que la garantía de que un paquete llegue a su destino no sea confiable en comparación con TCP.

Tampoco utiliza mecanismos de detección de errores, cuando se detecta un datagrama dañado, éste es descartado y no es entregado a la aplicación.

En la figura 2.22 se muestra el formato de un datagrama UDP.

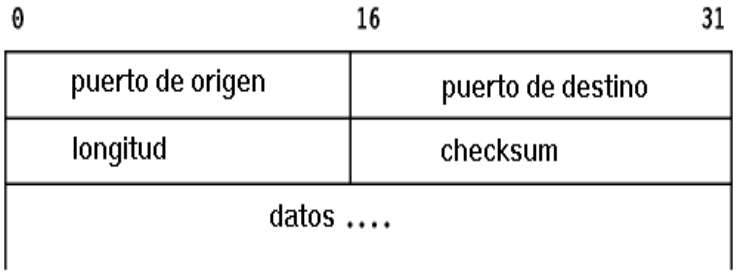


Figura 2.22 – Formato del datagrama UDP [22]

A continuación se especifica cada campo del datagrama:

- Puerto Origen: Es el número de puerto relacionado con la aplicación del remitente del segmento UDP. Este campo es opcional, se especifica en el caso que se necesita una respuesta.

- Puerto Destino: Este campo contiene el puerto correspondiente a la aplicación del equipo receptor al que se envía.
- Longitud: Este campo especifica la longitud total del datagrama, con el encabezado incluido. La longitud debería ser mayor a 8 bytes que es la longitud del encabezado.

- Checksum: Es una suma de comprobación realizada de tal manera que permita verificar la integridad del datagrama.

2.3.6. SOCKETS

Socket se designa a un concepto abstracto por el cual programas situados en computadoras distintas pueden intercambiar información de manera fiable y ordenada.

La programación de la red gira en torno al concepto de sockets. Un socket es un punto final de un enlace de comunicación de dos vías entre dos programas que se ejecutan en la red.

Los sockets permiten implementar una arquitectura cliente-servidor. La comunicación debe ser iniciada por uno de los programas que se denomina programa “cliente”. El segundo programa denominada “servidor” es la que espera a que el otro inicie la comunicación.

La figura 2.23 muestra un esquema de cómo se utiliza los sockets para intercambiar información entre dos computadoras.

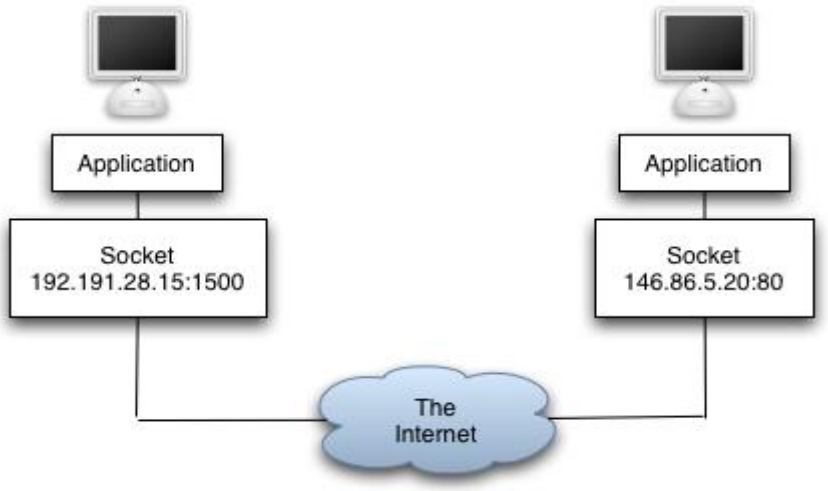


Figura 2.23 – Comunicación entre sockets [23]

En Java los sockets son creados con la ayuda de la clase Socket y estas se encuentran en el paquete java.net. Hay clases separadas en el paquete java.net para la creación de sockets en TCP y UDP.

Una dirección de socket está conformado por una tripleta: protocolo que usa para el transporte de datos, dirección IP y un número de puerto (el número que identifica una aplicación de Internet en particular).

2.3.7. CIFRADO DE DATOS

El cifrado de datos se basa en transmitir el contenido de los datos de una manera distorsionada desde una computadora hacia otra, a través de un canal, para que no sea legible por un sniffer de paquetes.

El cifrado de datos ofrece las siguientes ventajas:

- Autenticidad.
- Confidencialidad.
- Integridad.

Los métodos de cifrado pueden dividirse en dos grandes grupos:

- Simétrico.
- Asimétrico.

El cifrado simétrico utiliza la misma clave para cifrar y descifrar el mensaje y se debe intercambiar la clave entre los equipos por medio de un canal seguro para que ambos extremos puedan manejar los datos.

El cifrado asimétrico utiliza dos claves diferentes que poseen una propiedad fundamental: una clave puede descifrar lo que la otra ha cifrado.

En este tipo de criptografía una de las claves se llama clave privada y es usada por el propietario para cifrar datos, mientras que la otra se llama clave pública y es usada para descifrar el mensaje o viceversa. Figura 2.24.

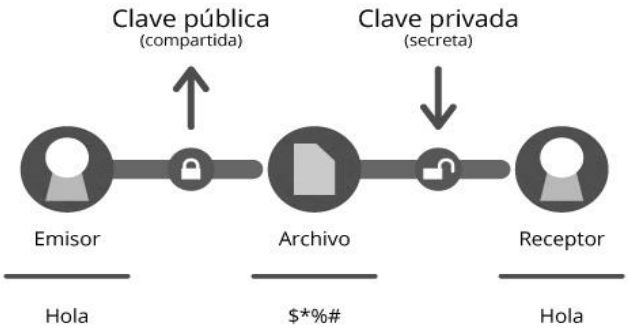


Figura 2.24 – Cifrado asimétrico [24]

2.4 SOFTWARE UTILIZADO PARA EL DISEÑO

En esta sección se detallan las aplicaciones que se requirieron para la implementación y pruebas del proyecto.

2.4.1 QUARTUS II

El software Quartus II es el entorno más completo disponible para diseño de sistemas programables en un chip (SOPC), ya que se adapta fácilmente a las necesidades se requiere en un diseño. Es utilizado para implementar circuitos en dispositivos FPGA de Altera.

Este sistema incluye un soporte completo de todos los métodos populares de descripción de circuitos en un sistema CAD (Computer Aided Design), como la introducción del circuito en forma de diagrama de bloques (esquema) o especificación del circuito usando lenguaje de descripción de hardware como Verilog o VHDL.

En la figura 2.25 se muestra la ventana de inicio de Quartus II.

Figura 2.25 – Ventana de trabajo QUARTUS II [25]

2.4.1.1 QSYS

Qsys es una herramienta de integración de sistema que viene incluido como parte del software Quartus II, esta permite el diseño de hardware de nivel de sistema en un alto nivel de abstracción a través de una interfaz gráfica.

En la figura 2.26 se muestra la ventana de inicio de Qsys.

Figura 2.26 – Ventana de inicio de QSYS [26]

Permite la incorporación de módulos de diseño propios programados en lenguaje HDL para personalizar los diversos periféricos.

De esta manera Qsys facilita la reutilización de diseños, componentes y sistemas personalizados, pudiendo utilizar estos componentes con los componentes que Altera y los desarrolladores de terceros proporcionan. En algunos casos, se puede implementar un diseño completo utilizando componentes de la biblioteca de Altera.

Durante la generación del sistema, QSYS crea automáticamente la lógica de interconexión de alto rendimiento de la conectividad que se especifique, eliminando la tarea lenta y propensa a errores de la escritura HDL para especificar las conexiones a nivel de sistema.

Se pueden diseñar sistemas basado en el microprocesador NIOS II.

2.4.2 NIOS II SOFTWARE BUILD TOOL PARA ECLIPSE

Es un entorno de desarrollo integrado (IDE), compuesto por un conjunto de herramientas de programación multiplataforma de código abierto.

Escrito principalmente en Java, Eclipse puede usarse para desarrollar aplicaciones en ese lenguaje o por medio de varios plug-ins puede ser utilizado también para desarrollar aplicaciones en otros lenguajes de programación tales como: Ada, ABAP, COBOL, PHP, Python, Ruby, etc.

NIOS II SBT para Eclipse es un conjunto de plugins que se agrega a la infraestructura de Eclipse, para programar los sistemas basados en el procesador NIOS II, logrando así crear interfaces usuario-hardware que se pueden escribir en distintos lenguajes como: Java, C, C++, Lenguaje Ensamblador.

En la figura 2.27 se muestra la ventana de inicio de Eclipse.

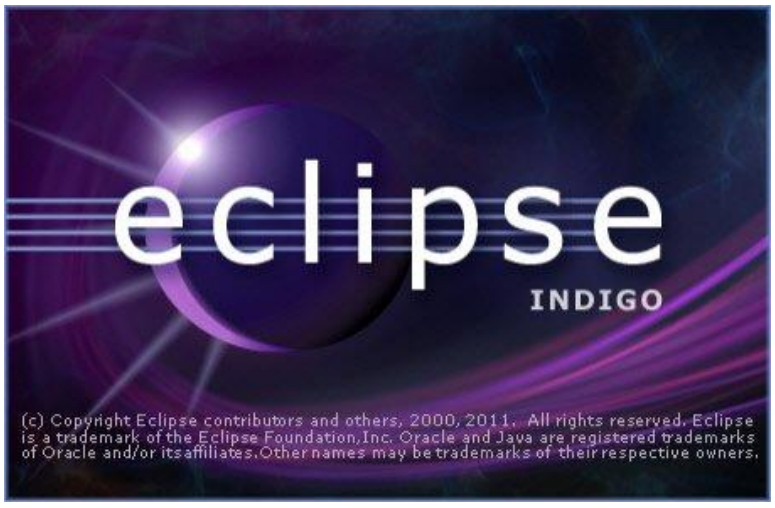


Figura 2.27 – Ventana de inicio de eclipse NIOS II SBT [27]

Este software se encuentra totalmente integrada en el paquete de herramientas proporcionado por Altera, lo que facilita su uso durante el desarrollo del sistema completo.

Esta herramienta puede llevar a cabo la mayoría de tareas de desarrollo de software dentro de Eclipse, incluyendo la creación, edición, ejecución, depuración y perfilado de los programas.

2.4.3 NETBEANS

NetBeans es un ambiente de desarrollo integrado (IDE) de código abierto, utilizado para el desarrollo de aplicaciones escritas principalmente en lenguaje de programación Java, pero también se puede utilizar con otros lenguajes de programación.

La plataforma NetBeans utiliza componentes, también conocidos como módulos, para el desarrollo del software. NetBeans instala módulos dinámicamente y permite a los usuarios descargar características actualizadas y mejoras autenticadas digitalmente.

En la figura 2.28 se muestra la ventana de inicio de NetBeans IDE 7.1.

Figura 2.28 – Ventana de inicio de NetBeans [28]

Esta herramienta soporta el desarrollo de todo tipo de aplicaciones Java: Java SE, Java ME, web, EJB y aplicaciones móviles, etc.

Entre las características destacadas de NetBeans encontramos: NetBeans Profiler, interfaz gráfica de usuario (GUI), colaboración de desarrolladores y NetBeans Platform.

El NetBeans Profiler ayuda a optimizar nuestras aplicaciones e intentar que se ejecuten más rápido con el mínimo uso de memoria, evitando así cuellos de botellas y pérdidas de

memoria, la interfaz gráfica (GUI) nos permite diseñar y desarrollar interfaces de usuarios gráficas, haciéndolo tan simple como arrastrar y posicionar componentes GUI.

Otra característica es que desarrolladores pueden trabajar juntos en proyectos de código abierto, a través de una estrecha integración con el proyecto Kenai. Además desde NetBeans podemos conectarnos a distintos sistemas gestores de bases de datos, como pueden ser Oracle, MySQL y demás, ver las tablas, realizar consultas y modificaciones y todo ello integrado en el propio IDE.

Por último los APIs provistos ayudan a hacer más fácil el manejo ciertas tareas de las aplicaciones de escritorio como: menús, gestión de ventanas, acciones, acceso a ficheros, etc.

2.4.4 MYSQL WORKBENCH

MySQL es un sistema de gestión de bases de datos relacional, multihilo y multiusuario. Una base de datos es una colección organizada de información de tal manera que permite una recuperación rápida y eficaz de los datos almacenados.

En la figura 2.29 se muestra la ventana de inicio de MySQLWorkbench.

Figura 2.29 – Ventana de inicio de MySQL Workbench [29]

Los sistemas de gestión de bases de datos (DBMSs) son herramientas que permiten definir, crear, consultar, actualizar y administrar una base de datos especialmente diseñados para conectarla con otras aplicaciones e interactúen con los usuarios.

En general, los DBMSs Entre los sistemas de gestión de bases de datos más conocidos se encuentran Oracle, MySQL, dBASE, PostgreSQL, ect.

MySQL Worlbench nos permite diseñar, crear y administrar gráficamente bases de datos MySQL, haciendo de esta manera más fácil el manejo de la base de datos.

Algunas ventajas que presenta esta herramienta es que se puede realizar una representación visual de tablas, vistas, procedimientos almacenados y claves foráneas de la base de datos. Además es capaz de sincronizar el modelo en desarrollo con la base de datos real.

Este software es libre y de código abierto.

2.4.5 PROTEUS

Proteus es un software que sirve para simular circuitos electrónicos, capturar esquemáticos y diseñar circuitos impresos (PCB).

En la siguiente figura 2.30 se muestra la ventana de inicio de Proteus.

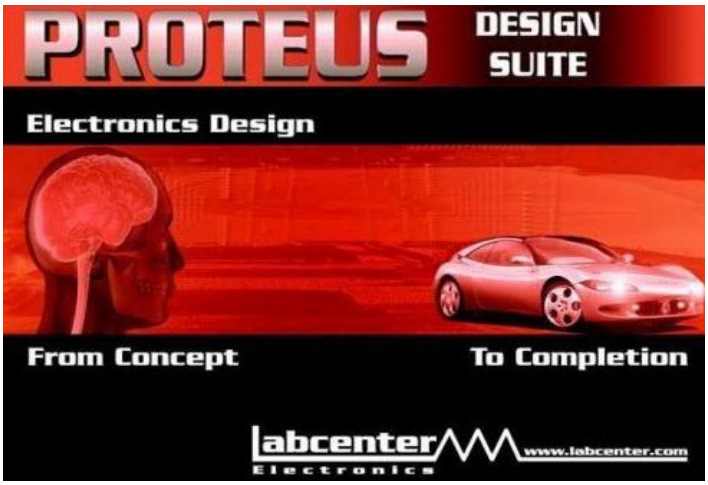


Figura 2.30 – Ventana de inicio de Proteus [30]

Ésta herramienta se apoya en el uso de dos programas adicionales elaboradas por el mismo fabricante Labcenter Electronics:

ISIS (Intelligent Schematic Input System) permite el diseño y la simulación de circuitos electrónicos en tiempo real, usando un amplio Stack de componentes, en donde se puede encontrar desde una simple resistencia a complejos circuitos integrados.

ARES (Advanced Routing and Editing Software) proporciona una herramienta de ruteo de pistas para circuitos impresos, de forma automática, usando los métodos Autorouter y/o Electra Autorouter, o de forma manual usando nuestro propio método de ruteo. Además, permitir una de visualización en 3D de placas PCB diseñadas.

2.4.6 WIRESHARK

Wireshark es sniffer de paquetes, que monitorea el flujo de datos a través de un enlace y realiza un análisis de las estructuras de los protocolos usados en la red, permitiendo la solución de problemas que se presentan comúnmente en comunicaciones de datos.

Este software es libre y de código abierto, soporta una interfaz gráfica de usuario (GUI) y está disponible para Linux, Unix y Windows.

En la siguiente figura 2.31 se muestra la ventana de inicio de Wireshark.

Figura 2.31 – Ventana de inicio de Wireshark [31]

Wireshark nos permite:

- Capturar paquetes en vivo desde una interfaz de red.
- Presentar los paquetes con información detallada de los mismos.
- Abrir y guardar paquetes capturados.
- Importar y exportar paquetes en diferentes formatos.
- Filtrar información de paquetes.
- Resaltar paquetes dependiendo del filtro.

- Crear estadísticas.
- Analizar nuevos protocolos mediantes la creación de plugins.

CAPÍTULO 3

3. DISEÑO E IMPLEMENTACIÓN

Para realizar un seguimiento detallado del diseño, implementación y ensamblaje del proyecto, el capítulo se dividió en 5 secciones:

- Diseño general.
- Implementación del hardware en la tarjeta DE2.
- Desarrollo de la aplicación en la tarjeta DE2.
- Descripción de la aplicación usada en la PC.

- Ensamblaje del sistema.

3.1. DISEÑO GENERAL

En esta sección se muestra el esquemático, lista de recursos utilizados, diagrama de bloque, diagrama funcional y una descripción del sistema.

3.2. ESQUEMÁTICO

En la figura 3.1 se muestra el diagrama esquemático del sistema de autenticación de usuarios implementado.

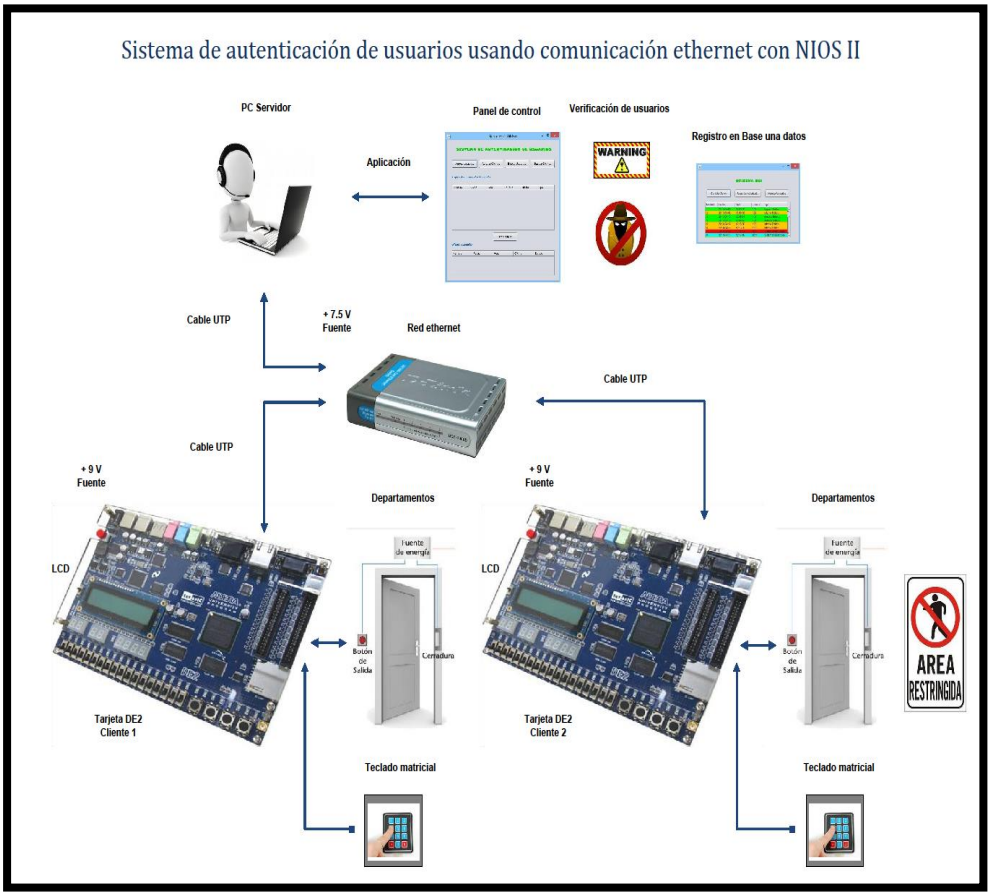


Figura 3.1 – Diagrama esquemático del sistema.

Como se observa en la figura 3.1, el sistema de autenticación de usuarios fue diseñado para ser implementado en condiciones reales. Sin embargo, por motivos de desarrollo, las pruebas se las realizó en un ambiente de laboratorio, simulando las componentes actuadores del sistema y creando nuestra propia red LAN para enviar los datos.

3.2.1. LISTA DE RECURSOS UTILIZADOS

Los materiales que utilizamos son:

- 1 Computadora.
- 1 Switch D-Link de 5 puertos.
- 2 Tarjetas DE2.
- 2 Teclados matriciales 4x4.
- 2 Buses de datos de 40 pines.
- 3 Cables UTP tipo directo.

Los recursos usados de la tarjeta DE2 son:

- LCD.
- Puerto ethernet.
- Conectores GPIO.
- Puerto JTAG – UART.
- 1 Led rojo.
- 1 Led verde.

1 Push Button.

Las características de la computadora para ejecutar la aplicación son:

- Soporte para aplicaciones Java (JDK).
- Software MySQL instalado.
- Base de datos del sistema creada.

El paquete de software utilizados para elaborar el proyecto son:

- Quartus II 12.1.
- Eclipse IDE 3.7.2.
- NetBeans IDE 7.1.
- MySQL 5.6.
- Proteus 7.6.

El software usado para realizar el monitoreo de paquetes en las pruebas del sistema es:

- Wireshark 1.10.4.

Para implementar el sistema de autenticación de usuarios, se tomó como referencia el modelo TCP/IP.

Recursos utilizados y protocolos implementados en cada capa:

Capa física:

- Chip DM9000A integrado en la Tarjeta DE2.

Capa de enlace:

- Protocolo ARP para enlazar direcciones lógicas con direcciones físicas.
- Protocolo Ethernet para comunicar los dispositivos en la red LAN.

Capa de red:

- Protocolo IP para enrutar datos dentro de la red LAN.

Capa de transporte:

- Protocolo UDP para el transporte de datos entre aplicaciones.

Capa de aplicación:

- Elaboración de una aplicación en lenguaje C usando el stack de funciones del datasheet del DM9000A para transmitir o recibir paquetes de datos en la tarjeta DE2.

- Construcción de una aplicación en lenguaje Java usando el stack de funciones de la librería Java.net para recibir o transmitir paquetes de datos en la PC.

- Desarrollo de un algoritmo de encriptación simétrico para enviar los datos cifrados, elaborada en lenguaje C para la tarjeta DE2 y en lenguaje Java para la PC.

3.2.2. DIAGRAMA DE BLOQUES

En el diagrama de bloques de la figura 3.2 se puede observar con mejor detalle cada uno de los recursos y conexiones utilizadas dentro y fuera de la tarjeta DE2.

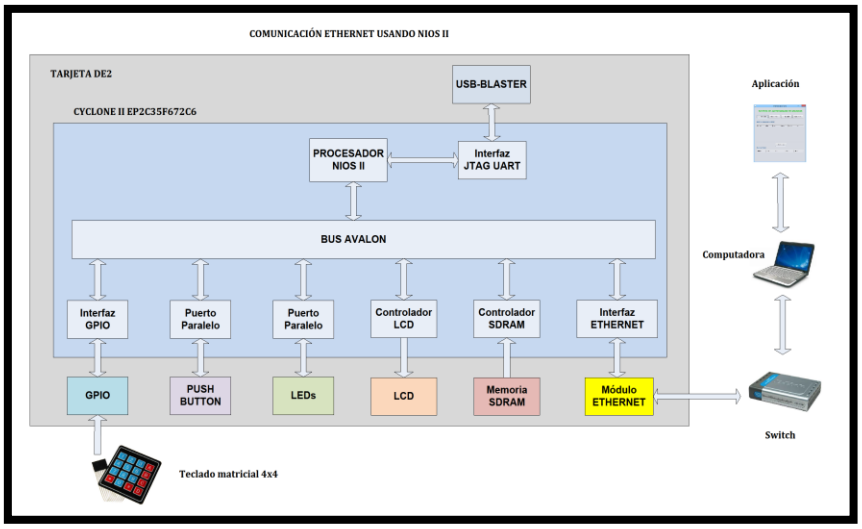


Figura 3.2 – Diagrama de bloques del sistema.

Además, se puede percibir a partir de la figura 3.2 que el sistema se divide en 5 etapas:

- Teclado: Permite al usuario interactuar con el sistema ingresando datos a través de un teclado matricial 4x4.

- Tarjeta DE2: Procesa los datos recibidos desde el teclado, los empaqueta para poder transmitirlos a través de la red LAN y realiza la ejecución de las órdenes recibidas desde la PC.

- Red ethernet: Sirve como medio de transporte para el envío de mensajes entre la tarjeta DE2 y la PC servidor.

- PC: Ejecuta la aplicación de autenticación de usuarios.

- Aplicación: realiza el proceso de autenticación de usuarios y envía mensajes de respuesta a la tarjeta DE2. Además, efectúa la gestión de la información con la base de datos.

3.2.3. DIAGRAMA FUNCIONAL

Para explicar el proceso de comunicación ethernet que ocurre entre la PC y la tarjeta DE2, el diagrama funcional del sistema se lo ha dividido en dos partes:

En la figura 3.3 y 3.4 se muestra el diagrama funcional de la aplicación de la tarjeta de la tarjeta DE2:

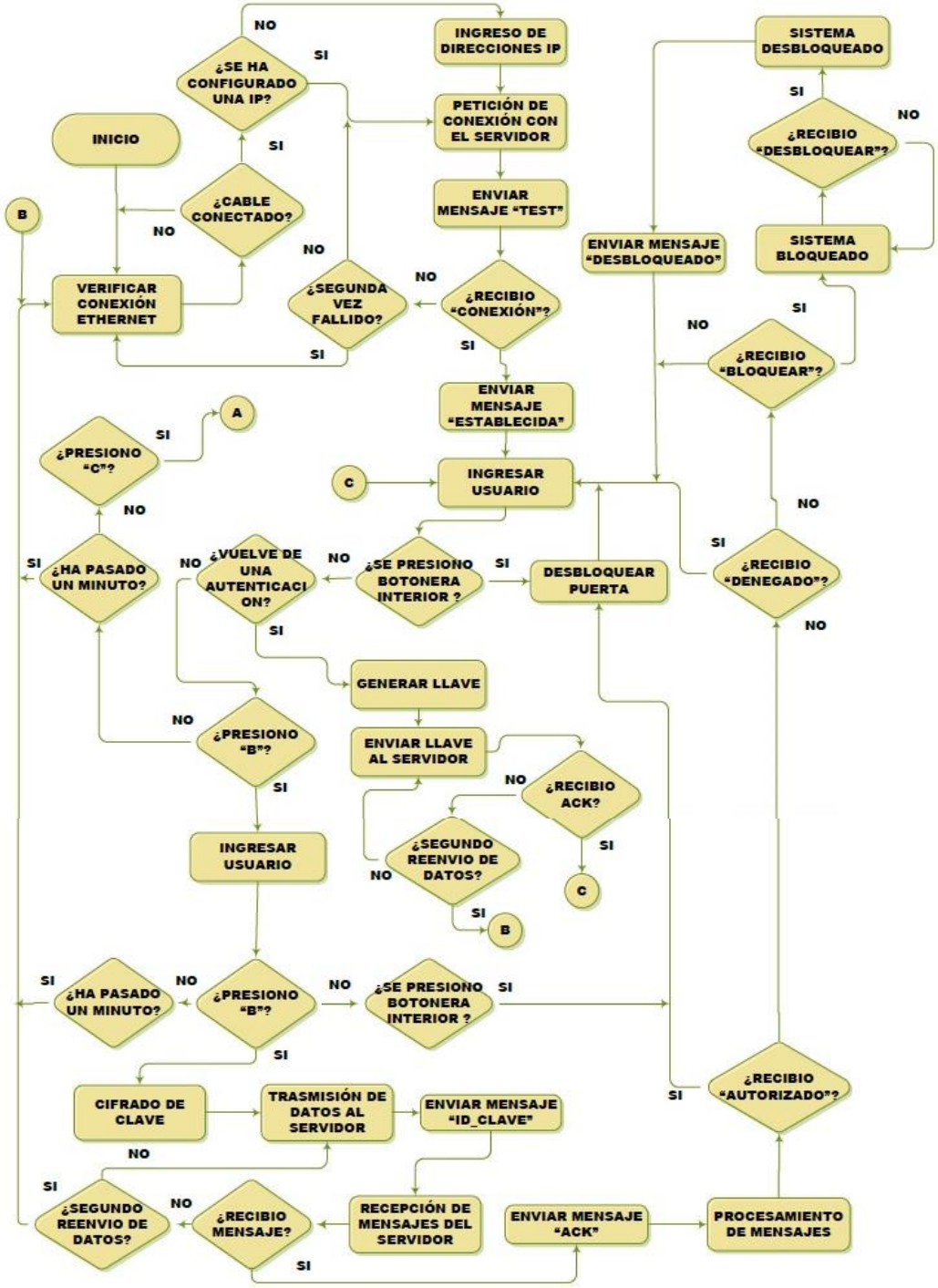


Figura 3.3 – Diagrama de flujo de la aplicación de la tarjeta DE2 - 1.

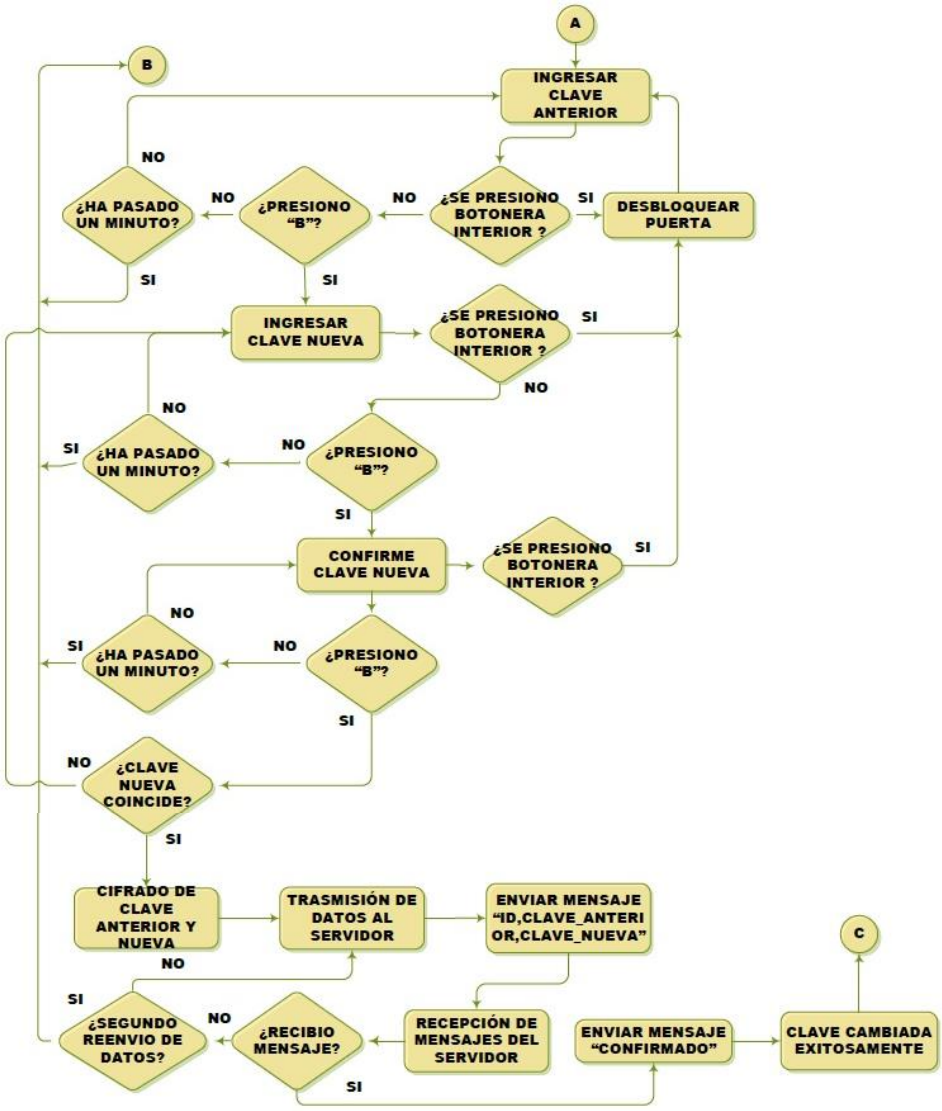


Figura 3.4 – Diagrama de flujo de la aplicación de la tarjeta DE2 - 2.

En la figura 3.5 se muestra el diagrama funcional del proceso de autenticación efectuada en la computadora servidor:

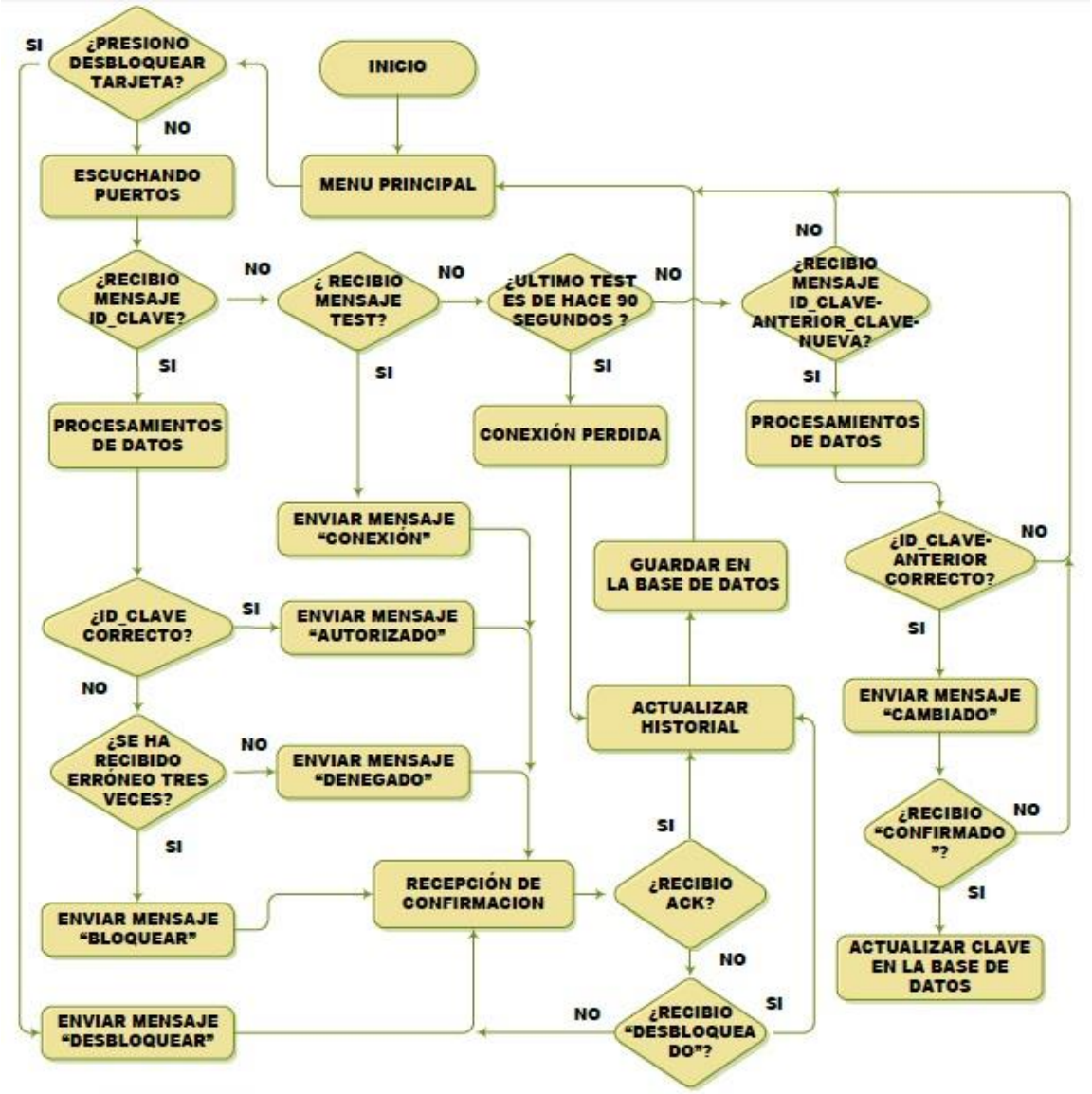


Figura 3.5 – Diagrama de flujo de la aplicación de la PC.

3.2.4. DESCRIPCIÓN DE FUNCIONAMIENTO DEL SISTEMA

Observando las figuras 3.3, 3.4 y 3.5 se puede tener una noción rápida de la lógica de la aplicación, pero debido al espacio reducido que se tiene en un diagrama de flujo, se realiza una breve descripción del funcionamiento, para explicar con mejor detalle las características, los tipos de mensajes y datos que maneja el sistema.

Funcionamiento de la tarjeta DE2

La aplicación en la tarjeta DE2 realiza en 4 procesos principales la autenticación de usuarios:

1. Verificación de conexión con el servidor:

Verifica si el cable de red está conectado a la tarjeta DE2, si no lo está, espera hasta que se conecte uno para poder continuar. El cable de red debe ser tipo directo.

Luego se pide configurar dirección IP de la tarjeta DE2 y de la PC servidor para empezar a testear la conexión. Estas direcciónes IP deben estar de acuerdo con las políticas de direccionamiento IP

establecida por el administrador de red para que no surgan problemas de conectividad entre equipos conectados a la red LAN.

Si el testeo fue exitoso, continúa al siguiente proceso, si fue fallido vuelve a pedir ingreso de direcciones IP.

2. Ingreso de datos:

Envía una llave aleatoria de 3 dígitos al servidor cada vez que entra al menú principal de ingreso de datos, para indicarle como descifrar los datos posteriores que la tarjeta DE2 le envíe al servidor.

Luego, una vez en el menú principal, permite a los usuarios ingresar un ID de 3 dígitos y elegir entre autenticarse o cambiar su clave.

Si el usuario elige autenticarse, la aplicación le solicitará que ingrese su clave, pero si escoge cambiarla deberá ingresar la clave anterior y la clave nueva.

El ID y la clave solo pueden contener números debido a que se usa las letras del teclado matricial 4x4 para opciones de configuración, borrado y confirmación de datos.

3. Proceso de comunicación:

La tarjeta DE2 cifra los datos con la llave trasmitida con anterioridad, la empaqueta y procede a enviarla a través de la red Ethernet a la PC servidor. Luego, espera hasta recibir un mensaje de respuesta desde la PC, con el contenido del mensaje dependiendo de cuál fue el estado de autenticación del usuario en el servidor o si la clave fue cambiada correctamente.

Si no recibe respuesta la tarjeta DE2 se asume el paquete perdido y se reenvía nuevamente la petición de autenticación o de cambio de clave al servidor. Si luego de esto no llega respuesta, se asume conexión perdida.

4. Fase actuador:

La tarjeta DE2 realiza las siguientes acciones al momento de recibir los siguientes mensajes:

- Enciende momentáneamente un LED verde, si recibió el mensaje “AUTORIZADO”, simulando que se abrió la puerta correctamente.

- Enciende por unos instantes un LED rojo, si recibió el mensaje “DENEGADO”, indicando que se ingresó incorrectamente la clave.

- Prende un LED rojo indefinidamente, si recibió el mensaje “BLOQUEADO”, indicando que se ha bloqueado el sistema por el tercer reingreso de clave erróneo desde un mismo ID de usuario.

- Desbloquea el sistema, si recibe un mensaje desde la PC con el mensaje “DESBLOQUEAR”, apagando el LED rojo para luego volver al proceso de ingreso de datos.

- Abre la puerta si se aplasta el PUSH BUTTON de la tarjeta DE2, que simula el botón de abrir que se suele encontrar dentro de la habitación de estos sistemas de autenticación.

En cada una de estas situaciones se muestran mensajes en el LCD correspondientes a la situación actual del sistema.

Funcionamiento de la Aplicación en Java

Los procesos que se ejecuta en la aplicación de la PC son 4, detallados a continuación:

1. Registro y Consulta de Usuarios y Oficinas:

Este proceso no se puede observar en el diagrama de flujo mostrado en la figura 3.4 de la sección anterior, pero se ejecuta en paralelo con el estado ‘Escuchando Paquetes’ y es la que permite interactuar al usuario con la base de datos cuando la aplicación se está ejecutando.

En este proceso se puede hacer el registro de nuevos usuarios y oficinas, consultas y modificaciones de usuarios y oficinas ya existentes dando clic en el botón correspondiente de la interfaz de usuario de la aplicación en cualquier momento que se desee.

2. Escuchando Paquetes:

En este proceso la aplicación escucha la red en donde se encuentran las oficinas a través del puerto 5005, hasta recibir un paquete de cualquier oficina del sistema.

Si recibe un mensaje, deja de escuchar la red hasta terminar de verificar los datos del mensaje y enviarle el estado de autenticación del usuario a la tarjeta DE2.

3. Procesamiento de los datos y envío de estado de autenticación:

Cuando se recibe un paquete, la aplicación verifica los datos del usuario con la base de datos y responde con un mensaje dependiendo del estado de autenticación como se muestra en la figura 3.5.

Después que la aplicación en la PC ha respondido con un mensaje, espera a que la tarjeta DE2 le confirme que le llegó la respuesta a través de un mensaje ACK para poder continuar con el siguiente proceso.

4. Almacenamiento en la Base de Datos:

Cuando se recibe el mensaje de confirmación de la tarjeta DE2, la aplicación registra el intento de autenticación sea exitoso o fallido en la base de datos, además lo muestra en el historial de intentos de la ventana principal. Finalmente, regresa al estado ‘Escuchando Paquetes.

3.3. IMPLEMENTACIÓN DEL HARDWARE EN LA TARJETADE2

En esta sección, se muestra paso a paso como agregar y configurar el hardware de la tarjeta DE2 con el software Quartus II y Qsys.

3.3.1. CREACIÓN DEL PROYECTO EN QUARTUS II

El hardware de nuestro sistema basado en el microprocesador NIOS II fue desarrollado en Quartus II. A continuación, se muestra como se crea un nuevo proyecto con este software.

Al iniciar Quartus, se despliega un menú como se muestra en la figura 3.6, para crear un proyecto nuevo utilizando el asistente.

File > New Project Wizard

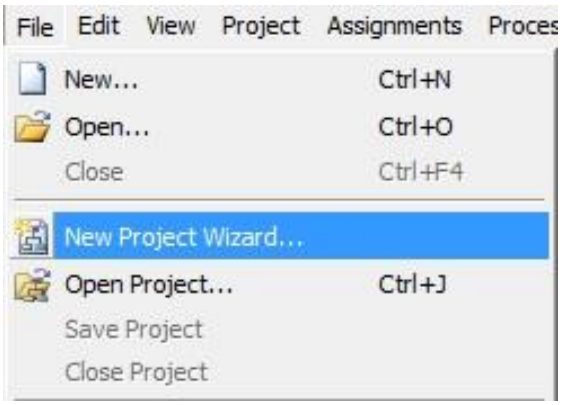


Figura 3.6 – Crear un nuevo proyecto en Quartus II.

En la primera ventana del asistente como se observa en la figura 3.7, elegimos el directorio de trabajo, donde se guardan todos los archivos concernientes al proyecto y el nombre del proyecto.

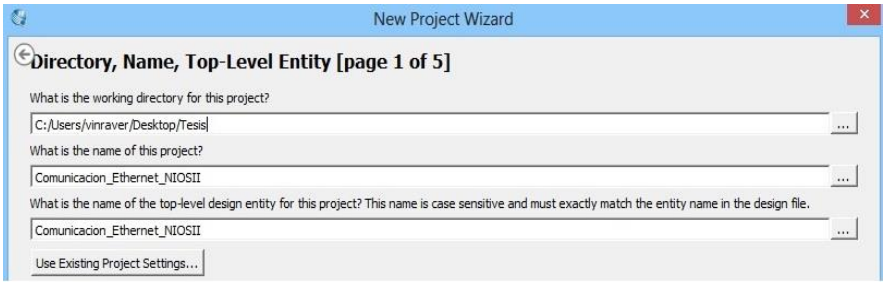


Figura 3.7 – Definir el nombre del nuevo proyecto.

En la siguiente ventana que se muestra en la figura 3.8, nos consulta si vamos a agregar algún archivo de diseño a nuestro proyecto, este puede ser un archivo VHDL que luego necesitemos usar. En este paso no agregamos nada.

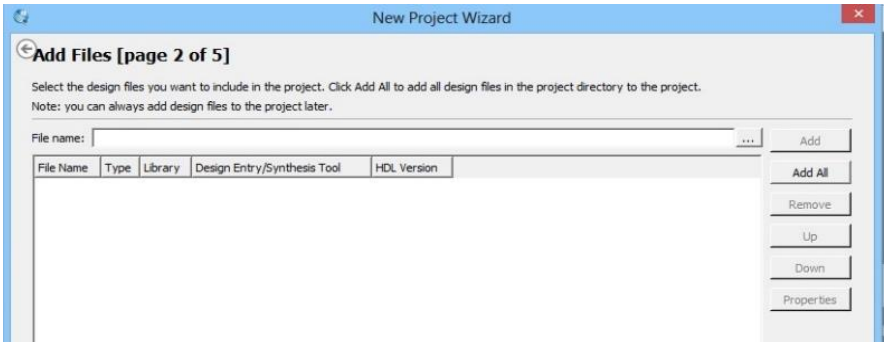


Figura 3.8 – Agregar archivos al proyecto.

Luego, tal como se ve en la figura 3.9, seleccionamos el dispositivo que vamos a utilizar. En nuestro caso, en la sección Device Family elegimos Cyclone II y luego en la lista Available Devices escogemos EP2C35F672C6.

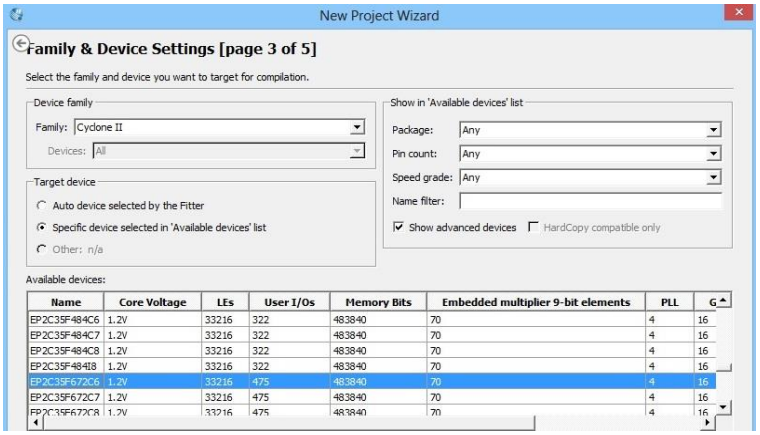


Figura 3.9 – Escoger el dispositivo a usar.

En la siguiente ventana podemos seleccionar diversas herramientas para la compilación y simulación del sistema. Para nuestro proyecto utilizamos las herramientas estándares que nos ofrece Quartus II, por lo tanto no modificamos nada.

Por último el asistente nos muestra tal como se ve en la figura 3.10, un resumen de las opciones seleccionadas de nuestro nuevo proyecto.

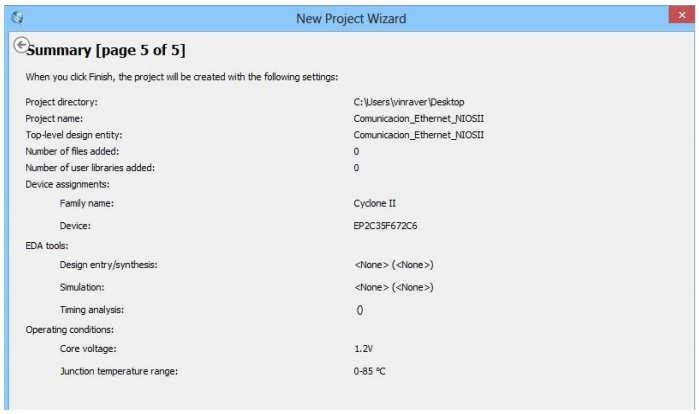


Figura 3.10 – Resumen de las opciones seleccionadas.

Luego de terminar estos pasos, creamos un archivo VHDL escogiendo la opción correspondiente tal como se observa en la figura 3.11, donde posteriormente precederemos a realizar las

conexiones del sistema NIOS II con los pines físicos del dispositivo FPGA.

File > New > VHDL File

.

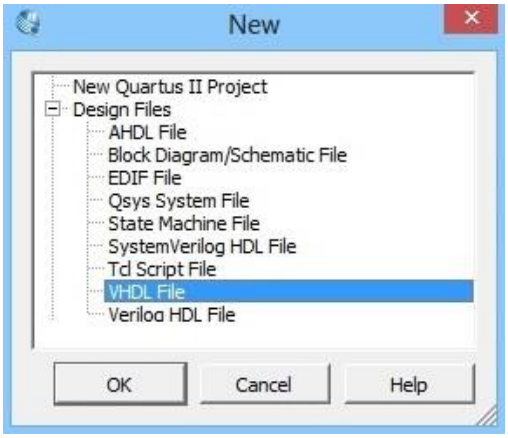


Figura 3.11 – Ventana para la creación de archivo VHDL.

3.3.2. CONSTRUCCIÓN DEL HARDWARE EN QSYS

En esta sección creamos un sistema basado en el microprocesador NIOS II usando el software Qsys que se encuentra en Quartus II.

Para acceder a esta herramienta nos situamos en la barra de menú de Quartus II como se muestra en la figura 3.12 y seleccionamos:

Tools > Qsys

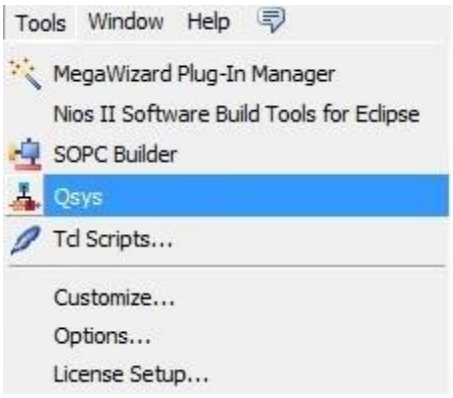


Figura 3.12 – Ventana para abrir la herramienta QSYS.

Después nos aparece la ventana principal de trabajo donde podemos crear un nuevo sistema:

File > New System

En la ventana principal de Qsys como se observa figura 3.13 hay una sección para agregar componentes, ubicada a la izquierda de la pantalla, donde tenemos todas las librerías necesarias para poder agregar módulos funcionales a nuestro sistema como: puertos GPIO, memorias, procesadores, LCD, entre otros.

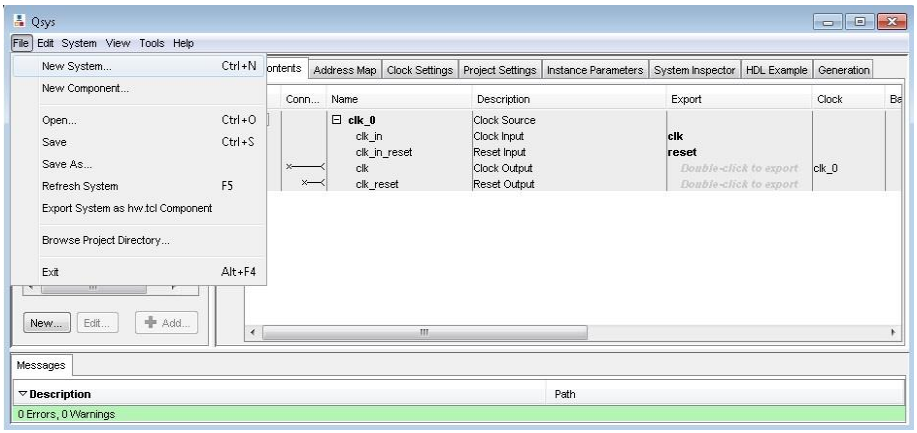


Figura 3.13 – Ventana principal de trabajo QSYS.

3.3.2.1. MÓDULOS GENERALES

Para un correcto funcionamiento del sistema es necesario añadir varios componentes generales como: procesadores, memorias, entre otros.

3.3.2.1.1. SYSTEM ID PERIPHERAL

El System Peripheral permite a las herramientas de desarrollo de software para NIOS II saber si la aplicación pertenece al hardware objeto. Es uno de los periféricos más importantes, el sistema solo

puede tener un componente de este tipo y debe llamarse sysid, caso contrario no será reconocido.

En la figura 3.14 se observa las opciones del System ID peripheral.

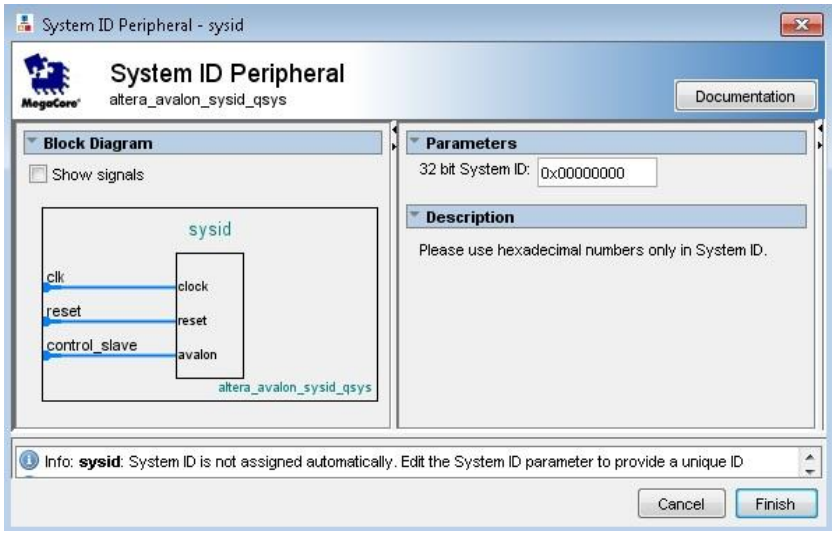


Figura 3.14 – Componente system ID Peripheral.

3.3.2.1.2. SDRAM CONTROLLER

La tarjeta DE2 contiene un chip SDRAM que puede almacenar 8Mbytes de datos. Esta memoria se organiza como 1M x 16 bits x 4 bancos. El chip

SDRAM requiere un cuidadoso control de temporización. Para facilitar el acceso al chip SDRAM, Qsys implementa un circuito controlador de SDRAM. Este circuito genera las señales necesarias para tratar con este chip.

Para añadir este componente seleccionamos Memories and Memory Controllers > External Memory Interfaces > SDRAM Interfaces > SDRAM Controller. A continuación aparece una ventana con las opciones que se pueden configurar tal como se muestra en la figura 3.15. Seleccionamos Custom de la lista desplegable en Presets, ancho de banda de los datos en 16 bits los demás valores quedan por defecto.

Figura 3.15 – Componente SDRAM Controller.

3.3.2.1.3. NIOS II PROCESSOR

Todo sistema embebido es controlado por un procesador, que es el encargado de interactuar con las diferentes interfaces conectadas a él, enviando las señales correspondientes a cada uno de sus periféricos para completar las tareas asignadas.

Para agregar este componente al sistema, en la sección Component Library seleccionamos Embedded Procesor > NIOS II Processor.

A continuación, en la figura 3.16 se muestra la ventana para agregar el NIOS II Procesor al sistema.

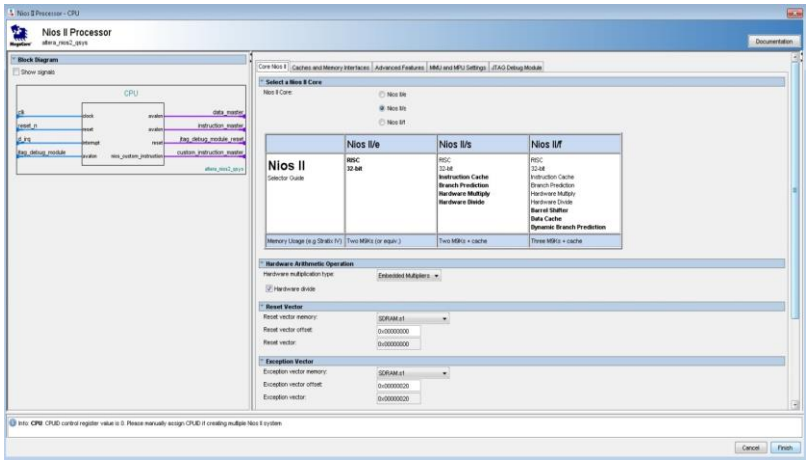


Figura 3.16 – Componente NIOS II Processor.

En esta ventana se realiza las siguientes configuraciones:

Se selecciona NIOS II/s como el tipo de procesador NIOS II y en la configuración de vector de reset y de excepción se escoge la memoria SDRAM que se añadió anteriormente, los demás valores quedan por defecto.

3.3.2.1.4. JTAG UART

Este módulo nos permite tener una comunicación entre la computadora y el procesador NIOS II, realizar cambios en la configuración del FPGA y posteriormente programar el chip. Para incluir este módulo en el sistema seleccionamos Interface Protocols > Serial > JTAG UART en la sección Component Library de la ventana principal de Qsys.

En la siguiente figura 3.17 se muestra la ventana para añadir este módulo, en donde se selecciona 64 bits como el ancho del buffer de los datos que se envían del Avalon al JTAG y del JTAG al Avalon, para que se establezca una conexión de forma correcta.

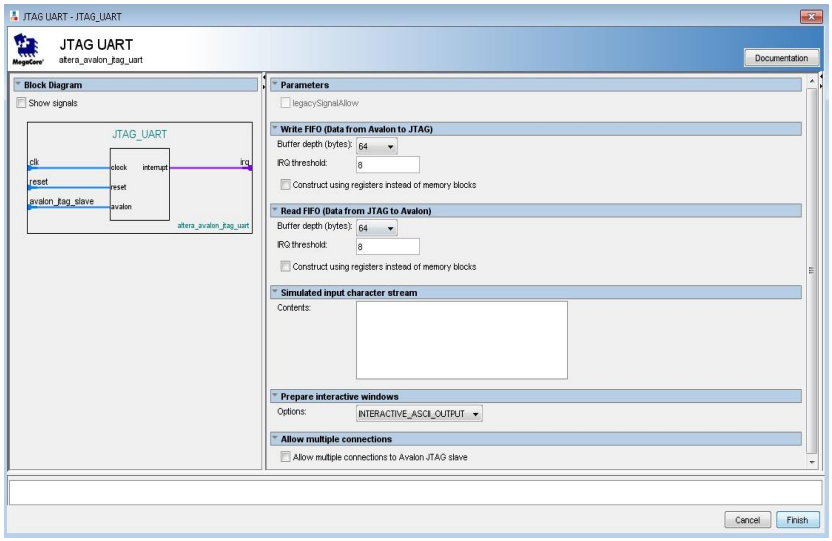


Figura 3.17 – Componente JTAG UART.

3.3.2.1.5. INTERVAL TIMER

La mayoría de los sistemas de control utilizan un componente temporizador para permitir un cálculo preciso de tiempo.

Para añadir este componente al sistema seleccionamos Peripherals > Microcontroller Peripherals > Interval Timer de la librería de componentes.

Se configura un periodo de 125 milisegundos, los otros valores no se cambian.

En la figura 3.18 se muestra la ventana con las opciones del Interval Timer.

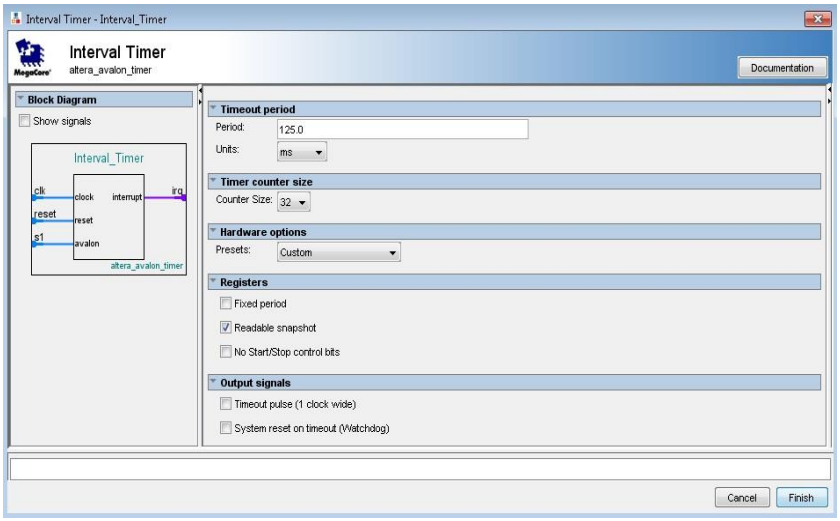


Figura 3.18 – Componente interval Timer.

3.3.2.2. MÓDULOS ESPECÍFICOS

Los módulos que se agregan en esta sección son esenciales para desarrollar la aplicación del proyecto.

3.3.2.2.1. MÓDULO DM9000A

El módulo DM9000A provee una interfaz entre el procesador NIOS II y el chip Davicom DM9000A, permitiendo acceder a los registros internos del chip. Al leer y escribir en los registros, es posible completar tareas como la inicialización del chip, envío y recepción de paquetes, además proporciona todas las señales lógicas requeridas hacia y desde el chip.

La figura 3.19 muestra el módulo DM9000A, éste no tiene parámetros configurables

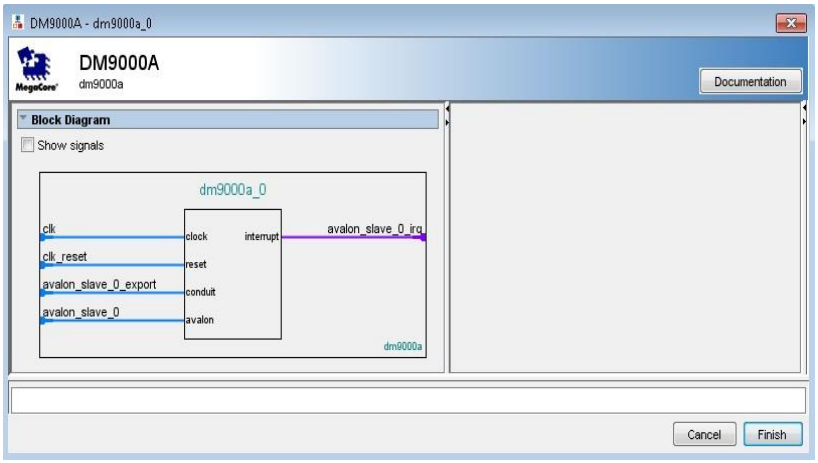


Figura 3.19 – Componente DM9000A.

Para integrar este componente al sistema seleccionamos Terasic Technologies Inc > DM9000A en la librería de componentes.

Es necesario suministrarle un reloj de 100MHz para su correcto funcionamiento.

3.3.2.2.2. 16X2 CHARACTER DISPLAY

El módulo 16x2 Character Display facilita la comunicación entre el LCD y la tarjeta DE2, éste se comunica con la pantalla a través de un conjunto de instrucciones y el registro de datos. Además incluye circuitos que inicializan automáticamente el LCD cuando Qsys es reiniciado.

La figura 3.20 muestra la configuración de la interfaz LCD.

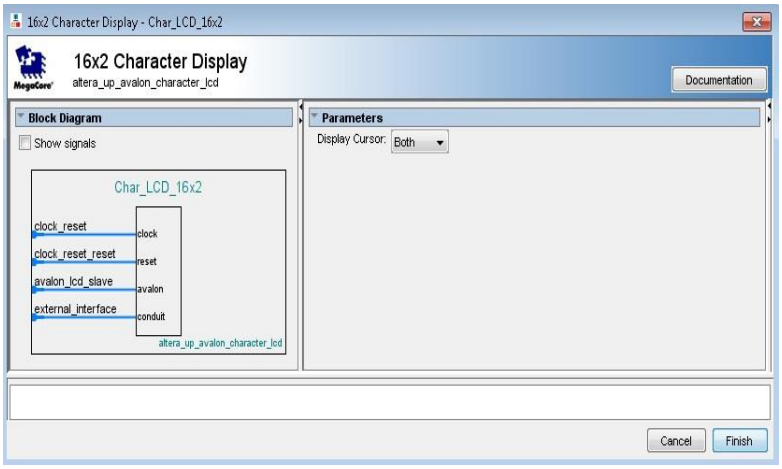


Figura 3.20 – Componente 16x2 Character Display.

En Qsys se indica los parámetros de configuración a nivel de hardware para la interfaz LCD, el control de visualización de los datos se lo realiza mediante un software con librerías que se genera una vez terminado el diseño.

Para añadir este módulo al sistema, seleccionamos en la librería de componentes de Qsys University Program > Audio & Video > 16x2 Character Display.

Para un correcto funcionamiento de esta componente es necesario proporcionarle una frecuencia de reloj de 50 MHz.

3.3.2.2.3. PARALLEL PORT

Este módulo provee una interfaz sencilla para propósitos generales de entrada/salida. Algunos ejemplos incluyen:

- Control de LEDs
- Adquisición de datos de los switches
- Control de displays

Los parallel ports proporcionan hasta 32 puertos de entrada/salida.

Para incluir este módulo al sistema seleccionamos University Program > Generic IO > Parallel Port en la librería de componentes de Qsys.

En el sistema necesitaremos agregar 3 componentes Parallel Port para controlar un LED rojo, verde y un pushbutton.

En la figura 3.21 se muestra la configuración para el LED rojo.

I/O Device: LEDs Colour: Red

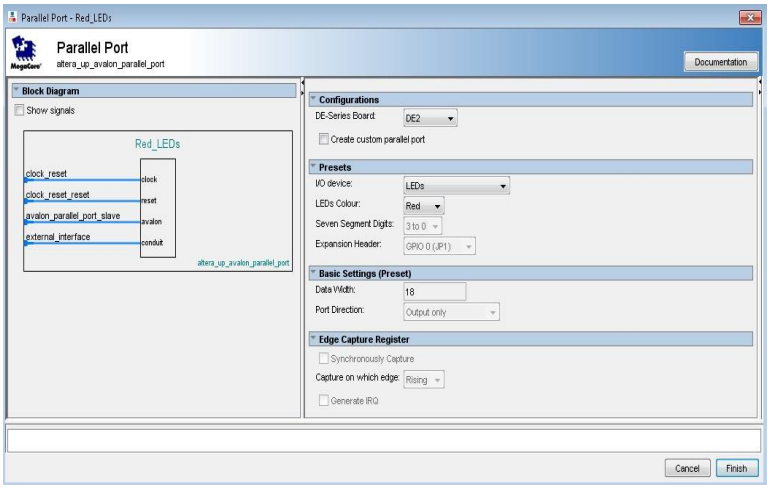


Figura 3.21 – Componente parallel port – Led rojo.

En la figura 3.22 se muestra la configuración para el LED verde.

I/O Device: LEDs Colour: Green

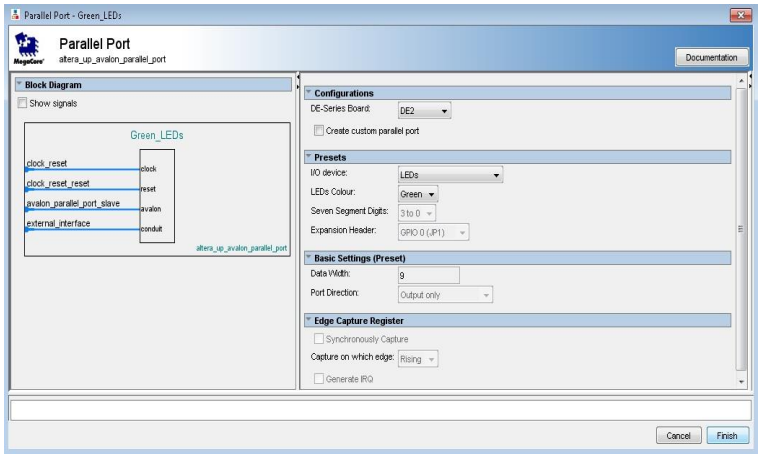


Figura 3.22 – Componente Parallel Port – Led verde.

En la figura 3.23 se muestra la configuración para el Pushbutton.

I/O Device: Pushbuttons

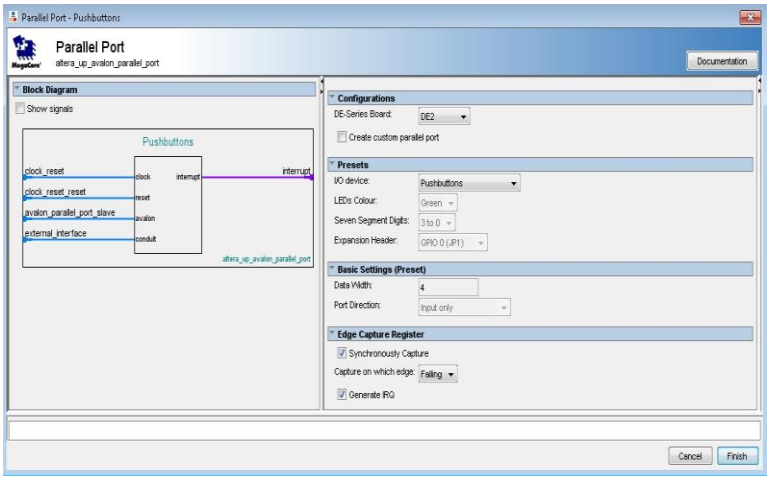


Figura 3.23 – Componente Parallel Port – Pushbutton.

3.3.2.2.4. PIO (PARALLEL I/O)

El PIO (Parallel IO) permite transferir datos ya sea de entrada, salida o ambos. La transferencia es hecha en paralelo y puede implicar de 1 a 32 bits. El número de bits y el sentido de transferencia son especificados por el usuario a través de Qsys cuando el módulo es creado.

En la figura 3.24 se muestra la configuración para el PIO que manejará las señales de entrada en el teclado matricial.

Número de bits: 4 Dirección: Entrada

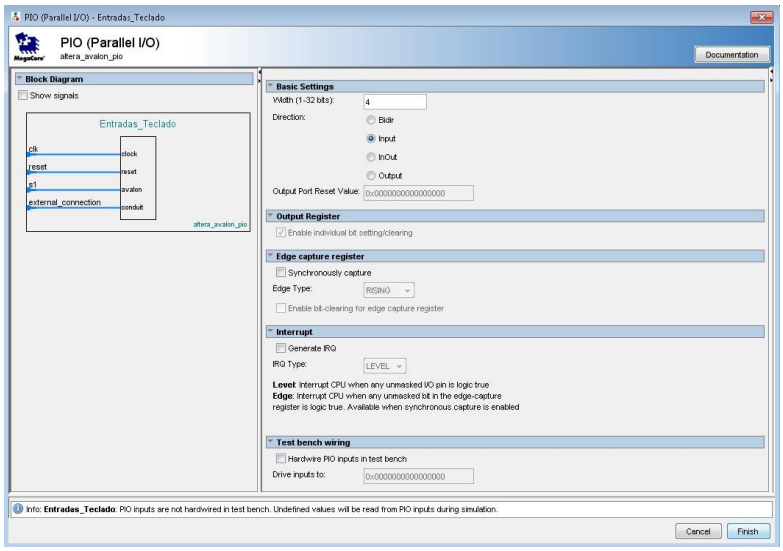


Figura 3.24 – Componente PIO – entradas del teclado.

En la figura 3.25 se muestra la configuración para el PIO que manejará las señales de salida del teclado matricial.

Número de bits: 4 Dirección: Salida

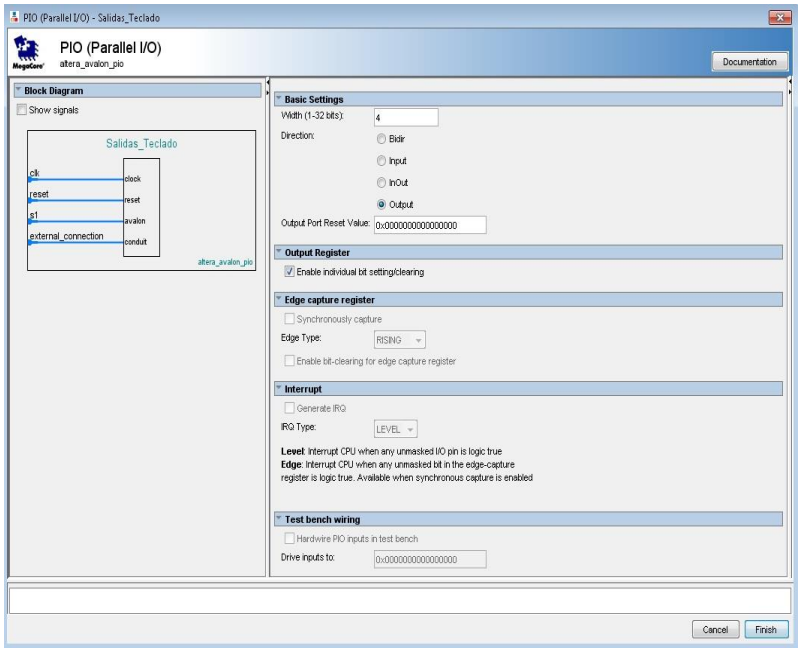


Figura 3.25 – Componente PIO – salidas del teclado.

En el sistema necesitamos incluir 2 PIO de 4 bits, uno para el manejo de las señales de entrada y otro para el manejo de las señales de salida del teclado.

3.3.2.3. LISTA DE COMPONENTES

Una vez que hemos añadido todas las componentes al sistema es necesario hacer la interconexión entre cada uno

de ellos. Las direcciones de memoria son asignadas automáticamente para cada componente.

La figura 3.26 muestra todas las componentes del sistema.

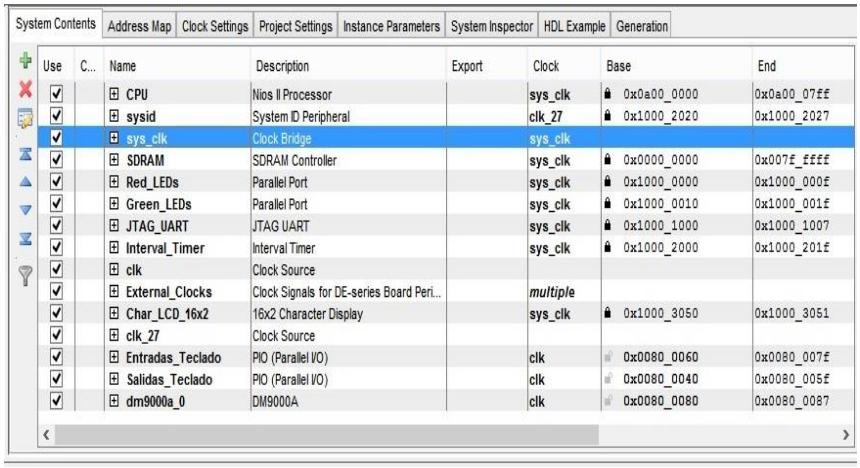


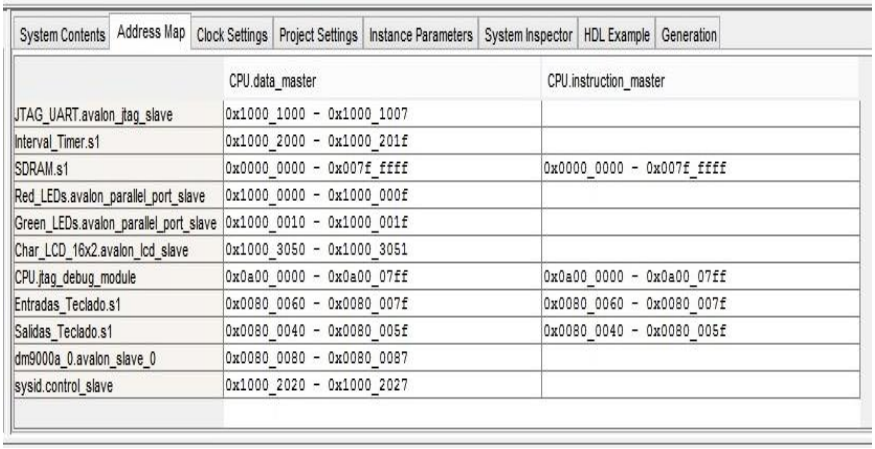
Figura 3.26 – Lista de componentes del sistema en QSYS.

Es importante mencionar que Qsys elige el nombre de los componentes por default, sin embargo los nombres no son tan descriptivos para identificarlos con los elementos del proyecto, pero se pueden cambiar dando clic derecho en el componente y seleccionando Rename.

3.3.2.4. MAPEO DE DIRECCIONES

Una vez que terminamos de añadir todas las componentes, necesitamos asignar las direcciones base en el sistema. Para esto seleccionamos en la barra de menú de Qsys System > Assign base addresses para que Qsys asigne automáticamente las direcciones bases del sistema.

La figura 3.27 muestra el mapeo de direcciones base del hardware.



	CPU.data_master	CPU.instruction_master
JTAG_UART.avalon_jtag_slave	0x1000_1000 - 0x1000_1007	
Interval_Timer.s1	0x1000_2000 - 0x1000_201f	
SDRAM.s1	0x0000_0000 - 0x007f_ffff	0x0000_0000 - 0x007f_ffff
Red_LEDs.avalon_parallel_port_slave	0x1000_0000 - 0x1000_000f	
Green_LEDs.avalon_parallel_port_slave	0x1000_0010 - 0x1000_001f	
Char_LCD_16x2.avalon_lcd_slave	0x1000_3050 - 0x1000_3051	
CPU.jtag_debug_module	0x0a00_0000 - 0x0a00_07ff	0x0a00_0000 - 0x0a00_07ff
Entradas_Teclado.s1	0x0080_0060 - 0x0080_007f	0x0080_0060 - 0x0080_007f
Salidas_Teclado.s1	0x0080_0040 - 0x0080_005f	0x0080_0040 - 0x0080_005f
dm9000a_0.avalon_slave_0	0x0080_0080 - 0x0080_0087	
sysid.control_slave	0x1000_2020 - 0x1000_2027	

Figura 3.27 – Mapa de direcciones base del sistema.

3.3.2.5. GENERACIÓN DEL SISTEMA EN QSYS

Después de haber especificado todos los componentes que se requieren, se puede proceder a generar el sistema.

Como se muestra en la figura 3.28, nos posicionamos en la pestaña Generation, en la cual seleccionamos que no cree un modelo de simulación, crear el archivo HDL para el diseño y el archivo de símbolo de bloque (.bsf); Después de esto damos clic en Generate.

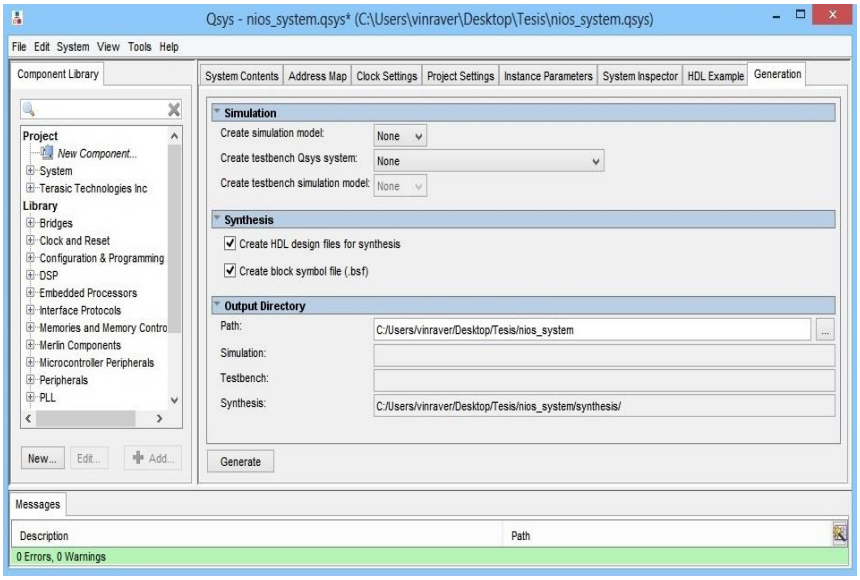


Figura 3.28 – Ventana para generar el sistema.

Si durante el proceso generación no se produce ningún error, se obtiene un mensaje como el mostrado en la figura 3.29, que indica que el sistema fue generado exitosamente.

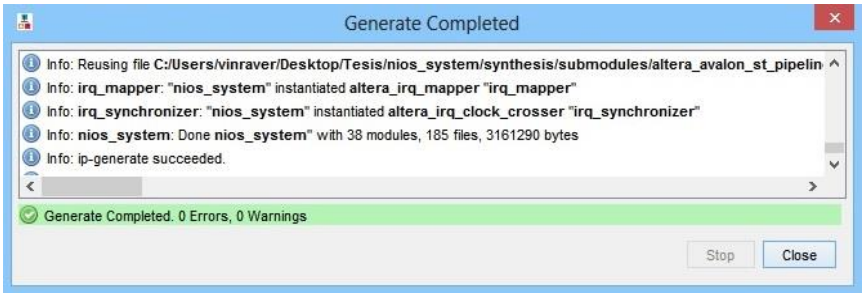


Figura 3.29 – Sistema generado exitosamente.

3.3.3. INTEGRACIÓN DEL SISTEMA DE QSYS EN QUARTUS II

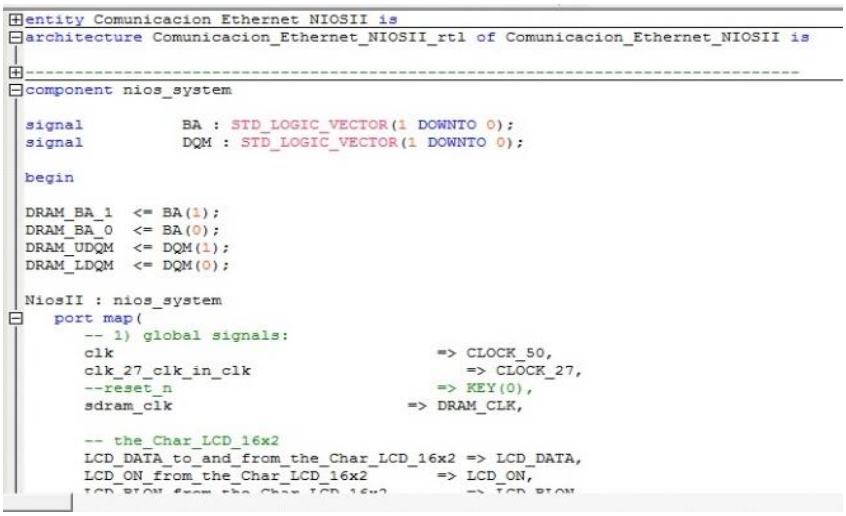
Luego de haber generado el hardware en Qsys, cerramos dicha aplicación y regresamos a Quartus II donde continuaremos con el desarrollo del proyecto.

3.3.3.1. INSTANCIACIÓN DEL SISTEMA

Cuando generamos el sistema en Qsys, una de las opciones seleccionadas fue crear el archivo HDL del sistema, el cual muestra las declaraciones de los componentes en VHDL.

Para instanciar el sistema creado en Qsys, copiamos y pegamos este código en nuestro archivo VHDL en Quartus como se muestra en la figura 3.30, para luego unir esta

componente con las señales internas en los puertos correspondientes.



```
entity Comunicacion_Ethernet_NIOSII is
architecture Comunicacion_Ethernet_NIOSII_rtl1 of Comunicacion_Ethernet_NIOSII is

--------------------------------------------------------------------------------
component nios_system

signal          BA : STD_LOGIC_VECTOR(1 DOWNTO 0);
signal          DQM : STD_LOGIC_VECTOR(1 DOWNTO 0);

begin

DRAM_BA_1  <= BA(1);
DRAM_BA_0  <= BA(0);
DRAM_UDQM  <= DQM(1);
DRAM_LDQM  <= DQM(0);

NiosII : nios_system
   port map(
      -- 1) global signals:
      clk                                    => CLOCK_50,
      clk_27_clk_in_clk                      => CLOCK_27,
      --reset_n                              => KEY(0),
      sdram_clk                            => DRAM_CLK,

      -- the_Char_LCD_16x2
      LCD_DATA_to_and_from_the_Char_LCD_16x2 => LCD_DATA,
      LCD_ON_from_the_Char_LCD_16x2          => LCD_ON,
```

Figura 3.30 – Instanciación del sistema en Quartus II.

Con esto, hemos añadido al diseño del sistema todos los componentes necesarios para lograr la funcionalidad de nuestro proyecto.

3.3.3.2. ASIGNACIÓN DE PINES DE LA FPGA

El siguiente paso es asignar los pines físicos de la tarjeta a las terminales de nuestro diseño. Esta tarea se la realiza después de compilar el proyecto realizado en Quartus.

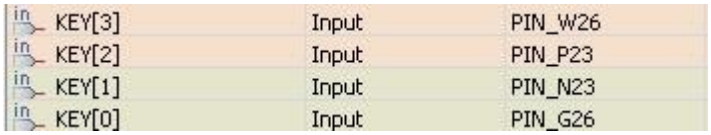
Processing > Start > Start Analysis and Elaboration

Si la compilación es exitosa, entonces podemos hacer la asignación de pines correspondientes tal como se muestra a partir de la figura 3.31 hasta la figura 3.39.

Assigments > Pin Planner

CLOCK_27	Input	PIN_D13
CLOCK_50	Input	PIN_N2

Figura 3.31 – Asignación de pines del reloj.



KEY[3]	Input	PIN_W26
KEY[2]	Input	PIN_P23
KEY[1]	Input	PIN_N23
KEY[0]	Input	PIN_G26

Figura 3.32 – Asignación de pines del push button.

UART_RXD	Input	PIN_C25
UART_TXD	Output	PIN_B25

Figura 3.33 – Asignación de pines del JTAG.

Entradas_Teclado[3]	Input	PIN_E25
Entradas_Teclado[2]	Input	PIN_E26
Entradas_Teclado[1]	Input	PIN_J22
Entradas_Teclado[0]	Input	PIN_D25
Salidas_Teclado[3]	Output	PIN_J20
Salidas_Teclado[2]	Output	PIN_J21
Salidas_Teclado[1]	Output	PIN_F23
Salidas_Teclado[0]	Output	PIN_F24

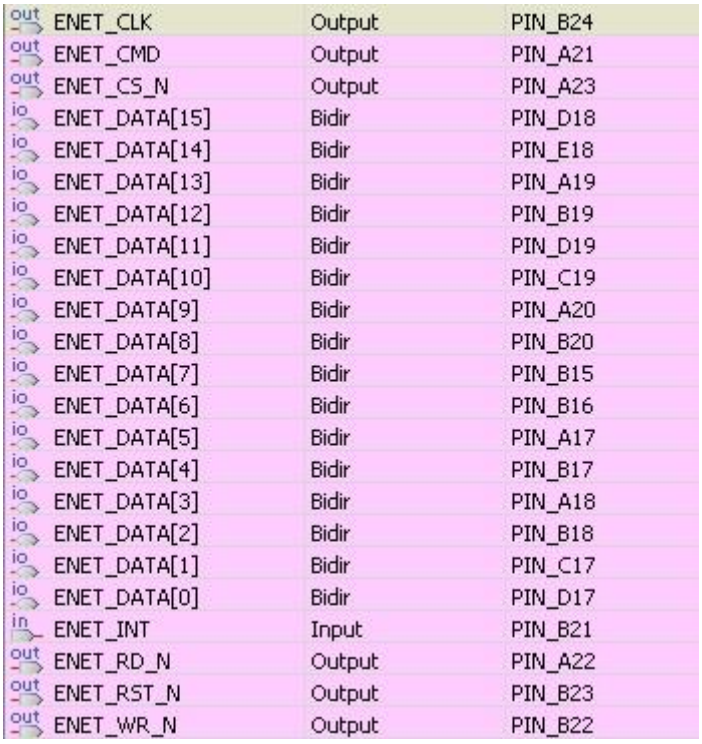
Figura 3.34 – Asignación de pines del teclado.

LCD_BLON	Output	PIN_K2
LCD_DATA[7]	Bidir	PIN_H3
LCD_DATA[6]	Bidir	PIN_H4
LCD_DATA[5]	Bidir	PIN_J3
LCD_DATA[4]	Bidir	PIN_J4
LCD_DATA[3]	Bidir	PIN_H2
LCD_DATA[2]	Bidir	PIN_H1
LCD_DATA[1]	Bidir	PIN_J2
LCD_DATA[0]	Bidir	PIN_J1
LCD_EN	Output	PIN_K3
LCD_ON	Output	PIN_L4
LCD_RS	Output	PIN_K1
LCD_RW	Output	PIN_K4

Figura 3.35 – Asignación de pines del LCD.

LEDG[8]	Output	PIN_Y12
LEDG[7]	Output	PIN_Y18
LEDG[6]	Output	PIN_AA20
LEDG[5]	Output	PIN_U17
LEDG[4]	Output	PIN_U18
LEDG[3]	Output	PIN_V18
LEDG[2]	Output	PIN_W19
LEDG[1]	Output	PIN_AF22
LEDG[0]	Output	PIN_AE22
LEDR[17]	Output	PIN_AD12
LEDR[16]	Output	PIN_AE12
LEDR[15]	Output	PIN_AE13
LEDR[14]	Output	PIN_AF13
LEDR[13]	Output	PIN_AE15
LEDR[12]	Output	PIN_AD15
LEDR[11]	Output	PIN_AC14
LEDR[10]	Output	PIN_AA13
LEDR[9]	Output	PIN_Y13
LEDR[8]	Output	PIN_AA14
LEDR[7]	Output	PIN_AC21
LEDR[6]	Output	PIN_AD21
LEDR[5]	Output	PIN_AD23
LEDR[4]	Output	PIN_AD22
LEDR[3]	Output	PIN_AC22
LEDR[2]	Output	PIN_AB21
LEDR[1]	Output	PIN_AF23
LEDR[0]	Output	PIN_AE23

Figura 3.36 – Asignación de pines de los Leds.



ENET_CLK	Output	PIN_B24
ENET_CMD	Output	PIN_A21
ENET_CS_N	Output	PIN_A23
ENET_DATA[15]	Bidir	PIN_D18
ENET_DATA[14]	Bidir	PIN_E18
ENET_DATA[13]	Bidir	PIN_A19
ENET_DATA[12]	Bidir	PIN_B19
ENET_DATA[11]	Bidir	PIN_D19
ENET_DATA[10]	Bidir	PIN_C19
ENET_DATA[9]	Bidir	PIN_A20
ENET_DATA[8]	Bidir	PIN_B20
ENET_DATA[7]	Bidir	PIN_B15
ENET_DATA[6]	Bidir	PIN_B16
ENET_DATA[5]	Bidir	PIN_A17
ENET_DATA[4]	Bidir	PIN_B17
ENET_DATA[3]	Bidir	PIN_A18
ENET_DATA[2]	Bidir	PIN_B18
ENET_DATA[1]	Bidir	PIN_C17
ENET_DATA[0]	Bidir	PIN_D17
ENET_INT	Input	PIN_B21
ENET_RD_N	Output	PIN_A22
ENET_RST_N	Output	PIN_B23
ENET_WR_N	Output	PIN_B22

Figura 3.37 – Asignación del DM9000A.

DRAM_ADDR[11]	Output	PIN_V5
DRAM_ADDR[10]	Output	PIN_Y1
DRAM_ADDR[9]	Output	PIN_W3
DRAM_ADDR[8]	Output	PIN_W4
DRAM_ADDR[7]	Output	PIN_U5
DRAM_ADDR[6]	Output	PIN_U7
DRAM_ADDR[5]	Output	PIN_U6
DRAM_ADDR[4]	Output	PIN_W1
DRAM_ADDR[3]	Output	PIN_W2
DRAM_ADDR[2]	Output	PIN_V3
DRAM_ADDR[1]	Output	PIN_V4
DRAM_ADDR[0]	Output	PIN_T6
DRAM_BA_0	Output	PIN_AE2
DRAM_BA_1	Output	PIN_AE3
DRAM_CAS_N	Output	PIN_AB3
DRAM_CKE	Output	PIN_AA6
DRAM_CLK	Output	PIN_AA7
DRAM_CS_N	Output	PIN_AC3
DRAM_DQ[15]	Bidir	PIN_AA5
DRAM_DQ[14]	Bidir	PIN_AC1
DRAM_DQ[13]	Bidir	PIN_AC2
DRAM_DQ[12]	Bidir	PIN_AA3
DRAM_DQ[11]	Bidir	PIN_AA4
DRAM_DQ[10]	Bidir	PIN_AB1
DRAM_DQ[9]	Bidir	PIN_AB2
DRAM_DQ[8]	Bidir	PIN_W6
DRAM_DQ[7]	Bidir	PIN_V7
DRAM_DQ[6]	Bidir	PIN_T8
DRAM_DQ[5]	Bidir	PIN_R8
DRAM_DQ[4]	Bidir	PIN_Y4
DRAM_DQ[3]	Bidir	PIN_Y3
DRAM_DQ[2]	Bidir	PIN_AA1
DRAM_DQ[1]	Bidir	PIN_AA2
DRAM_DQ[0]	Bidir	PIN_V6
DRAM_LDQM	Output	PIN_AD2
DRAM_RAS_N	Output	PIN_AB4
DRAM_UDQM	Output	PIN_Y5
DRAM_WE_N	Output	PIN_AD3

Figura 3.38 – Asignación de pines de la memoria RAM.

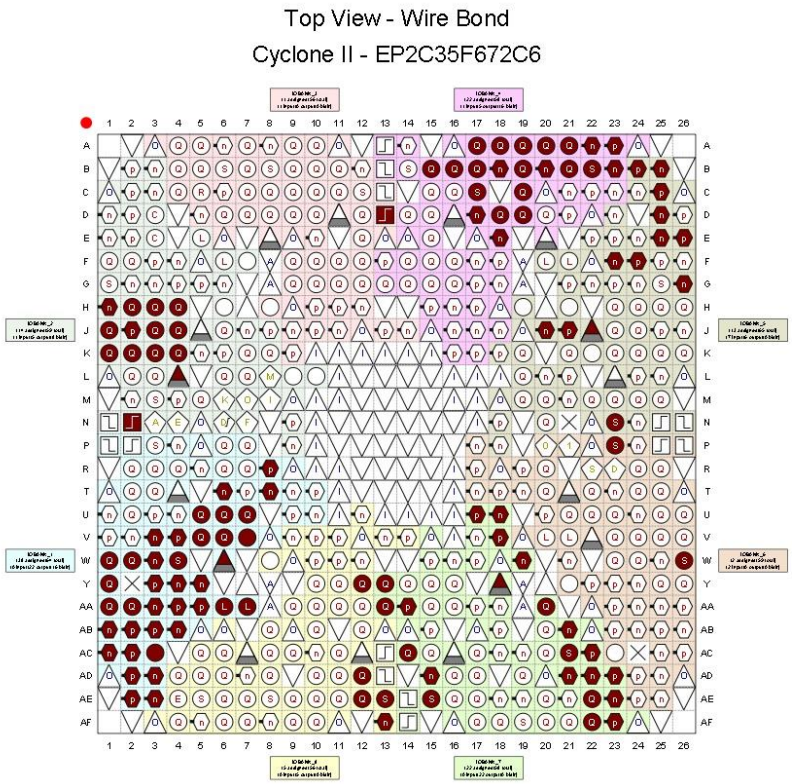


Figura 3.39 – Asignación de pines del sistema.

La correspondencia de pines de cada componente en la tarjeta DE2 la puede ser consultada en “DE2 Development and Education Booard – User Manual”.

3.3.3.3. COMPILACIÓN DEL PROYECTO

Por último, debemos realizar la compilación final del proyecto. Volvemos a hacer clic en:

Processing > Start Compilation.

Es importante que el proceso de compilación no presente errores, caso contrario no se genera el archivo .sof que es el que se utiliza para programar la tarjeta DE2.

Si la durante la compilación no se presenta ningún error, se muestra un mensaje como en la figura 3.40

Figura 3.40 – Compilación exitosa del proyecto.

3.3.4. PROGRAMACIÓN DEL SISTEMA EN LA TARJETADE2

Una vez que el proyecto ha sido compilado exitosamente, en Quartus son generados los archivos necesarios para la programación del FPGA.

Para realizar la programación damos click en:

Tools > Programmer.

Antes de presionar el botón Start debemos asegurarnos de que:

- Que en hardware setup se encuentre un USB Blaster válido.
- La extensión del archivo que va a ser cargado sea .sof.
- Si no se ha escogido por defecto, seleccionar JTAG en el cuadro mode.
- La tarjeta DE2 y la computadora estén conectadas mediante el USB Blaster y que el interruptor RUN/PROG de la tarjeta DE2 este en la posición RUN.

En la figura 3.41 se muestra la ventana para programar la FPGA.

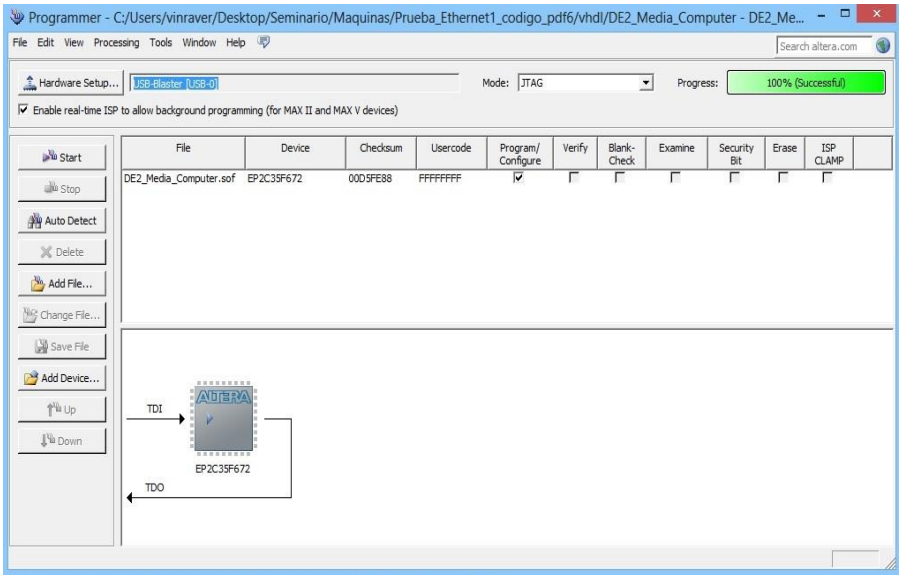


Figura 3.41 – Ventana para programar el FPGA.

Luego, presionamos el botón Start. Si se programa el FPGA correctamente la barra de progreso muestra 100%.

3.4. IMPLEMENTACIÓN DE LA APLICACIÓN EN LA TARJETADE2

Luego de configurar el hardware y que el chip FPGA es programado en Quartus procedemos a crear la lógica de la aplicación en lenguaje C.

3.4.1. CREACIÓN DE UN PROYECTO EN NIOS II SBT PARA ECLIPSE

El primer paso es abrir el IDE NIOS II y cambiar el workspace a la ubicación donde se encuentra el proyecto en Quartus tal como se observa en la figura 3.42.

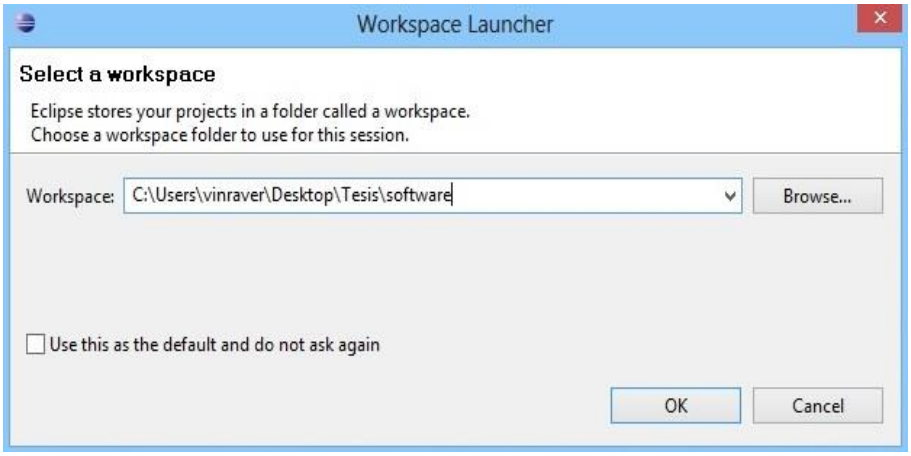


Figura 3.42 – Ventana para cambiar el workspace.

Después de esto en la ventana principal de trabajo de Eclipse creamos un nuevo proyecto.

File > New > Nios II Application and BSP from Template

Luego, se abre una ventana como se muestra en la figura 3.43, donde especificamos el archivo de descripción de sistema generado por Qsys (.sopcinfo) y el nombre del proyecto. Eclipse ofrece varias plantillas para la creación del proyecto, como nosotros vamos a crear el proyecto desde cero, elegimos la plantilla Blank Project.

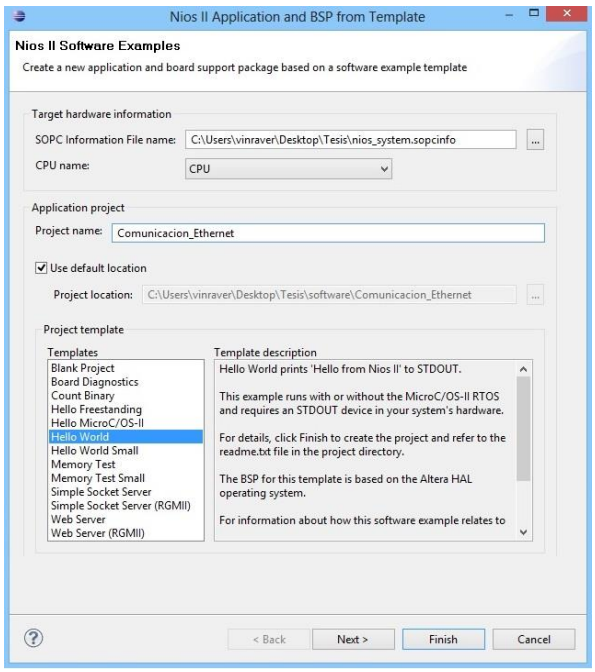


Figura 3.43 – Ventana para creación del proyecto.

En la siguiente ventana que se muestra en la figura 3.44, escogemos la siguiente opción:

Create a New BSP Project based on the application Project template

Con esta opción el asistente se encarga de dar información a Eclipse sobre el hardware que se encuentra disponible para su uso.

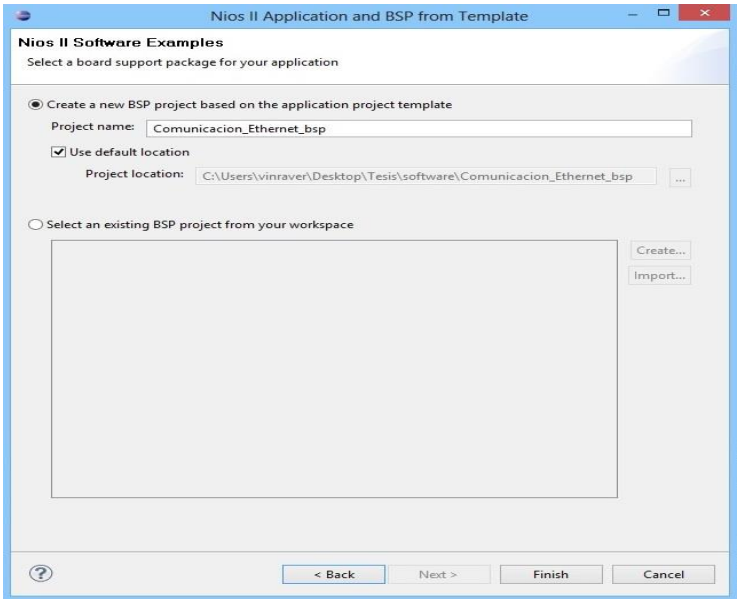


Figura 3.44 – Ventana para la creación de nuevo BSP.

Por último, damos clic en el botón finish para terminar con la creación de un nuevo proyecto en blanco en Eclipse.

3.4.2. CONSTRUCCIÓN DE LA APLICACIÓN EN LENJUAGUE C

Una vez terminada de crear una nueva aplicación en Eclipse se procede a escribir el código.

El código C principal y las funciones de la aplicación se encuentran en el anexo A.

3.4.2.1. LIBRERÍAS

Para empezar a desarrollar la aplicación escrita en lenguaje C, primero declaramos las librerías que son utilizadas a lo largo del programa.

Librería stdio.h.- “standard input – output header” (cabecera estándar entrada/salida), es la librería estándar del lenguaje de programación C que contiene las definiciones de macros, constantes, declaraciones de

funciones y la definición de tipos usados por varias operaciones estándar de entrada y salida.

Entre las funciones más importantes tenemos: printf, scanf y funciones de manipulación de ficheros.

Librería stdlib.h.- “standard library”, es el archivo de cabecera de la librería estándar de propósito general del lenguaje de programación C. Contiene los prototipos de funciones de C para gestión de memoria dinámica, control de procesos y otras. Las funciones de esta librería pueden ser agrupadas en 3 categorías básicas: Aritméticas, números aleatorios y conversión de cadenas.

Entre las funciones más relevantes están: atoi, atof, atol, rand, malloc, exit, abs, div, etc.

Librería unistd.h.- Esta librería proporciona acceso a la API del sistema operativo POSIX (Portable Operating System Interface Unix). Se conoce como estándar POSIX, a la base de la Single Unix Specification que debería estar disponible

en cualquier sistema operativo Unix como: Unix, GNU/Linux, Solaris, etc.

Debido a la compatibilidad de Unix con Cygwin y MinGW estas tienen sus propias versiones de unistd.h. De hecho estos sistemas proporcionan funciones en términos de Win32 con la ayuda de las bibliotecas de traducción.

Librería io.h.- "Input Output**"** (Entrada Salida), es un archivo de cabecera del lenguaje de programación C que es utilizado para operaciones de entrada/salida. El flujo de entrada y salida de datos en C no se encuentra definido dentro de la sintaxis básica, éste se provee por medio de librerías de funciones especializadas como io. En esta se definen los siguientes objetos:

- Flujo de Entrada y Salida.
- Flujo de error no almacenado.
- Flujo de error almacenado.

3.4.2.2. PUNTEROS

Los punteros implementados en el sistema fueron p_gpio_in, p_gpio_out, p_green_leds, p_red_leds.

Las direcciones de memoria a las que apuntan p_gpio_in y p_gpio_out son 0x00800060 y 0x00800040 respectivamente. Es en esta dirección de memoria es donde se guardan los datos que ingresan (columnas del teclado) al teclado matricial y los datos que se generan por la aplicación en C (filas del teclado). Los datos son leídos por la aplicación con la ayuda de los punteros para determinar que tecla fue presionada por el usuario y de esta manera obtener su id de usuario y su clave. La lógica usada para determinar que tecla fue presionada por el usuario se detalla más adelante.

Por último p_green_leds y p_red_leds que apuntan a las direcciones de memoria 0x10000010 y 0x10000000 respectivamente, son utilizadas para simular cuando la puerta es desbloqueada y bloqueada, dependiendo si la id y clave ingresada por el usuario es correcta.

Estas direcciones serán escritas durante el proceso de autenticación, si es un usuario autorizado se escribe un valor que hace que se encienda el Led correspondiente al color verde, si no lo es, se escribe el valor para prender el Led color rojo.

3.4.2.3. FUNCIONES BÁSICAS

A continuación se presenta una breve descripción de las funciones básicas en el desarrollo de la aplicación.

void LCD_cursor (int x, int y).- Mueve el cursor del LCD, a la posición indicada (x, y).

void LCD_text (char * texto).- Envía una cadena de texto al LCD.

void LCD_cursor_off (void).- Apaga el cursor del LCD.

void LCD_clear (void).- Limpia la pantalla del LCD.

char leer_tecla (void).- Lee la tecla presionada en el teclado matricial.

int EsNumero (char).- Devuelve un 1 si el carácter recibido es un número, y 0 si no lo es.

static unsigned short Calcular_Checksum (char* buffer).- Calcula el checksum del paquete recibido.

void Setear_Longitud_Paquete (unsigned int length).- Cambia la longitud del paquete por la longitud recibida.

void Setear_ID_Paquete (void).- Setea el ID del paquete.

void Setear_Checksum (char* buffer, unsigned int checksum).- Cambia el checksum del paquete por el checksum recibido.

char leer_tecla_tiempo (void).- Lee la tecla presionada en el teclado, además de ejecutar un proceso para cada cierto tiempo enviar una prueba de conexión con el servidor.

void ingreso_direccion_ip_tarjeta (void).- Permite ingresar la dirección IP de la tarjeta por medio del LCD usando el teclado.

void ingreso_direccion_ip_servidor (void).- Permite ingresar la dirección IP del servidor por medio del LCD usando el teclado.

void ingreso_usuario_clave (void).- Permite ingresar el id y la clave de un usuario por medio de la pantalla del LCD usando la función leer_tecla.

void generar_llave_nueva (void).- Genera un arreglo aleatorio de 3 números que servirá como llave para el cifrado de datos.

void cifrar_clave (void).- Genera un arreglo de 12 números que representa la clave cifrada del usuario, usando el método mencionado anteriormente.

void clave_cifrada_caracter (void).- Convierte el arreglo de números que representa la clave cifrada a un arreglo de caracteres.

void clave_cifrada_entero (void).- Convierte un arreglo de números binarios a enteros.

void desplazamiento_binario_entero (void).- Convierte el desplazamiento que se utiliza en el algoritmo de cifrado de la clave de binario a entero.

void entero_binario (void).- Transforma un número de entero a binario de 4 bits.

3.4.2.4. FUNCIONES PRINCIPALES

A continuación, se explican las funciones más importantes que se utilizan para manejar el chip DM9000A y realizar la comunicación ethernet.

3.4.2.4.1. INICIALIZACIÓN_DM9000

Para empezar a transmitir y recibir paquetes con el chip DM9000A se debe inicializarlo, configurando los registros NCR (Network Control Register), IMR (Interrupt Mask Register), NSR (Network Status

Register), los registros del chip PHY (Physical Layer), entre otros.

Para inicializar el DM9000 es necesario hacer lo siguiente:

Primero tenemos que encender el PHY interno, debido a que este viene apagado por defecto. Escribiendo “0” en la dirección 0x1F.

Luego, debemos programar el registro NCR, el valor que se escriba en este registro definirá el modo de operación de nuestro chip PHY. Estos modos pueden ser: modo loopback, modo full/half dúplex o forzar las colisiones. Para la aplicación, se escogió el modo loopback escribiendo 0x03 en el registro NCR (REG. 00).

Después, escribimos 0x81 en el registro IMR (REG. FF), para habilitar la función de auto retorno de puntero, que apunta a la dirección de memoria donde se guardarán los paquetes que se van a transmitir y los paquetes recibidos TX/RX FIFO SRAM.

A continuación, se debe guardar 6 bytes en la NIC, que representa la dirección MAC de la tarjeta, escribiendo en los registros de la dirección física (REG. 16 ~ 21).

Siguiendo con el proceso de inicialización, limpiamos los registros NSR (REG. 01) e ISR (REG. FE), estos registros representan el estado de red y el estado de las interrupciones. Además de programar otros registros como: RTFCR (0A), BPTR (REG. 08), FCTR (REG. 09), que representan el control de flujo de transmisión, recepción y el umbral para el control de flujo.

Por último, habilitamos las interrupciones RX/TX escribiendo en el registro IMR (REG. FF) y programamos el registro RCR (REG. 05) para habilitar la recepción de mensajes.

La figura 3.45 y 3.46 muestra el código para inicializar el DM9000A.

```
//------------------------------------------------------------------------
/* DM9000_init I/O routine */
unsigned int Inicializacion_DM9000 (void)  /* initialize DM9000 LAN chip */
{
  unsigned int  i;

  /* set the internal PHY power-on (GPIOs normal settings) */
  iow(0x1E, 0x01);  /* GPCR REG. 1EH = 1 selected GPIO0 "output" port
                       for internal PHY */
  iow(0x1F, 0x00);  /* GPR  REG. 1FH GEPIO0 Bit [0] = 0 to
                       activate internal PHY */
  msleep(5);        /* wait > 2 ms for PHY power-up ready */

  /* software-RESET NIC */
  iow(NCR, 0x03);   /* NCR REG. 00 RST Bit [0] = 1 reset on,
                       and LBK Bit [2:1] = 01b MAC loopback on */
  usleep(20);       /* wait > 10us for a software-RESET ok */
  iow(NCR, 0x00);   /* normalize */
  iow(NCR, 0x03);
  usleep(20);
  iow(NCR, 0x00);

  /* set GPIO0=1 then GPIO0=0 to turn off and on the internal PHY */
  iow(0x1F, 0x01);  /* GPR PHYPD Bit [0] = 1 turn-off PHY */
  iow(0x1F, 0x00);  /* PHYPD Bit [0] = 0 activate phyxcer */
  msleep(10);       /* wait >4 ms for PHY power-up */

  /* set PHY operation mode */
  phy_write(0,PHY_reset);   /* reset PHY: registers back to the default
                               states */
  usleep(50);               /* wait >30 us for PHY software-RESET ok */
  phy_write(16, 0x404);     /* turn off PHY reduce-power-down mode only */
  phy_write(4, PHY_txab);   /* set PHY TX advertised ability:
                               ALL + Flow_control */
  phy_write(0, 0x1200);     /* PHY auto-NEGO re-start enable
                           (RESTART_AUTO_NEGOTIATION + AUTO_NEGOTIATION_ENABLE)
                             to auto sense and recovery PHY registers */
  msleep(5);               /* wait >2 ms for PHY auto-sense linking to partner*/

  /* store MAC address into NIC */
  for (i = 0; i < 6; i++)
  iow(16 + i, ether_addr[i]);

  /* clear any pending interrupt */
  iow(ISR, 0x3F);     /* clear the ISR status: PRS, PTS, ROS, ROOS 4 bits,
                         by RW/C1 */
  iow(NSR, 0x2C);     /* clear the TX status: TX1END, TX2END, WAKEUP 3 bits,
                         by RW/C1 */
```

Figura 3.45 – Segmento 1 – Inicialización_DM9000 [34].

```
    /* program operating registers~ */
    iow(NCR,  NCR_set);   /* NCR REG. 00 enable the chip functions
                           (and disable this MAC loopback mode back to normal) */
    iow(0x08, BPTR_set);  /* BPTR  REG.08  (if necessary) RX Back Pressure
                    Threshold in Half duplex moe only: High Water 3KB, 600 us */
    iow(0x09, FCTR_set);  /* FCTR  REG.09  (if necessary) Flow Control Threshold
                           setting High/ Low Water Overflow 5KB/ 10KB */
    iow(0x0A, RTFCR_set); /* RTFCR REG.0AH (if necessary) RX/TX Flow Control
                           Register enable TXPEN, BKPM (TX_Half), FLCE (RX) */
    iow(0x0F, 0x00);      /* Clear the all Event */
    //printf("0x%.2X",ior(0x0F));
    iow(0x2D, 0x80);      /* Switch LED to mode 1 */

    /* set other registers depending on applications */
    iow(ETXCSR, ETXCSR_set); /* Early Transmit 75% */

    /* enable interrupts to activate DM9000 ~on */
    iow(IMR, INTR_set);   /* IMR REG. FFH PAR=1 only,
                           or + PTM=1& PRM=1 enable RxTx interrupts */

    /* enable RX (Broadcast/ ALL_MULTICAST) ~go */
    iow(RCR , RCR_set | RX_ENABLE | PASS_MULTICAST);  /* RCR REG. 05
                          RXEN Bit [0] = 1 to enable the RX machine/ filter */

    //printf("0x%.2X",ior(0x2D));
    /* RETURN "DEVICE_SUCCESS" back to upper layer */
    return  (ior(0x2D)==0x80) ? DMFE_SUCCESS : DMFE_FAIL;
}
```

Figura 3.46 – Segmento 2 – Inicialización_DM9000 [34].

3.4.2.4.2. PRUEBA_ENLACE_DM9000

Este método es el encargado de verificar que existe una conexión física entre la tarjeta DE2 y la computadora servidor, usando el registro de estatus de red (NSR) del chip DM9000A.

En la figura 3.47 se muestra el código que se uso en la función prueba_enlace_DM900.

```
int prueba_enlace_DM9000()
{
    // read network status bit
    iow( NSR, 0xff );
    IOWR(DM9000A_BASE,IO_addr,NSR);
     usleep(STD_DELAY);
     unsigned char tmp = IORD(DM9000A_BASE,IO_data);
    // link status bit
    return !!(tmp & (1<<6));
}
```

Figura 3.47 – Función prueba_enlace_DM9000 [34].

Las primeras 4 líneas de código leen el bit que indica el estado de red y lo guardan en la variable tipo char tmp.

Se realiza una operacion “and” bit a bit a la variable tmp para determinar el bit que indica el estado del enlace, este valor es retornado por la función prueba_enlace_DM9000.

Esta función se ejecutará en el código principal hasta que el valor retornado indique que existe una conexión física válida.

3.4.2.4.3. ENVIAR_MENSAJE

Esta función se encarga de armar el paquete, seteando campos como: ID, longitud del mensaje, el checksum para la cabecera IP, entre otros que pueden ir variando dependiendo del paquete UDP que se va a transmitir.

Además, recibe como parámetro un arreglo de caracteres que será el payload del paquete a transmitir.

La función Enviar_Mensaje utiliza la función Transmitir_Paquete para enviar los mensajes, esta se describe a continuación en la sección 3.4.2.4.4.

La función Enviar_Mensaje se muestra en la figura 3.48.

```
// ENVIAR MENSAJE ---------------------------------------------
void Enviar_Mensaje(char* message){
  /*This sends a UDP packet with the string message as the payload to
    the recipient at IPADDRESS */

  unsigned int longitud_mensaje = 0;
  unsigned int packet_checksum;

   while(message[longitud_mensaje] != '\0'){
    transmit_buffer[UDP_PACKET_PAYLOAD_OFFSET + longitud_mensaje]
    = message[longitud_mensaje];
    longitud_mensaje++;
   }
   transmit_buffer[UDP_PACKET_PAYLOAD_OFFSET + longitud_mensaje] = '\0';
   longitud_mensaje++;

      // Set Packet ID
    Setear_ID_Paquete();

    // Set Packet Length
    Setear_Longitud_Paquete(longitud_mensaje);

    // get checksum for ip header
    pointer_to_transmit_buffer = &transmit_buffer[0];
    packet_checksum = Calcular_Checksum(pointer_to_transmit_buffer);
    Setear_Checksum(pointer_to_transmit_buffer, packet_checksum);

     int j; /*printf("\n");*/ for (j = 0; j < 14 + 20 + 8 +
         longitud_mensaje; j++){ // 14 ethernet, 20 ip, 8 udp headers
     //printf("%x\t",transmit_buffer[j]);
     if ((j+1) % 6 == 0){
     //printf("\n");
     }
    }

    if (Transmitir_Paquete(transmit_buffer,
         /*message_length + 14 + 20 + 8*/0x4c)==DMFE_SUCCESS) {
        //printf("\nMensaje enviado exitosamente");
    } else {
        //printf("\nOcurrio un falla enviando el mensaje\n");
    }
}
```

Figura 3.48 – Función Enviar_Mensaje [35].

3.4.2.4.4. TRANSMITIR_PAQUETE

La figura 3.49 muestra la función Transmitir_Paquete, esta se encarga del envío de datagramas usando el chip DM9000A de la tarjeta DE2.

```
//---------------------------------------------------------------------
/*  Transmit one Packet TX I/O routine  */
unsigned int  Transmitir_Paquete(unsigned char *data_ptr,unsigned int tx_len)
{
  unsigned int  i;

   /* mask NIC interrupts IMR: PAR only */
  iow(IMR, PAR_set);

  /* issue TX packet's length into TXPLH REG. FDH & TXPLL REG. FCH */
  iow(0xFD, (tx_len >> 8) & 0xFF);  /* TXPLH High_byte length */
  iow(0xFC, tx_len & 0xFF);         /* TXPLL Low_byte  length */

  /* wirte transmit data to chip SRAM */
  IOWR(DM9000A_BASE, IO_addr, MWCMD);  /*set MWCMD REG. F8H TX I/O port ready*/
  for (i = 0; i < tx_len; i += 2)
  {
    usleep(STD_DELAY);
    IOWR(DM9000A_BASE, IO_data, (data_ptr[i+1]<<8)|data_ptr[i] );
  }

  /* issue TX polling command activated */
  iow(TCR , TCR_set | TX_REQUEST);  /* TXCR Bit [0] TXREQ auto
                                       clear after TX completed */

  //printf("transmision");

  /* wait TX transmit done */
  //while(!(ior(NSR)&0x0C)){
     //iow(NSR,0x00);
     /*printf("%x",ior(NSR));
     usleep(2000000);*/
  //}
  usleep(STD_DELAY);

  //printf("confirmacion");

  /* clear the NSR Register */
  iow(NSR,0x00);

  /* re-enable NIC interrupts */
  iow(IMR, INTR_set);

  /* RETURN "TX_SUCCESS" to upper layer */
  return  DMFE_SUCCESS;
}
```

Figura 3.49 – Función Transmitir_Paquete [34].

Para transmitir un paquete, es necesario realizar lo siguiente:

El dato se debe guardar en la TX FIFO SRAM, el cual está en la dirección 0 ~ 0xBFF de la SRAM interna del DM9000A.

Setear el registro MWCDM REG.F8 para que el puerto de entrada/ salida esté listo para transmitir y deshabilitar las interrupciones que se utilizan para la recepción de paquetes.

Luego se debe colocar la longitud del paquete en los registros, REG. FDH para el byte superior, y en el registro REG. FCH para el byte inferior.

El paso final es setear el TXREQ (Transmit Request), escribiendo en el registro TCR REG.02, para transmitir el paquete.

Para asegurarnos que el paquete se transmitió correctamente, leemos el registro NSR REG.01, ya que una vez que se transmite el paquete este registro cambia de valor.

Al terminar el proceso de transmisión, limpiamos el registro NSR y habilitamos las interrupciones nuevamente.

3.4.2.4.5. RECIBIR_PAQUETE

Cuando la tarjeta DE2 recibe paquetes ocurre lo siguiente:

Los paquetes recibidos son guardados en la RX FIFO, el cual está la dirección 0x0C00 ~ 0x3FFF (13Kbyte) de la SRAM interna del DM9000A. Hay 4 bytes para la cabecera MAC de cada paquete guardado en la RX FIFO SRAM.

El primer byte es usado para determinar si un paquete es recibido por la RX SRAM. Si es “01” significa que el paquete fue recibido en la RX SRAM y si es “00” significa que no se ha recibido un paquete. Antes de leer los otros bytes debemos asegurarnos que este byte sea 01.

El segundo byte guarda el “status” del paquete recibido. De acuerdo al valor de este byte, el paquete puede ser clasificado como erróneo o correcto.

El tercer y cuarto byte representa la longitud del paquete recibido. Los otros bytes son el dato del paquete recibido.

Usando los registros MRCMDX Y MRCMD podemos obtener la información del paquete entrante.

La función Recibir_Paquete se muestra en las figura 3.50 y 3.51.

```
//---------------------------------------------------------------------------
/* Receive One Packet I/O routine */
unsigned int  Recibir_Paquete (unsigned char *data_ptr,unsigned int *rx_len)
{
  unsigned char rx_READY,GoodPacket;
  unsigned int  Tmp, RxStatus, i;

  RxStatus = rx_len[0] = 0;
  GoodPacket=FALSE;

   /* mask NIC interrupts IMR: PAR only */
  iow(IMR, PAR_set);

  /* dummy read a byte from MRCMDX REG. FOH */
  rx_READY = ior(MRCMDX);

  /* got most updated byte: rx_READY */
  rx_READY = /*0x01;*/IORD(DM9000A_BASE,IO_data)&0x03;
  usleep(STD_DELAY);

  /* check if (rx_READY == 0x01): Received Packet READY? */
  if (rx_READY == DM9000_PKT_READY)
  {
     //printf("ready");
    /* got RX_Status & RX_Length from RX SRAM */
    IOWR(DM9000A_BASE, IO_addr, MRCMD); /* set MRCMD REG. F2H RX I/O
                                           port ready */
    usleep(STD_DELAY);
    RxStatus = IORD(DM9000A_BASE,IO_data);
    usleep(STD_DELAY);
    rx_len[0] = IORD(DM9000A_BASE,IO_data);

    /* Check this packet_status GOOD or BAD? */
    if ( !(RxStatus & 0xBF00) && (rx_len[0] < MAX_PACKET_SIZE) )
    {
      /* read 1 received packet from RX SRAM into RX buffer */
      for (i = 0; i < rx_len[0]; i += 2)
      {
        usleep(STD_DELAY);
        Tmp = IORD(DM9000A_BASE, IO_data);
        data_ptr[i] = Tmp&0xFF;
        data_ptr[i+1] = (Tmp>>8)&0xFF;
      }
      GoodPacket=TRUE;
      //printf("good");
    } /* end if (GoodPacket) */
```

Figura 3.50 – Segmento 1 – Recibir_Paquete [34].

```
    else
    {
       //printf("bad");
      /* this packet is bad, dump it from RX SRAM */
      for (i = 0; i < rx_len[0]; i += 2)
      {
        usleep(STD_DELAY);
        Tmp = IORD(DM9000A_BASE, IO_data);
      }
      //printf("\nError\n");
      rx_len[0] = 0;
    } /* end if (!GoodPacket) */
  } /* end if (rx_READY == DM9000_PKT_READY) ok */
  else if(rx_READY) /* status check first byte: rx_READY Bit[1:0]
                       must be "00"b or "01"b */
  {
     //printf(" no ready ");
    /* software-RESET NIC */
    iow(NCR, 0x03);   /* NCR REG. 00 RST Bit [0] = 1 reset on, and
                         LBK Bit [2:1] = 01b MAC loopback on */
    usleep(20);       /* wait > 10us for a software-RESET ok */
    iow(NCR, 0x00);   /* normalize */
    iow(NCR, 0x03);
    usleep(20);
    iow(NCR, 0x00);
    /* program operating registers~ */
    iow(NCR,  NCR_set);   /* NCR REG. 00 enable the chip functions
                          (and disable this MAC loopback mode back to normal) */
    iow(0x08, BPTR_set);  /* BPTR  REG.08  (if necessary) RX Back Pressure
                  Threshold in Half duplex moe only: High Water 3KB, 600 us */
    iow(0x09, FCTR_set);  /* FCTR  REG.09  (if necessary) Flow Control
                         Threshold setting High/ Low Water Overflow 5KB/ 10KB */
    iow(0x0A, RTFCR_set); /* RTFCR REG.0AH (if necessary) RX/TX Flow Control
                           Register enable TXPEN, BKPM (TX_Half), FLCE (RX) */
    iow(0x0F, 0x00);      /* Clear the all Event */
    iow(0x2D, 0x80);      /* Switch LED to mode 1 */
    /* set other registers depending on applications */
    iow(ETXCSR, ETXCSR_set); /* Early Transmit 75% */
    /* enable interrupts to activate DM9000 ~on */
    iow(IMR, INTR_set);   /* IMR REG. FFH PAR=1 only, or + PTM=1& PRM=1
                             enable RxTx interrupts */
    /* enable RX (Broadcast/ ALL_MULTICAST) ~go */
    iow(RCR , RCR_set | RX_ENABLE | PASS_MULTICAST);  /* RCR REG. 05
                          RXEN Bit [0] = 1 to enable the RX machine/ filter */
  } /* end NIC H/W system Data-Bus error */

  return GoodPacket ? DMFE_SUCCESS : DMFE_FAIL;
}
```

Figura 3.51 – Segmento 2 – Recibir_Paquete [34].

Para saber si hemos recibido un paquete, primero leemos el registro MRCMDX (REG. F0), si este es “01” significa que el paquete recibido esta listo para ser leído. Luego debemos leer el registro MRCMD (REG. F2), con la ayuda del puntero read RX, este puntero será incrementado después de leer el registro MRCMD, y de esa manera saber el estatus del paquete, longitud y dato del paquete almacenado en la SRAM.

3.4.2.4.6. ETHERNET_INTERRUPT_HANDLER

La función ethernet_interrupt_handler es la que se encarga de verificar si el tipo de paquete recibido es correcto, lee el paquete y cambia los estados del sistema dependiendo del contenido del mensaje.

Esta función hace uso de Recibir_Paquete para almacenar el paquete recibido en el puntero receive_buffer y poder verificar que el estatus del paquete sea correcto, es decir, la longitud sea mayor

a 14 para que sea considerado un paquete Ethernet, que sea un paquete IP y que el protocolo usado en la capa de transporte sea UDP. Si se cumplen estas consideraciones, leemos el dato del paquete y cambiamos la variable estado_sistema. Esta función se muestra en las figuras 3.52 y 3.53.

```
static void ethernet_interrupt_handler() {
  unsigned int receive_status;
  if(imprimir_consola==1){
     receive_buffer_length=0;
     receive_status = Recibir_Paquete(receive_buffer, &receive_buffer_length);
  if (receive_status == DMFE_SUCCESS) {
    if (receive_buffer_length >= 14) {
       // Es un Paquete Ethernet
      if ((receive_buffer[12] == 8 && receive_buffer[13] == 0 &&receive_buffer_length >= 34)) {
         // Es un paquete IP
         if ((receive_buffer[23] == 0x11)){
            // Es un paquete UDP
            if (receive_buffer_length >= UDP_PACKET_PAYLOAD_OFFSET) {
               receivedMessageFlag = 1;
               char* message_received = receive_buffer + UDP_PACKET_PAYLOAD_OFFSET;
               mensaje=message_received;
               if(strcmp(message_received,"autorizado")==0){
                 estado_sistema=1;
                  printf("\n  Mensaje AUTORIZADO recibida desde el servidor\n");
            }else if(strcmp(mensaje,"denegado")==0){
                 estado_sistema=2;
                  printf("\n  Mensaje DENEGADO recibida desde el servidor\n");
            }else if(strcmp(mensaje,"bloquear")==0){
               estado_sistema=3;
                  printf("\n  Mensaje BLOQUEAR recibida desde el servidor\n");
            }else if(strcmp(mensaje,"desbloquear")==0){
               estado_sistema=4;
                  printf("\n  Mensaje DESBLOQUEAR recibida desde el servidor\n");
            }else if(strcmp(mensaje,"conexion")==0){
               estado_sistema=5;
                  printf("\n  Mensaje CONEXION recibida desde el servidor\n");
            }else if(strcmp(mensaje,"idincorrecto")==0){
               estado_sistema=6;
```

Figura 3.52 – Segmento 1 – Ethernet_interrupt_handler [35].

```
                printf("\n  Mensaje IDINCORRECTO recibida desde el servidor\n");
            }else if(strcmp(mensaje,"cambiado")==0){
               estado_sistema=7;
                printf("\n  Mensaje CAMBIADO recibida desde el servidor\n");
            }else if(strcmp(mensaje,"ack")==0){
                 estado_sistema=8;
                printf("\n  Mensaje ACK recibida desde el servidor\n");
               pointer_to_receive_buffer = &receive_buffer[0];
               receive_checksum = Calcular_Checksum(pointer_to_receive_buffer);
               if( ((receive_checksum >> 8) == (receive_buffer[IP_HEADER_CHECKSUM_OFFSET]))
                    && ((receive_checksum & 0x00ff) == (receive_buffer[IP_HEADER_CHECKSUM_OFFSET+1])) )
               {
                  //printf("\nChecksum OK");
               } else {
                  //printf("\nChecksum test failed. Message ignored.\n");
               }}} else {
              //printf("\nReceived non-UDP packet\n");
           }} else {
           Transmitir_Paquete(arp_buffer, 60);
        }} else {
        //printf("Malformed Ethernet packet\n");
      }} else {
      //printf("Error recibiendo paquete\n");
   }
   /* Limpiar el registro ISR del DM9000A ISR: PRS, PTS, ROS, ROOS 4   bits, by RW/C1 */
   iow(ISR, 0x3F);
   /* Re-Habilitar las interrupciones del DM9000A */
   iow(IMR, INTR_set);
}
   /* Limpiar el registro ISR del DM9000A ISR: PRS, PTS, ROS, ROOS 4   bits, by RW/C1 */
   iow(ISR, 0x3F);
   /* Re-Habilitar las interrupciones del DM9000A */
   iow(IMR, INTR_set);
}
```

Figura 3.53 – Segmento 2 – Ethernet_interrupt_handler [35].

3.4.2.5. PROGRAMA PRINCIPAL

En esta sección se explica en forma estructurada el código de la función principal, dividiéndola en los estados del sistema.

Los segmentos de código se encuentran al final de la explicación.

Verificación de enlace Ethernet

La verificación del enlace ethernet es lo primero que se verifica al iniciar el programa y se la realiza con la función Prueba_enlace_DM9000a.

El valor de la variable estado_conexión_ethernet cambia a verdadero solo cuando se conecta un cable ethernet a la tarjeta DE2 permitiendo continuar con la ejecución del código tal como se muestra en la figura 3.55.

Configuraciones de red

Una vez conectado la tarjeta a la red LAN, se pide al administrador que ingrese la dirección IP de la tarjeta DE2 y del servidor tal como se puede observar en la figura 3.56 a través de la función ingreso_direccion_ip_tarjeta e ingreso_direccion_ip_servidor.

Verificación de conexión

Después que se realiza el seteo de las configuraciones de red, la tarjeta DE2 realiza una prueba de conexión con el servidor tal como se observa en el segmento de código de la figura 3.56.

Usa la función Enviar_Mensaje para transmitir un paquete con el mensaje “TEST” para que le responda el servidor.

Al recibir un mensaje de respuesta, el valor de la variable cambia estado_sistema= 5, indicando que se ha establecido conexión, saliendo del bucle y continuando con el código del programa.

En cambio, si después de enviar dos veces el mensaje de prueba no recibe una respuesta, el sistema asumirá que las direcciones IP ingresadas son incorrectas, volviendo a pedir reingreso de las mismas.

Ingreso de datos

Antes que el sistema obtenga datos de los usuarios, el programa utiliza la función Enviar_Mensaje para transmitir un mensaje al servidor con la llave que utilizará para cifrar los datos.

Los datos son ingresados a través del teclado matricial usando la función Ingreso_usuario_clave tal como se muestra en la figura 3.58.

Dentro del código de esta función se utiliza otra para cifrar la clave.

Trasmisión de datos

Previo al envío de un paquete se reinicializa el DM9000 con la función inicializacion_DM9000. La razón de la reinicialización, se la explica en el capítulo 4 en la sección de pruebas de rendimiento del sistema.

Una vez hecho esto, se procede a enviar el paquete con los datos de los usuarios, usando la función Enviar_Mensaje tal como se observa en la figura 3.59.

Recepción de datos

La variable estado_sistema tiene el valor de 0 que indica que no se ha recibido en ese momento ningún mensaje desde la computadora.

Cuando la tarjeta DE2 recibe un mensaje se activa la interrupción Alt_irq_register que se encuentra en la sección de código de la figura 3.54 llamando a la función ethernet_interrupt_handler que leerá el contenido del mensaje.

Después que se obtiene el contenido del paquete recibido, el programa envía un mensaje ACK al servidor utilizando la función Enviar_Mensaje para confirmar que el proceso de autenticación se ha efectuado con éxito tal como se muestra en la figura 3.59.

Actuador

Una vez la función ethernet_interrupt_handler ya leyó el contenido del mensaje recibido y cambio el valor de la variable estado_sistema, el sistema procede a actuar dependiendo de su valor tal como se muestra en la figura 3.59 y 3.60, 3.61, 3.62 y 3.63.

Si estado_sistema=1, significa que fue un usuario autorizado, encendiendo el Led verde por 7 segundos que

simula puerta abierta. Luego vuelve al estado de ingreso de datos.

Si estado_sistema=2, significa que no esta autorizado, encendiendo el Led rojo por 3 segundos y volviendo al estado de ingreso de datos.

Si estado_sistema=3, significa que se debe bloquear el sistema, encendiendo el Led rojo indefinidamente y quedándose en el bucle de bloquear hasta recibir un mensaje de desbloqueo

Si estado_sistema=4, significa que se intenta desbloquear el sistema, apagando el Led rojo, saliendo del bucle y volviendo al estado de ingreso de datos.

Si estado_sistema=6, significa que se ha ingresado un ID erróneo, pidiendo reingreso de datos.

```
int main(void)
{
  volatile int *p_green_leds=green_leds, *p_red_leds=red_leds;
  int i=0, intentos=0, bloqueo=0, desbloqueo=0,bandera1=0,prueba1=0,
    repeticion_envio=0, estado_conexion_ethernet=0, contador_intentos_conexion=0, c=0;
  volatile int delay_count;
  char tecla='q';

  alt_irq_register(DM9000A_IRQ, NULL, (void*)ethernet_interrupt_handler);

  (*p_red_leds)=0x0;
  (*p_green_leds)=0x0;
  imprimir_consola=1;
  contador_intentos_conexion=0;

  printf("Configuracion Anterior:\n");
  printf("  Tarjeta DE2:\n");
  printf("    Direccion IP: %d",transmit_buffer[26]);
  printf(".%d",transmit_buffer[27]);
  printf(".%d",transmit_buffer[28]);
  printf(".%d\n",transmit_buffer[29]);
  printf("    Direccion MAC: %x",arp_buffer[6]);
  printf(".%x",arp_buffer[7]);
  printf(".%x",arp_buffer[8]);
  printf(".%x",arp_buffer[9]);
  printf(".%x",arp_buffer[10]);
  printf(".%x\n",arp_buffer[11]);
  printf("  Servidor:\n");
  printf("    Direccion IP: %d",transmit_buffer[30]);
  printf(".%d",transmit_buffer[31]);
  printf(".%d",transmit_buffer[32]);
  printf(".%d\n",transmit_buffer[33]);
  printf("    Direccion MAC: %x",arp_buffer[0]);
```

Figura 3.54 – Segmento 1 – Programa principal.

```
printf(".%x",arp_buffer[1]);
printf(".%x",arp_buffer[2]);
printf(".%x",arp_buffer[3]);
printf(".%x",arp_buffer[4]);
printf(".%x\n",arp_buffer[5]);
printf("  Puerto Origen/destino: 5005\n");
printf("\nIniciando conexion con el servidor...");

while(estado_sistema!=5&&contador_intentos_conexion<2){
  estado_sistema=0;
  LCD_clear();
  LCD_cursor (0,0);
  LCD_text ("  Verificando ");
  LCD_cursor (0,1);
  LCD_text ("   Conexion... ");
  LCD_cursor_off ();
  Inicializacion_DM9000();
  usleep(2000000);

  if (!prueba_enlace_dm9000a()){
      LCD_clear();
      LCD_cursor (0,0);
      LCD_text ("  Cable de red ");
      LCD_cursor (0,1);
      LCD_text ("  Desconectado ");
      LCD_cursor_off ();
      printf("\n  Problema: Cable de red desconectado");
      estado_conexion_ethernet=0;

      while(estado_conexion_ethernet==0){
          usleep(2000000);
          if (prueba_enlace_dm9000a()){
              estado_conexion_ethernet=1;
           }
```

Figura 3.55 – Segmento 2 – Programa principal.

```
      Inicializacion_DM9000();
   }
      usleep(2000000);
}
printf("\n  Mensaje TEST enviado al servidor");
Enviar_Mensaje("test");
usleep(1000000);

if(estado_sistema==5){
    printf("  Mensaje ACK enviado al servidor\n");
    LCD_clear();
    LCD_cursor (0,0);
    LCD_text ("    Conexion    ");
    LCD_cursor (0,1);
    LCD_text ("   Establecida  ");
    LCD_cursor_off ();
    Enviar_Mensaje("establecida");
    printf("  Tarjeta DE2 conectada con servidor\n");

 }else{
    if(contador_intentos_conexion==0){
    printf("\n  Error: Tiempo de espera para recibir una "
           "respuesta del servidor agotada (1 segundo)");
    }

    if(contador_intentos_conexion==1){
    printf("\n  Problema: Servidor inalcanzable\n");
    LCD_clear();
    LCD_cursor (0,0);
    LCD_text ("   Configure    ");
    LCD_cursor (0,1);
    LCD_text (" Direcciones IP ");
    LCD_cursor_off ();
```

Figura 3.56 – Segmento 3 – Programa principal.

```
            printf("\nConfiguracion de red:\n");
            usleep(2000000);
            printf("  Direccion IP Tarjeta DE2: ");
            ingreso_direccion_ip_tarjeta();
            printf("  Direccion IP Servidor:     ");
            ingreso_direccion_ip_servidor();
            contador_intentos_conexion=-1;
            }
          }
        contador_intentos_conexion++;
 }

contador_verificacion_conexion=0;
while(1){
  estado_conexion_ethernet=0;
  salir=1;
  mensaje_enviar[9]="qqqqqqqqq";
  clave[6]="qqqqqq";
  usuario[3]="qqq";
  i=0;

  estado_sistema=0;
  if(saltodes==0){
  generar_llave_nueva();
  }

  estado_sistema=0;
  repeticion_envio_llave=0;
  while(estado_sistema==0 && repeticion_envio_llave<2 && saltodes==0){
      Inicializacion_DM9000();
      usleep(2000000);
      Enviar_Mensaje(llave_actual);
      printf("\n  Mensaje %s enviado al servidor",llave_actual);
      usleep(1000000);
```

Figura 3.57 – Segmento 4 – Programa principal.

```
    if(estado_sistema==8){
        printf("  Llave enviada correctamente al servidor\n");
        printf("\nIngreso de datos:\n");
     }else{
        printf("\n  Error: Tiempo de espera para recibir una "
               "respuesta del servidor agotada (1 segundo)");
         }

    repeticion_envio_llave++;
    if(repeticion_envio_llave==2 && estado_sistema==0){
    printf("\n\nProblema: Servidor sin llave de cifrado actual\n");
    printf("\nIngreso de datos:\n");
    }
 }

(*p_red_leds)=0x0;
(*p_green_leds)=0x0;
while(salir==1){
    ingreso_usuario_clave();
}

estado_sistema=0;
repeticion_envio=0;
while(estado_sistema==0 && repeticion_envio<2){

    if(repeticion_envio==1){
        printf("\n  Error: Tiempo de espera para recibir una respuesta "
               "del servidor agotada (1 segundo)");
        LCD_cursor (0,0);
        LCD_text (" Autenticando...");
        LCD_cursor (0,1);
        LCD_text ("                ");
        LCD_cursor_off ();
```

Figura 3.58 – Segmento 5 – Programa principal.

```
        }
        Inicializacion_DM9000();
        usleep(2000000);
        Enviar_Mensaje(mensaje_enviar);
        printf("\n  Mensaje ");
        for (c=0; c<16; c++){
        printf("%c",mensaje_enviar[c]);
        }
        printf(" enviado al servidor ");
        usleep(1000000);
        repeticion_envio++;
    }

    if(estado_sistema!=0){
        Enviar_Mensaje("ack");
        printf("  Mensaje ACK enviado al servidor\n");

        if(estado_sistema==1){
            (*p_green_leds)=0x01;
            printf("  Puerta desblogueada\n");
            LCD_clear();
            LCD_cursor (0,0);
            LCD_text ("    Usuario     ");
            LCD_cursor (0,1);
            LCD_text ("  Autorizado   ");
            LCD_cursor_off ();
            usleep(2000000);

            LCD_clear();
            LCD_cursor (0,0);
            LCD_text ("     Puerta      ");
            LCD_cursor (0,1);
            LCD_text ("  Desblogueada   ");
            LCD_cursor_off ();
```

Figura 3.59 – Segmento 6 – Programa principal.

```
}else if(estado_sistema==2){
    (*p_red_leds)=0x20000;
    printf("  Puerta bloqueada\n");
    LCD_clear();
    LCD_cursor (0,0);
    LCD_text ("     Clave     ");
    LCD_cursor (0,1);
    LCD_text ("   Incorrecta  ");
    LCD_cursor_off ();
    usleep(2000000);
   }
else if(estado_sistema==3){
    printf("\nTarjeta DE2 blogueada...");

    while(estado_sistema!=4){
        (*p_red_leds)=0x20000;
        LCD_clear();
        LCD_cursor (0,0);
        LCD_text ("    OFICINA  ");
        LCD_cursor (0,1);
        LCD_text ("   BLOQUEADA  ");
        LCD_cursor_off ();
        estado_sistema=0;
        Inicializacion_DM9000();
        usleep(2000000);

        if (!prueba_enlace_dm9000a()){
            LCD_clear();
            LCD_cursor (0,0);
            LCD_text ("  Cable de red ");
            LCD_cursor (0,1);
            LCD_text ("  Desconectado ");
            LCD_cursor_off ();
```

Figura 3.60 – Segmento 7 – Programa principal.

```
        prueba1=0;
        while(prueba1==0){
            usleep(2000000);
            if (prueba_enlace_dm9000a()){
                  prueba1=1;
            }
            Inicializacion_DM9000();
         }
    LCD_clear();
    LCD_cursor (0,0);
    LCD_text ("    OFICINA  ");
    LCD_cursor (0,1);
    LCD_text ("   BLOQUEADA  ");
    LCD_cursor_off ();
    usleep(2000000);
}

if(contador_verificacion_conexion1>=10){
    while(estado_sistema!=5 && estado_sistema!=4){

        if(bandera1>=1){
            Inicializacion_DM9000();
            usleep(2000000);
        }
        Enviar_Mensaje("test");
        printf("\n  Mensaje TEST enviado al servidor");
        usleep(1000000);

        if(estado_sistema==5){
            contador_verificacion_conexion1=0;
            Enviar_Mensaje("bloqueado");
            printf("  Mensaje BLOQUEADO enviado al servidor\n");
            bandera1=0;
        }
```

Figura 3.61 – Segmento 8 – Programa principal.

```
                if(estado_sistema==4){
                    contador_verificacion_conexion1=0;
                    bandera1=0;
                }

                if(bandera1>=1){
                    LCD_clear();
                    LCD_cursor (0,0);
                    LCD_text ("  Desconexion  ");
                    LCD_cursor (0,1);
                    LCD_text ("   Servidor    ");
                }
                bandera1++;
            }
            bandera1=0;
        }
        if(estado_sistema==0){
           usleep(1000000);
        }
        contador_verificacion_conexion1=contador_verificacion_conexion1+4;
    }
    LCD_clear();
    LCD_cursor (0,0);
    LCD_text ("    OFICINA  ");
    LCD_cursor (0,1);
    LCD_text (" DESBLOQUEADA ");
    Enviar_Mensaje("desbloqueado");
    printf("  Mensaje DESBLOQUEADO enviado al servidor");
    printf("\n  Tarjeta DE2 desbloqueada\n");
    usleep(2000000);
}
else if(estado_sistema==6){
    printf("\n  Problema: ID de usuario ingresado no es valido\n");
```

Figura 3.62 – Segmento 9 – Programa principal.

```
            LCD_clear();
            LCD_cursor (0,0);
            LCD_text ("   ID Usuario   ");
            LCD_cursor (0,1);
            LCD_text ("   Incorrecto   ");
            LCD_cursor_off ();
            usleep(2000000);
        }
      }else{
        printf("\n  Problema: Autenticacion no recibida del servidor");
        saltodes=1;
        contador_verificacion_conexion=16000000;
        }

}
return 0;
```

Figura 3.63 – Segmento 10 – Programa principal.

3.4.3. COMPILACIÓN Y EJECUCIÓN

Antes de ejecutar la aplicación con el icono que se observa en la figura 3.65, es necesario realizar el proceso de compilación para generar el archivo en código (.elf) y poderla ejecutar en la tarjeta.

El proceso de compilación como se muestra en la figura 3.64 efectúa un análisis léxico, sintáctico y semántico del código para verificar su validez. Para realizar esta tarea seleccionamos: Project > Build Project.

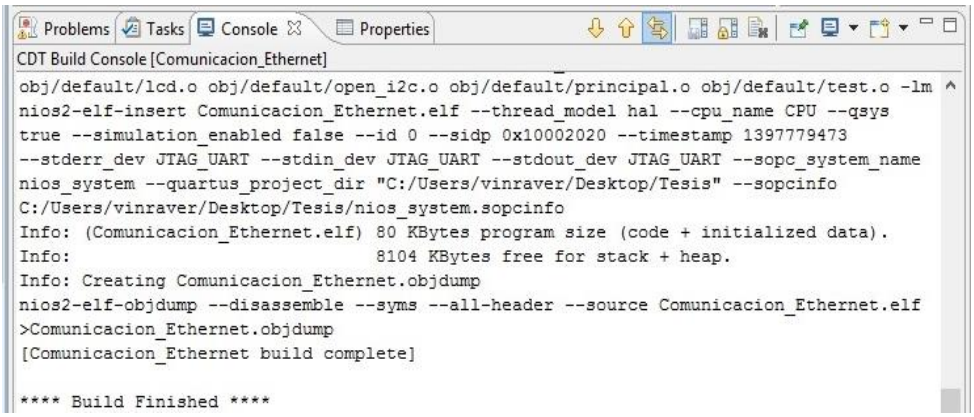


Figura 3.64 – Compilación exitosa de la aplicación.

Para ejecutar la aplicación debemos asegurarnos que el hardware realizado en Quartus II, se haya cargado al FPGA.

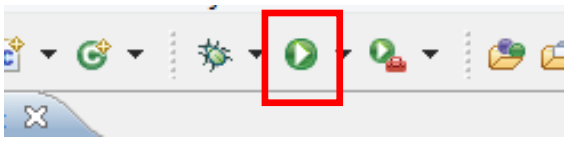


Figura 3.65 – Ejecutar la aplicación en la tarjeta DE2.

3.5. IMPLEMENTACIÓN DE LA APLICACIÓN EN LA PC

En esta sección, se muestra a breve rasgos el diseño de la base de datos y de la aplicación desarrollada en Java.

3.5.1. DESCRIPCIÓN BÁSICA DE LA BASE DE DATOS EN MYSQL

Se hace una explicación de cómo maneja la base de datos la información recibida por la aplicación.

3.5.1.1. MODELO LÓGICO

El modelo lógico de la figura 3.66 indica como se estructura la información en la base de datos.

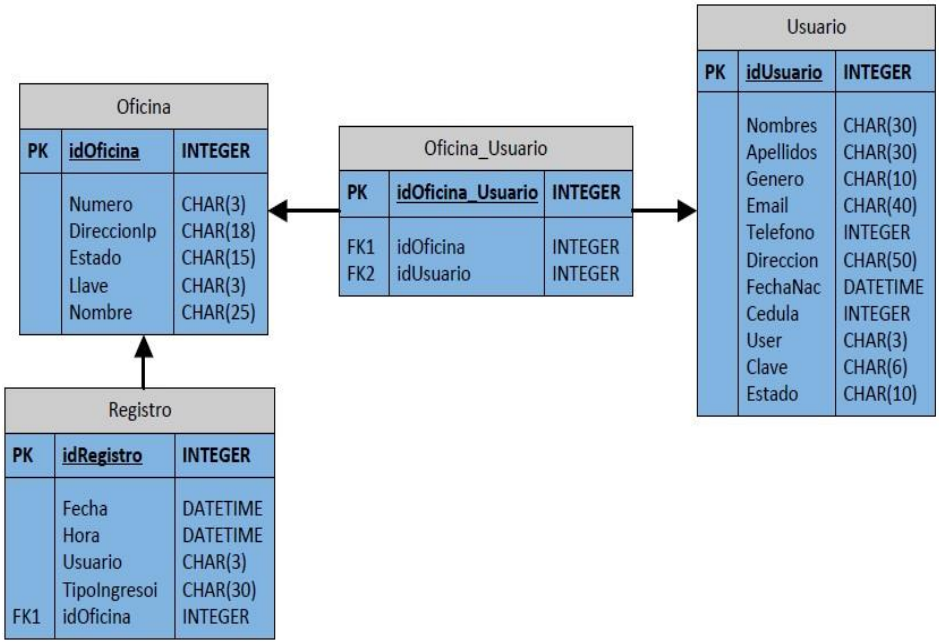


Figura 3.66 – Modelo lógico de la base de datos.

Las entidades Oficina y Usuario representan las oficinas y usuarios que se utilizan dentro del sistema, cada una de estas tienen atributos que los caracterizan y que se necesitan indicar a la hora de crear una nueva entidad de este tipo.

Este modelo lógico nos indica que en una oficina puede haber muchos usuarios autorizados para el ingreso y que un

usuario puede estar autorizado para ingresar a varias oficinas.

Estas relaciones son manejadas por la tabla Oficina_Usuario que tiene un id de oficina y uno de usuario como atributos para efectuar la correspondencia.

También, se puede observar que cada oficina tiene varios registros, donde se almacenan los intentos de ingreso de cada usuario, sean estos exitosos o fallidos con los datos necesarios correspondientes.

3.5.2. DESCRIPCIÓN BÁSICA DE LA APLICACIÓN EN JAVA

Se realiza una breve descripción de la interfaz principal, librerías y funciones utilizadas en la aplicación.

3.5.2.1. INTERFAZ GRÁFICA

La figura 3.67 muestra la ventana principal de la aplicación en Java, en donde se puede observar un registro de todos los usuarios que han solicitado autenticación, este registro

incluye fecha y hora de la solicitud, el id, nombre y apellido del usuario que intenta autenticarse, el nombre de la oficina que recibió la solicitud y si la autenticación fue exitosa o fallida.

En esta ventana también se muestra un registro del estado de las oficinas conectadas al sistema, mostrando la fecha y hora que la oficina cambio de estado a conectada, desonectada, bloqueada o desbloqueada.

Cuando una oficina se bloquea, el botón Desbloquear es habilitado. Si existen varias oficinas bloqueadas, al hacer clic en este botón se muestra una lista de estas oficinas, para poder seleccionar la que queremos desbloquear.

Para el ingreso de nuevos usuarios y oficinas están los botones Agregar Usuarios y Agregar Oficinas, al hacer clic en estos botones se despliega una ventana donde debemos ingresar los datos de las oficinas, como dirección IP en el caso de una oficina o nombre para un usuario.

Figura 3.67 – Ventana principal de la aplicación java.

Cuando se presiona el botón Buscar Usuarios se muestra una lista de todos los usuarios existentes en el sistema, permitiendo buscar por nombre o id de usuario, después de seleccionar uno se abrirá una ventana donde se mostrará

todos los registros de autenticación de este usuario y un botón para modificar sus datos.

Si se presiona el botón Buscar Oficinas, se muestra una lista de todas las oficinas existentes en el sistema, permitiendo buscar por nombre o id de oficina, después de seleccionar una, se despliega una ventana donde se muestra el registro de todos los usuarios que han solicitado autenticación desde que la oficina empezó a funcionar. Además, podemos modificar sus datos, agregar, quitar y ver los usuarios autorizados que pueden ingresar a la oficina.

En la ventana principal de la aplicación se encuentra el botón Consultar Historial, que permite ver todo el historial de autenticación de los usuarios o realizar la búsqueda por fechas.

Para poder observar todas las ventanas utilizadas por nuestra aplicación se puede consultar el anexo B.

3.5.2.2. LIBRERÍAS

Entre las librerías más importantes que manejamos dentro de la aplicación en Java están javax.swing, java.net, java.sql y javax.sound, las cuales se describen a continuación:

javax.swing: Es una librería gráfica de Java que proporciona un conjunto más sofisticado de componentes GUI (Interfaz gráfica de usuario) que su predecesor AWT (Abstract Window Toolkit), de quien hereda todo el manejo de eventos. Entre las componentes que Swing ofrece están: Paneles, botones, tablas, listas, casillas de verificación, etc.

Para tener acceso a los componentes que Swing provee es necesario importar el paquete javax.swing dentro del programa.

java.net: Es el API (Application Programming Interface) de Java que provee las clases necesarias para realizar conexiones y transacciones a través de la red. Importando el paquete java.net en nuestro programa podemos comunicar dos o más dispositivos de red.

Java.net brinda soporte para enviar y recibir paquetes UDP y TCP.

Las clases más importantes proporcionadas por esta librería son: InetAddress, Sockets y DatagramPacket y DatagramSocket, etc.

java.sql: Es una librería de Java que brinda los recursos necesarios para trabajar con bases de datos SQL. Utilizando esta librería podemos hacer una conexión básica o realizar cualquier tipo de tarea como: creación, consulta, actualización, modificación, borrado de tablas y ejecución de procedimientos almacenados en la base de datos SQL.

Esta librería contiene su propia estructura y hace llamado de sus propias clases, entre las principales tenemos: DriverManager, Connection, Statement, ResultSet, etc.

javax.sound: Es un API (Application Programming Interface) de Java que proporciona funcionalidad para la captura, procesamiento y reproducción de sonidos, incluidos los de audio y de interfaz digital de instrumentos musicales (MIDI) de datos.

Este API consta de 4 paquetes: javax.sound.midi, javax.sound.midi.spi, javax.sound.sampled, javax.sound.sampled.spi.

Los paquetes que terminan en (.spi) sirven para leer archivos sonoros, y los que terminan en (.midi) sirven para leer archivos de instrumentos.

Para el desarrollo del programa se usaron otras librerías que no son descritas aquí ya que son básicas de java, esta se las encuentran en el código del programa.

3.5.2.3. FUNCIONES ESENCIALES EN JAVA

A continuación, se explican solo las funciones más importantes que se utilizan en la aplicación para realizar la comunicación ethernet con la tarjeta DE2 y la conexión con la base de datos en la computadora.

El código principal se encuentra en el anexo C.

3.5.2.3.1. ENVIAR_PAQUETES

Para el envío de datos desde la aplicación en Java hacia la tarjeta DE2 utilizamos la función Enviar_Paquetes que se muestra en la figura 3.68.

```
public static void Enviar_Paquetes(String dato, byte[] direccion_ip)
        throws UnknownHostException, SocketException, IOException{

    String data=dato;
    byte[] buffer = data.getBytes();
    String b=new String(buffer);

    InetAddress address = InetAddress.getByAddress(direccion_ip);
    DatagramPacket packet = new DatagramPacket(
            buffer, buffer.length, address, 5005
            );
    DatagramSocket datagramSocket = new DatagramSocket();
    datagramSocket.send(packet);

    System.out.println(address.getHostAddress());
    System.out.println(b);
}
```

Figura 3.68 – Enviar_Paquetes en la aplicación java.

Esta función recibe los siguientes parámetros:

Dato que es el mensaje que se va a enviar y direccion_ip que es un arreglo de bytes que representa la dirección IP del destino.

La función Enviar_Paquetes es declarada como estática para poderla usar en cualquier clase dentro del programa.

Para su correcto funcionamiento necesita tres tipos de excepciones para manejar errores que se puedan presentar durante la transmisión de un paquete, estas son UnknowHostException que es lanzada para indicar que la dirección IP de un host no puede ser determinada, SocketException indica que hay un error al crear o acceder a un socket y IOException que es la clase general de excepciones producidas por fallos o interrupciones en las operaciones de entrada/salida.

Antes de transmitir un paquete se debe crear una instancia de la clase DatagramPacket, en la cual se envían cuatro parámetros: un arreglo de bytes que representan los datos a transmitir, la longitud del arreglo, un objeto de la clase InetAddress que indica la dirección IP destino y el número de puerto al que

se pretende enviar el paquete. En este caso se utiliza el 5005.

Para enviar estos parámetros al constructor de DatagramPacket, debemos convertir el dato que se recibió como String, a un arreglo de bytes y crear un objeto de la clase InetAddress a partir del arreglo de bytes que representa la dirección IP destino.

Por último creamos una instancia de la clase DatagramSocket llamado DatagramSocket que abre un socket para enviar paquetes a través de la función send, que recibe como parámetro un objeto de la clase DatagramPacket con el contenido del mensaje.

3.5.2.3.2. RECIBIR_PAQUETES

Para la recepción de datos desde la tarjeta DE2 utilizamos la función Recibir_Paquetes que se muestra en la figura 3.69.

Esta función recibe el siguiente parámetro:

Un entero que indica el puerto por el cual estará escuchando hasta recibir un paquete. Retorna un String con el dato.

La función Recibir_Paquetes es declarada como estática para poder usarla dentro de todas las clases del programa,

Dentro de esta función se declara una variable llamada receiveData que es un arreglo de 1024 bytes, el cual se utiliza para almacenar el contenido del paquete UDP.

Luego, se crea un DatagramPacket denominado receivePacket para la recepción de paquetes UDP, proporcionándole dos parámetros al constructor: receiveData y la longitud de este arreglo.

También, se crea una instancia de la clase DatagramSocket llamado serverSocket, que maneja un socket para poder recibir datagramas UDP. Además, se le específica a serverSocket donde debe guardar el datagrama recibido, utilizando la función

receive de DatagramSocket y mandándole como parámetro receivePacket.

DatagramPacket proporciona algunas funciones útiles como: obtener el dato del paquete, obtener la dirección IP del remitente, obtener el número de puerto desde donde se envió el datagrama, etc.

Una vez que recibimos un datagrama, usamos las funciones de DatagramPacket para obtener el dato del paquete y la dirección IP de la oficina que envío el paquete,

Por último, se procede a cerrar el socket abierto para la recepción de paquetes.

```
public static String Recibir_Paquetes(int port) {

    String mens="";
    try {
        DatagramSocket serverSocket = new DatagramSocket(port);
        byte[] receiveData = new byte[1024];

        System.out.printf("Listening on udp:%s:%d%n",
                InetAddress.getLocalHost().getHostAddress(), port);
        DatagramPacket receivePacket = new DatagramPacket(receiveData,
                        receiveData.length);

    while(true)
    {
          serverSocket.receive(receivePacket);
          String sentence = new String( receivePacket.getData(), 0,
                            receivePacket.getLength() );
          ip_envio=receivePacket.getAddress();
          mens=sentence;
          serverSocket.close();
          //System.out.println("RECEIVED: " + sentence);
          break;
    }
  } catch (IOException e) {
          System.out.println(e);
  }
  // should close serverSocket in finally block
    return mens;
}
```

Figura 3.69 – Recibir_Paquetes en la aplicación java.

3.5.2.3.3. DATABASE

En nuestra aplicación la conexión con la base de datos es manejada por medio de la clase Database.

Cuando se cree un objeto de esta clase a través de su constructor, se establece una conexión con la base de datos. Esta conexión es guardada en una variable tipo Connection.

Esta clase tiene cinco atributos que son necesarios para hacer la conexión con la base de datos: bd indica el nombre de la base de datos, login y password, el usuario y contraseña de administrador de la base bd, URL que corresponde a la ubicación de la base de datos, y un objeto de la clase Connection que es instanciado en null por defecto para indicar que esta desconectado al inicio.

En la figura 3.70 se muestra la clase Database.

```
public class Database {

   /* DATOS PARA LA CONEXION */
  private String bd = "oficinas";
  private String login = "root";
  private String password = "123456";
  private String url = "jdbc:mysql://localhost/"+bd;
  private Connection conn = null;

   public Database(){
      try{
         //obtenemos el driver de para mysql
         Class.forName("com.mysql.jdbc.Driver");
         //obtenemos la conexión
         conn = DriverManager.getConnection(url,login,password);
         if (conn!=null){
            //System.out.println("OK base de datos "+bd+" listo");
         }
      }catch(SQLException e){
         System.out.println(e);
      }catch(ClassNotFoundException e){
         System.out.println(e);
      }
   }
```

Figura 3.70 – Clase Database en java.

Dentro del constructor se crea una nueva instancia de la clase “com.mysql.jdbc.Driver”. Si ésta es creada correctamente, implica que el driver está registrado en el gestor de controladores JDBC. De esta manera se pueden crear conexiones MySQL, a través de getConnection que recibe como parámetros url, login y password.

Esta clase tiene dos excepciones dentro del constructor para manejar errores que se pueden presentar durante el intento de conexión con la base de datos: SQLException que provee información sobre errores relacionados con el acceso a la base de datos como tiempo de espera agotado, acceso denegado, entre otros y ClassNotFoundException que es lanzada cuando se intentar cargar una clase a través de su nombre, pero no se pudo encontrar una definición para la clase con el nombre especificado.

3.6. MONTAJE FINAL DEL SISTEMA

Para poner en marcha el sistema de autenticacion de usuarios, es necesario lo siguiente:

Correr la aplicación en el servidor, crear la base de datos en la computadora, conectar la PC y las tarjetas DE2 al swicth usando cable UTP tipo directo y encender los equipos.

Luego, se debe setear una dirección IP en la tarjeta DE2 y especificar cual será la dirección IP del servidor con la cual se va a conectar.

Posteriormente, de debe asignar una dirección IP a la computadora servidor.

Una vez realizado esto, se debe crear las oficinas asignandoles las direcciones IP de las tarjetas DE2 en la aplicación de la computadora y se debe crear los usuarios que se van a autenticar en la base de datos guardando su información básica.

Finalmente, se puede empezar a realizar el proceso de autenticación.

En la figura 3.71 se muestra el ensamblaje final del proyecto.

Figura 3.71 – Ensamblaje del proyecto.

CAPÍTULO 4

4. PRUEBAS Y RESULTADOS

Para realizar un análisis y demostrar cómo funciona el sistema en un ambiente real, las pruebas de laboratorio se efectuaron tomando en consideración los siguientes aspectos:

- Conexión.
- Rendimiento.
- Seguridad.
- Funcionamiento.

4.1. PRUEBAS DE CONEXIÓN

En esta sección se realizan pruebas del sistema en el escenario en que se pierde conexión entre la tarjeta DE2 y el servidor.

4.1.1. ESCENARIO A: PÉRDIDA DE CONEXIÓN

Cuando el sistema está funcionando con total normalidad, los dispositivos pueden perder conexión por siguientes motivos:

- Desconexión accidental del cable de red que conecta la tarjeta DE2 con el servidor.
- Cierre imprevisto de la aplicación que se está ejecutando en el servidor.

Para poder reconocer que hay problemas de conexión la tarjeta DE2 realiza un testeo de conexión cada 80 segundos.

Primero revisa el registro de estatus de red del DM9000A para ver si el cable de red está conectado a la red. Si está conectado entonces envía un paquete con el mensaje TEST al servidor, si no recibe un paquete con el mensaje CONEXIÓN, asume conexion perdida con el servidor.

El servidor en cambio para poder reconocer que hay problemas de conexión tiene un contador de 90 segundos en donde espera recibir cualquier paquetes desde la tarjeta DE2, si no recibe ningún paquete en ese lapso de tiempo asume pérdida de conexión con la tarjeta DE2 y la actualiza en el historial de la base de datos.

Para probar cómo reacciona el sistema ante cada una de estas situaciones procedimos a realizar las siguientes pruebas:

Desconexión del cable de red

Quitamos el cable de red que conectaba la tarjeta DE2 con la computadora servidor, luego esperamos los 80 segundos para que se realice la prueba de conexión, una vez finalizada la prueba se mostró un mensaje de error detallando que el cable de red se había desconectado tal como se observa en la figura 4.1.

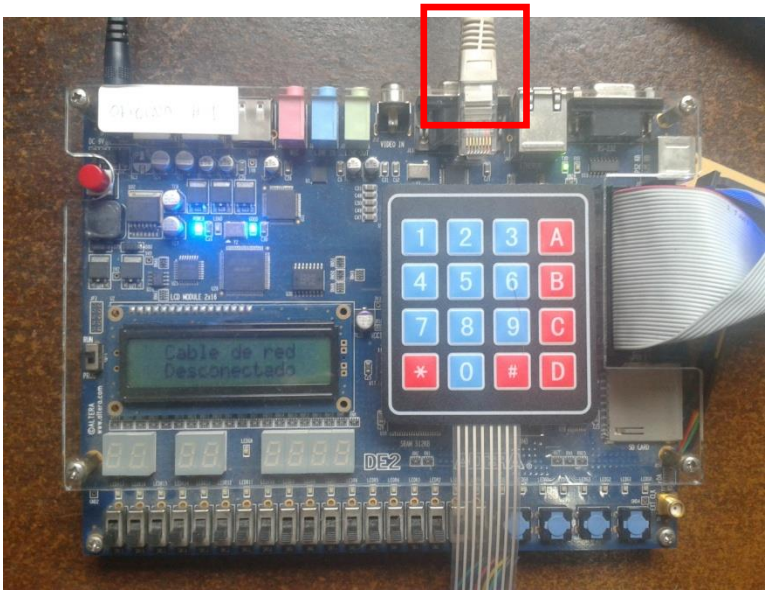


Figura 4.1 – Cable desconectado en la tarjeta DE2.

Cierre imprevisto de la aplicación del servidor

Cerramos la aplicación que se estaba ejecutando en el servidor, luego esperamos los 80 segundos para que se realice la prueba de conexión, una vez finalizada la prueba se mostró un mensaje de error detallando que se había perdido conexión con el servidor tal como se observa en la figura 4.2.

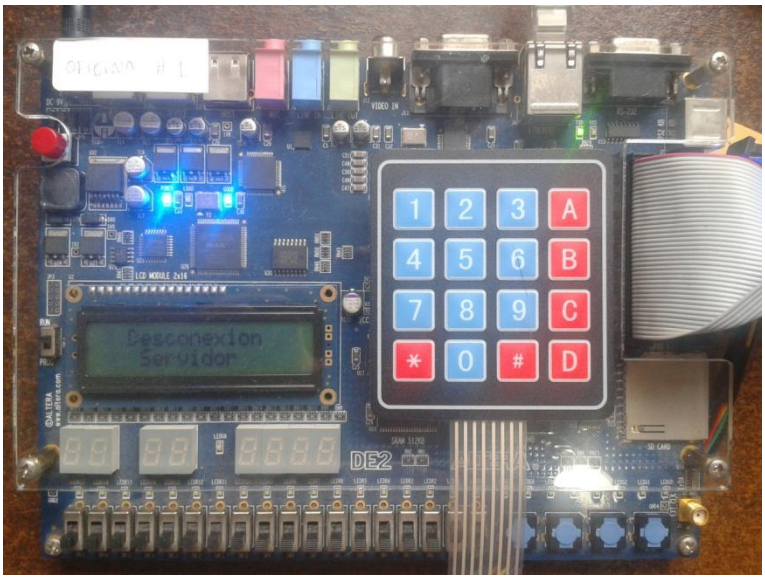


Figura 4.2 – Conexión perdida en la tarjeta DE2.

En cada una de las pruebas realizadas, el sistema nos permitió detectar el error y corregirlo fácilmente, mostrando en pantalla el problema que había que solucionar, haciendo solvente el sistema.

4.2. ANÁLISIS DE RENDIMIENTO

En este sección se analiza la razón de por qué se pierden paquetes, la tasa de paquetes perdidos vs recibidos, el tiempo que toma una autenticación de usuario en el sistema y el porcentaje autenticaciones realizadas con éxitos.

4.2.1. PÉRDIDA DE PAQUETES

Cuando enviamos paquetes desde la tarjeta DE2 al servidor y viceversa, se perdían paquetes durante la transmisión o no se los alcanzaba a procesar debido al siguiente motivo:

- Agotamiento del tiempo de espera (1 segundo) en la tarjeta DE2 para recibir respuesta desde el servidor.

La tarjeta DE2 cada vez que envía un paquete espera solamente un segundo para recibir una respuesta del servidor, que le permite saber que éste ha recibido el paquete enviado con normalidad, si no recibe o no alcanza a procesar la respuesta dentro en ese tiempo, asume que el paquete se ha perdido, dejando de procesar los paquetes que le lleguen después de ese tiempo, hasta que se

vuelva a enviar un nuevo paquete desde la tarjeta DE2 que requiera una respuesta del servidor.

La tarjeta DE2 en cada proceso de autenticación, envía con anterioridad un paquete con la llave al servidor para que pueda descifrar los datos. Luego cuando un usuario ingresa sus datos y realiza una petición de autenticación, se envía otro paquete con los datos cifrados al servidor.

Si cuando se enviaron los dos paquetes, la tarjeta DE2 recibió ambas respuestas desde el servidor, entonces se puede decir que el sistema realizó una autenticación de usuario con éxito.

Para observar el comportamiento del sistema en este punto, realizamos una prueba enviando 500 solicitudes de autenticaciones al servidor, llevando un registro de los paquetes perdidos tal como se muestra en la Tabla 4.1.

Nº AUTENTICACIONES	LLAVES PERDIDAS	DATOS PERDIDOS	AUTENTICACIONES FALLIDAS [%]
1-100	1	3	4
101-200	4	2	6
201-300	2	2	4
301-400	0	2	2
401-500	2	6	8
TOTAL	9/500	15/500	**4.8 %**

Tabla 4.1 – Tasa de paquetes perdidos vs recibidos

En la figura 4.3 se muestra gráficamente la proporción de los paquetes perdidos vs los paquetes recibidos en la tarjeta DE2.

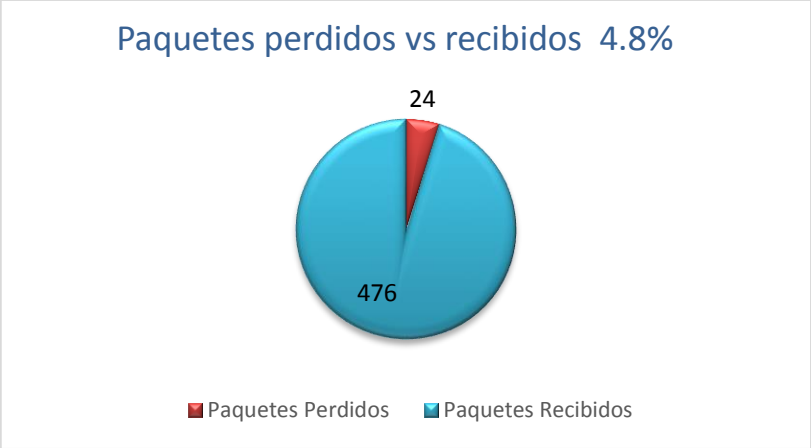


Figura 4.3 – Tiempo de autenticación.

Aunque la tasa de paquetes perdidos es baja, esto igual generaba problemas provocando que no se autentiquen los datos de los usuarios correctamente, haciendo que el proceso de autenticación no sea certero.

Solucionamos este inconveniente agregando un reenvío de paquete en caso que no se reciba respuesta desde el servidor.

Nuevamente realizamos las pruebas con otras 500 solicitudes de autenticaciones al servidor, en este caso aunque se seguían perdiendo paquetes, en el segundo reenvío si recibía la respuesta del servidor correctamente dando como resultado 0% de autenticaciones fallidas.

Terminado este análisis pudimos constatar que el sistema puede realizar autenticaciones de usuarios de manera certera.

4.2.2. TIEMPO DE AUTENTICACIÓN

Cuando realizamos una petición de autenticación al servidor existe un retardo de tiempo hasta mostrar en pantalla de la tarjeta DE2 el estado de autenticación del usuario.

Para medir el tiempo que se demora cada autenticación, se tomó en cuenta los siguientes aspectos:

- Etapa de envió de datos desde la tarjeta DE2.
- Tiempo de validación de los datos en el servidor.
- Etapa de procesamiento de la respuesta del servidor en la tarjeta DE2.

En la etapa de envío de datos, la tarjeta DE2 reinicia el DM9000A dando un retardo de 2 segundos, que es el tiempo necesario que necesita el chip para reiniciarse correctamente, luego empaqueta los datos y los envía al servidor.

Una vez que el servidor recibe el paquete con los datos del usuario este los verifica en la base de datos y envía una respuesta.

La tarjeta DE2 espera un tiempo de 1 segundo para recibir una respuesta tal como se explicó en la sección 4.4.1, luego de esto muestra en pantalla el estado de autenticación del usuario.

Para observar el comportamiento del sistema en cada una de estas etapas, procedimos a realizar 10 autenticaciones de usuarios midiendo el tiempo en cada una de ellas. Tabla 4.2

PRUEBA	ENVÍO DE DATOS TARJETA DE2 [s]	RESPUESTA SERVIDOR [s]	PROCESAMIENTO TARJETA DE2 [s]	TIEMPO TOTAL[s]
1	2.207	0.027	1.027	3.261
2	2.209	0.026	1.025	3.260
3	2.255	0.027	1.034	3.316
4	2.273	0.031	1.026	3.330
5	2.262	0.030	1.033	3.325
6	2.294	0.027	1.021	3.342
7	2.210	0.025	1.018	3.253
8	2.239	0.029	1.024	3.292
9	2.196	0.028	1.028	3.252
10	2.280	0.031	1.022	3.333
PROMEDIO	2.2425	0.0281	1.0258	**3.2964**

Tabla 4.2 – Tiempo de autenticación

Finalmente obtuvimos un valor de tiempo de 3.2964 segundos, que es un valor aceptable para realizar una autenticación de usuario en nuestro sistema.

4.3. PRUEBAS DE SEGURIDAD

En esta sección se realiza las pruebas seguridad del sistema en el escenario en que se realiza un sniffing de paquetes en la red.

4.3.1. ESCENARIO B: SNIFFING DE PAQUETES

Cuando enviamos paquetes con los datos de los usuarios desde la tarjeta DE2 al servidor a través de la red LAN, fue posible capturar los datos de los usuarios durante la transmisión de los paquetes utilizando el programa de sniffing de paquetes Wireshark.

Para asegurar la confiabilidad de los datos, el sistema envía las claves cifradas. El algoritmo de cifrado que utiliza es simétrico ya que utiliza la misma llave para cifrar y descifrar los datos, esta es generada de manera aleatoria en la tarjeta DE2 y enviada al servidor cada vez que finaliza una autenticación.

La característica principal del algoritmo de cifrado es que da valores diferentes a la mayoría de número así se repitan en la misma clave, haciendo más difícil que se pueda buscar un patrón para descifrar los datos, lo que le da robustez al algoritmo utilizado.

Para comprobar la efectividad del algoritmo de cifrado en este punto, realizamos la captura de 10 paquetes con los datos de los usuarios utilizando diferentes llaves, y observamos cómo se cifraban los datos.

La tabla 4.3 muestra los 5 primeros paquetes capturados con la clave cifrada cuando se ingresa un clave con los dígitos repetidos.

LLAVE	CLAVE	CIFRADO
765	222222	06100700 1204
526	222222	43600700 0700
167	222222	27602500 0700
450	222222	14300700 0300
037	222222	07004640 1700

Tabla 4.3 – Clave con dígitos iguales

La tabla 4.4 muestra los 5 siguientes paquetes capturados con la clave cifrada cuando se ingresa un clave con los dígitos diferentes.

LLAVE	CLAVE	CIFRADO
436	123456	44302570 5300
760	123456	53701440 2610
018	123456	26104261 3000
307	123456	63702100 6610
836	123456	54300661 5300

Tabla 4.4 – Clave con dígitos diferentes

Para mostrar cómo se capturaron los paquetes en esta prueba, se tomó una captura de pantalla del Wireshark figura 4.4, cuando se envió el paquete con la clave 123456 del usuario 101 cifrada utilizando la llave 836 (Tabla 4.4).

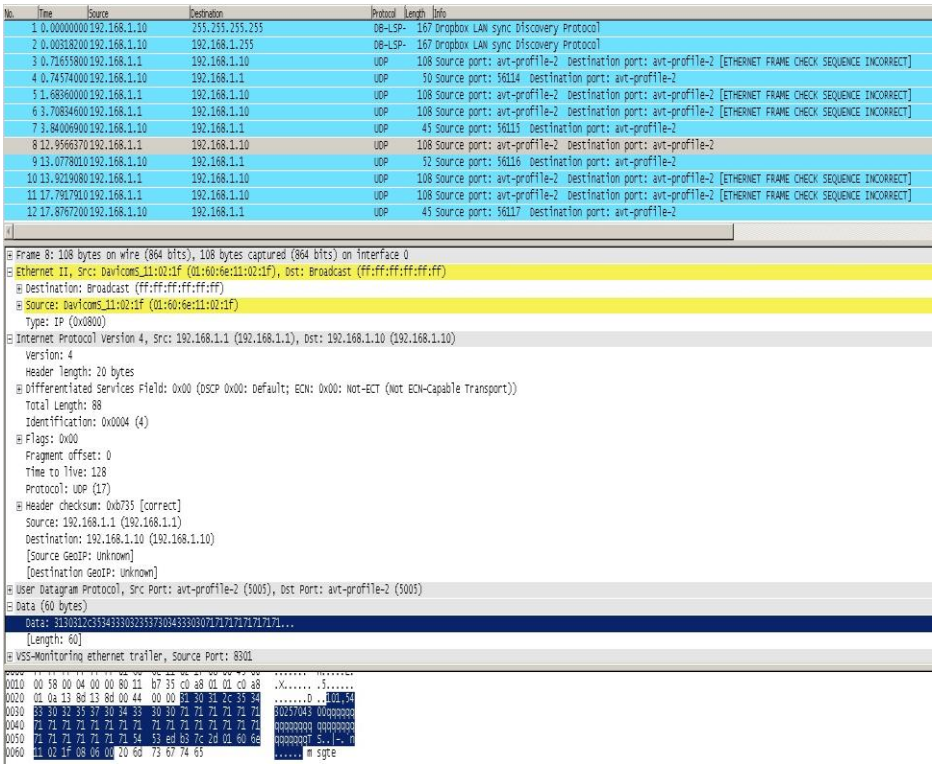


Figura 4.4 – Paquete capturado en Wireshark.

Cuando concluimos la prueba, pudimos constatar la confiabilidad de los datos ya que no fue posible obtener la clave del usuario.

4.4. PRUEBAS DE DESEMPEÑO

En esta sección se realizan las pruebas cuando el sistema funciona con normalidad.

4.4.1. ESCENARIO C: COMUNICACIÓN EXITOSA

Cuando un usuario ingresa sus datos, estos se empaquetan y se envían desde la tarjeta DE2 al servidor para validar los datos.

En este caso, cada una de las tarjetas DE2 espera que se ingrese un ID y una clave de usuario para empezar el proceso de autenticación con el servidor.

La figura 4.5 muestra el momento en que se ingresa el ID de un usuario.

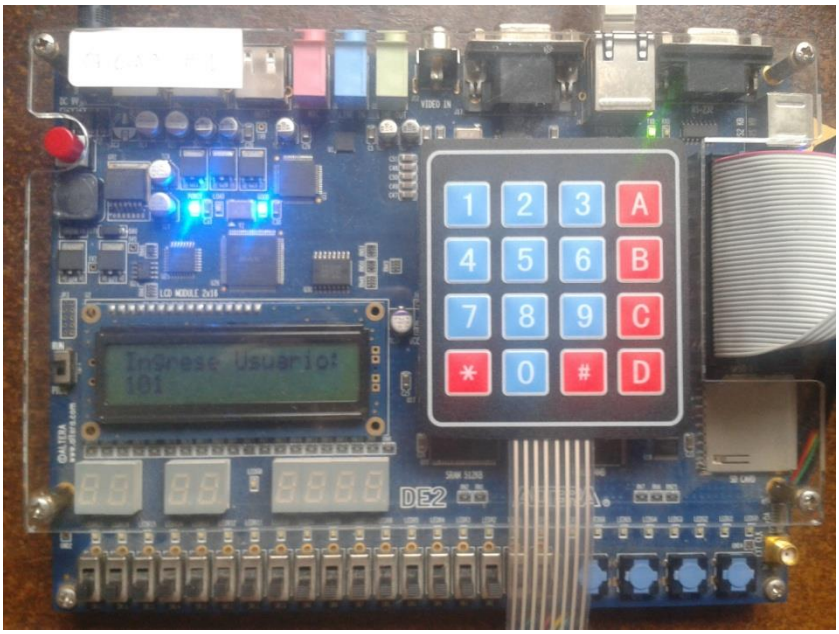


Figura 4.5 – Ingreso del ID de usuario en la tarjeta DE2.

Si el ID y clave ingresados y enviados a la aplicación en el servidor son correctos, el sistema permite el ingreso a la oficina.

En la figura 4.6 se observa cuando el sistema permite el ingreso de un usuario autorizado, encendiendo el Led verde que simula que se abre la puerta.

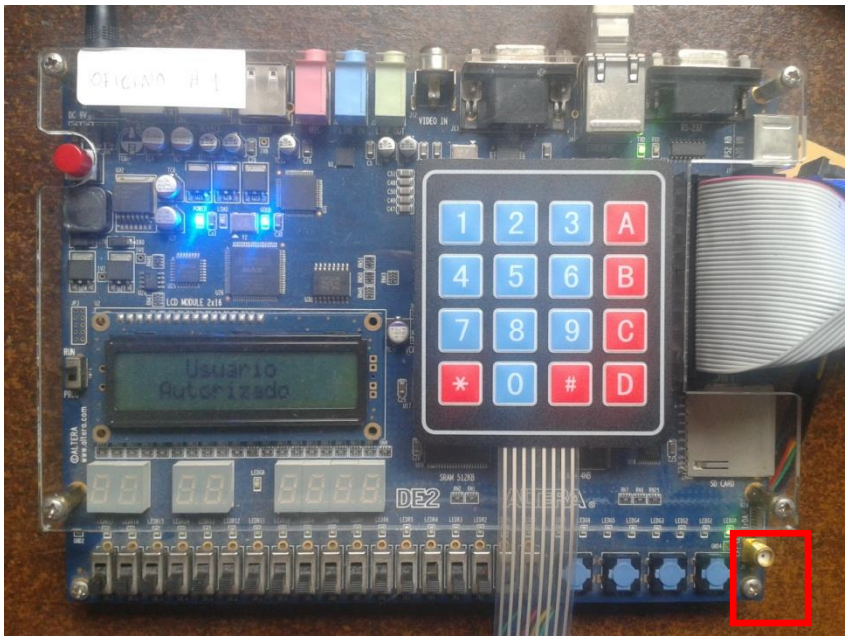


Figura 4.6 – Simulación de puerta desbloqueada.

Una vez que desbloquea la tarjeta vuelve al menú de ingreso de usuario para empezar la autenticación de un nuevo usuario.

La figura 4.7 muestra la oficina llamada Recursos Humanos conectada al sistema, además de la autenticación exitosa realizada por el usuario 101.

Figura 4.7 – Autenticación mostrada en la aplicación.

Los intentos fallidos de un usuario se muestran en color rojo claro y los exitosos en color amarillo, cada mensaje tiene la hora y fecha en que se recibió el mensaje, la oficina que envió el mensaje y el usuario que desea autenticarse.

Cuando una oficina está conectada el mensaje se muestra en color verde, si por algún motivo pierde conexión, se muestra de color gris y si el administrador desbloquea la oficina el mensaje es de color celeste. Estos mensajes incluyen la fecha y hora en el que sucedió el evento.

El historial que se muestra en la figura 4.7 es la que se guarda en la base de datos.

Durante el proceso de autenticación en la consola de Eclipse y Netbeans se muestran mensajes que indican el proceso que se está llevando a cabo en el sistema, como se muestran en la figura 4.8 y 4.9.

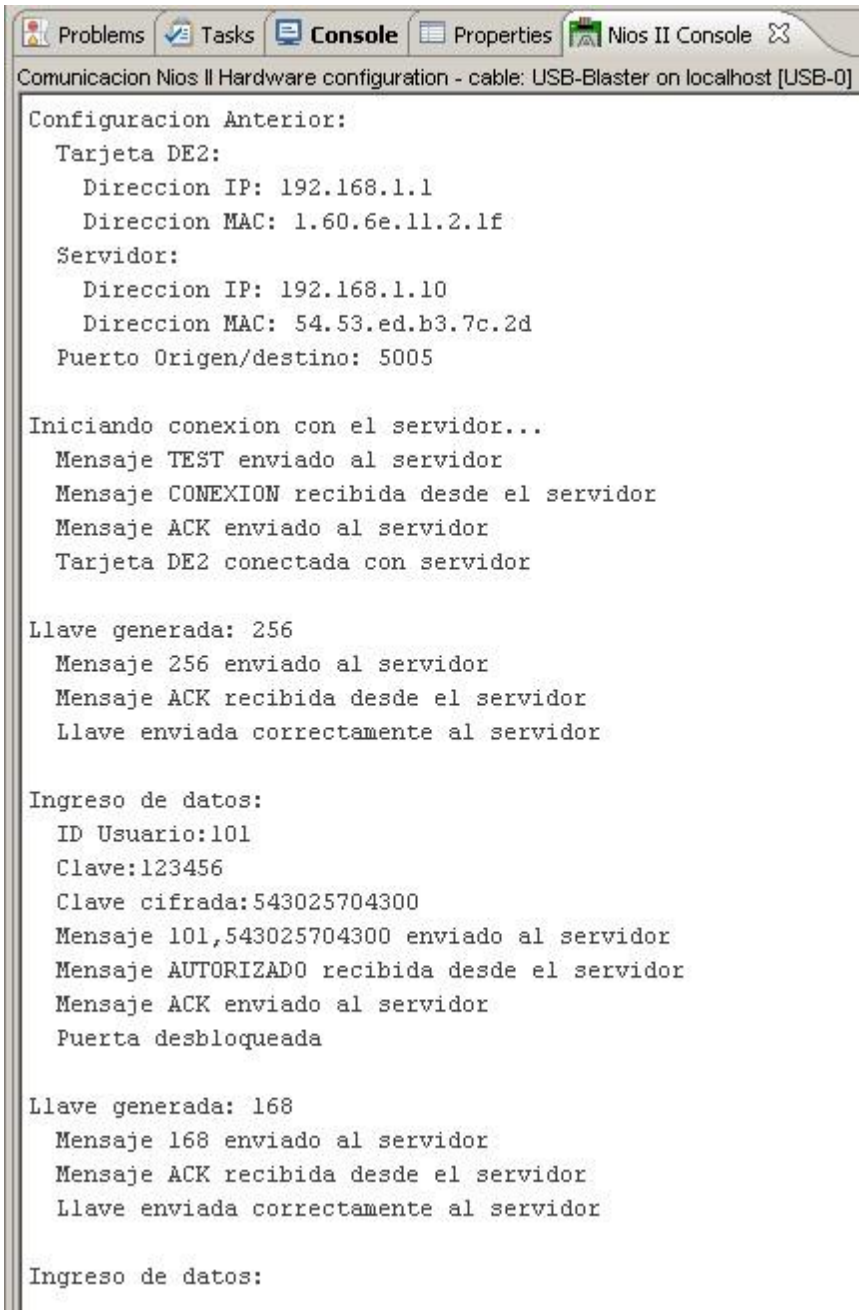


Figura 4.8 – Mensajes en consola - tarjeta DE2.

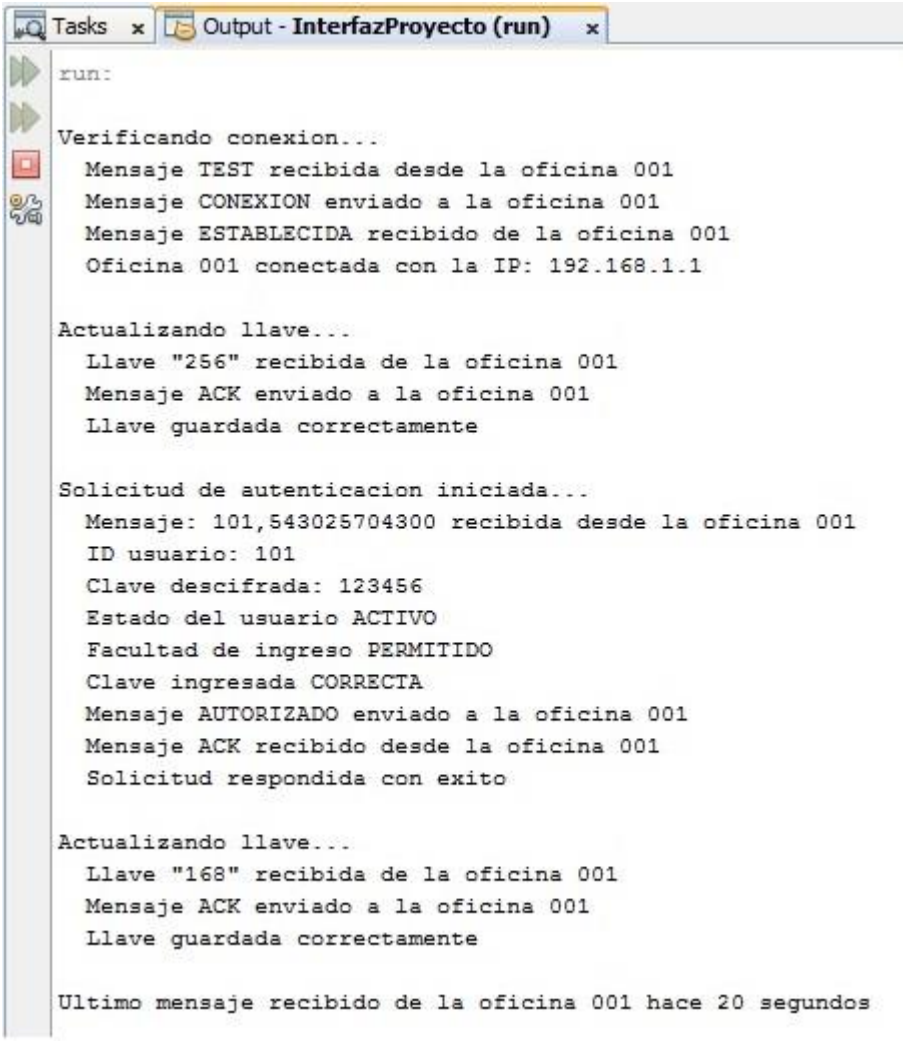


Figura 4.9 – Mensajes en consola – PC Servidor.

CONCLUSIONES

1. Los sistemas embebidos ofrecen una gran versatilidad para elaborar aplicaciones, debido a que permite agregar módulos y realizar cambios en el hardware y software en cualquier etapa de desarrollo del proyecto.

2. La tarjeta DE2 con microprocesador NIOS II permite realizar una comunicación Ethernet exitosa con una computadora a través de una red LAN utilizando el modelo de referencia TCP/IP.

3. El diseño del sistema de autenticación es robusto, ya que tiene mecanismos de detección de errores que indican cuando se configura erróneamente las direcciones IP, cuando se pierde conexión entre los dispositivos y cuando se pierden paquetes.

4. Los driver para manejar la comunicación Ethernet en el chip DM9000A no funcionan correctamente cuando se reciben muchos paquetes en el buffer de memoria del chip.

5. La tasa de paquetes perdidos vs enviados se redujo aproximadamente a cero, haciendo un reenvío de datos cuando se pierde el paquete por primera vez.

6. El tiempo para autenticar usuarios es un poco elevado debido a que se reinicializa el chip DM9000A y se agrega un retardo en el

programa para poder leer los paquetes recibidos sin que se descarte ningún paquete.

7. El sistema de autenticación ofrece seguridad contra el sniffing de paquetes siempre y cuando no se conozcan las llaves transmitidas ni el algoritmo de cifrado de datos.

RECOMENDACIONES

1. Usar la computadora media o básica que brinda Qsys para la creación del hardware del sistema, y luego añadir o eliminar los módulos necesarios para nuestro diseño.

2. Realizar un direccionamiento IP previo de las tarjetas DE2 que se utilizarán en las oficinas y en la computadora servidor, antes de configurarlas y conectarlas al sistema.

3. Elaborar un documento con las políticas de direccionamiento IP, este archivo ayudara a resolver problemas de conectividad en el sistema que ocurran posteriormente.

4. Utilizar vlans en el switch para aislar al sistema del tráfico que no sea generado por la aplicación.

5. En el caso que el switch utilizado tenga configurado vlans, debe asegurarse que todos los puertos conectados a las oficinas y al servidor se encuentren en una misma vlan.

6. Agregar políticas de seguridad en los puertos del switch del sistema para limitar que computadoras no autorizadas se conecten a la red y evitar que se pueda realizar un sniffing de paquetes.

7. Desactivar las aplicaciones en la computadora servidor que generen un exceso de tráfico dentro de la red para no sobrecargar el buffer de memoria del DM9000A.

8. Se sugiere utilizar una computadora para ejecutar la aplicación ubicada en un cuarto que tenga protección contra el manejo de usuarios no autorizados y fuentes de alimentación alternas en caso de pérdida de energía eléctrica.

ANEXOS

ANEXO A

CÓDIGO FUENTE EN LENGUAJE C

LIBRERÍAS

```
/* Librerías Utilizadas */
#include "basic_io.h"
#include "test.h"
#include "dm9000a.h"
#include <time.h>
#include <lcd.h>
```

DECLARACIÓN DE CONSTANTES

```
/* Declaración de Constantes */
#define gpio_in 0x00800060 // Dirección de memoria PIO Entrada
#define gpio_out 0x00800040 // Dirección de memoria PIO Salida
#define green_leds 0x10000010  // Dirección de memoria Leds Verdes
#define red_leds 0x10000000  // Dirección de memoria Leds Rojos
#define MAX_MSG_LENGTH 128   // Máxima Longitud del Mensaje
#define IP_LENGTH_OFFSET 16  // Desplazamiento del buffer para obtener ip
#define UDP_PACKET_PAYLOAD_OFFSET 42 // Desplazamiento del buffer para obtener
datos
#define UDP_PACKET_LENGTH_OFFSET 38  //Desplazamiento del buffer para obtener
longitud del paquete
#define UDP_PACKET_PAYLOAD (transmit_buffer + UDP_PACKET_PAYLOAD_OFFSET)
#define IP_HEADER_CHECKSUM_OFFSET 24  // Desplazamiento del buffer para obtener
checksum de la cabecera ip
#define IP_PACKET_ID_OFFSET 18  // Desplazamiento del buffer para obtener el id
del paquete
#define IP_DESTINATION_ADDRESS_OFFSET 30  // Desplazamiento del buffer para
obtener la ip destino
```

DECLARACIÓN DE VARIABLES

```
/* Declaracion de variables */

int contador_verificacion_conexion=0, jjj=1, salir=1, estado_ingreso=0;
int contador_verificacion_conexion1=0;
int paquetes_perdidos=0, paquetes_recibidos=0,paquetes_perdidos_doble=0,
```

```
    repeticion_envio_llave=0;
int p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,p11,p12,p13,p14,p15,p16,p17,p18,
    p19,p20;
int pp1,pp2,pp3;
int salto=0, saltodes=0;
char* mensaje;
int binario_llave[4]={0,0,0,0};
int binario_llave_nuevo[4]={0,0,0,0};
int clave_cifrada[12]={0,0,0,0,0,0,0,0,0,0,0,0};
int clave_cifrada_nuevo[12]={0,0,0,0,0,0,0,0,0,0,0,0};
char clave_cifrada_car[12]=" ";
char clave_cifrada_car_nuevo[12]=" ";
int clave_cifrada_binario[12]={0,0,0,0,0,0,0,0,0,0,0,0};
int clave_cifrada_binario_nuevo[12]={0,0,0,0,0,0,0,0,0,0,0,0};
int imprimir_consola = 0, receivedMessageFlag = 0, estado_sistema=0,
    desplazamiento;
int desplazamiento_nuevo;
extern char clave[6]="qqqqqq",clave_cambio[6]="qqqqqq",
            llave_actual[3]=" ";
extern char clave_nueva[6]="qqqqqq", clave_nueva_confirmacion[6]=
           "qqqqqq";
int clave_num[6],llave_actual_num[3],clave_num_nuevo[6],
    llave_actual_num_nuevo[3];
extern char usuario[3]="qqq", usuario_cambio[3]="qqq";
extern char temporal[9]="qqqqqqqqq";
extern char mensaje_enviar[16]="qqqqqqqqqqqqqqqq";
extern char mensaje_enviar_cambio_clave[29]="qqqqqqqqqqqqqqqqqqq
            qqqqqqqqqq";
unsigned char mac_address_target[6] = { 0x01, 0x60, 0x6E, 0x11,
              0x02, 0x0F  };
unsigned char mac_address_pc[6] = { 0x54, 0x53, 0xeD, 0xb3,
              0x7C, 0x2D };
unsigned int interrupt_number,receive_buffer_length = 0;
unsigned char receive_buffer[1600];
char receivedMessage[1600];
unsigned short transmit_checksum, receive_checksum, packet_id = 0x0000;
char* pointer_to_transmit_buffer, pointer_to_receive_buffer;


unsigned char arp_buffer[] = {  //Campos de la trama ARP
  //Cabecera Ethernet

  /*Servidor*/
  0x54, 0x53, 0xED, 0xB3, 0x7C, 0x2D,   //Dirección MAC destino
  /*Tarjeta DE2*/
  0x01, 0x60, 0x6E, 0x11, 0x02, 0x2F,   //Dirección MAC origen

  0x08, 0x06, // Tipo de Paquete = ARP
  0x00, 0x01, // Tipo de Hardware = Ethernet 0x0001
  0x08, 0x00, // Tipo de Protocolo = IP (0x0800)
  0x06, 0x04, // 6 bytes para dirección MAC, 4 bytes para dirección IP
  0x00, 0x02, // 0x0002 indica un ARP reply

  /*Tarjeta DE2*/
  0x01, 0x60, 0x6E, 0x11, 0x02, 0x2F,   //Dirección MAC origen
```

```
  0xc0,0xa8,0x01,0x02,                  //Dirección IP origen
  /*Servidor*/
  0x54, 0x53, 0xED, 0xB3, 0x7C, 0x2D,   //Dirección MAC destino
  0xc0,0xa8,0x01,0x0A                   //Dirección IP destino
};

unsigned char transmit_buffer[] = {   //Campos de la trama mensaje
               // Cabecera Ethernet

  /*Broadcast*/
  0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,  //Dirección MAC broadcast
  /*Tarjeta DE2*/
  0x01, 0x60, 0x6E, 0x11, 0x02, 0x2F,  //Dirección MAC origen

  0x08, 0x00,                         // Tipo de Paquete: 0x800 = IP

               // Cabecera IP
  0x45,               // version (IPv4), Longitud de la cabecera =
                                   //20 bytes
  0x00,               // Campos de servicio diferenciado
  0x00,0x9C,          // Longitud total: 20 bytes para cabecera IP +
                          // 8 bytes para cabecera UDP + bytes de los
                                          //datos
  0x00, 0x00,         // ID del paquete
  0x00,               // Banderas
  0x00,               // fragment offset
  0x80,               // time-to-live
  0x11,               // Protocolo de la capa de transporte: 11 = UDP
  0x00,0x00,          // checksum de la cabecera //

  /*Tarjeta DE2*/
  0xc0,0xa8,0x01,0x02,    //Dirección IP origen
  /*Servidor*/
  0xc0,0xa8,0x01,0x0A,    //Dirección IP destino

  // Cabecera UDP
  0x13,0x8D, // Puerto Fuente (5005)
  0x13,0x8D, // Puerto Destino (5005)
  0x00,0x88, // Longitud (136: 8 para la cabecera UDP + dato)
  0x00,0x00, // checksum: 0 = ninguno

  // Carga UDP
  0x74, 0x65, 0x73, 0x74, 0x20, 0x6d, 0x73, 0x67,
  0x74, 0x65, 0x73, 0x74, 0x20, 0x6d, 0x73, 0x67,
  0x74, 0x65, 0x73, 0x74, 0x20, 0x6d, 0x73, 0x67,
  0x74, 0x65, 0x73, 0x74, 0x20, 0x6d, 0x73, 0x67,
  0x74, 0x65, 0x73, 0x74, 0x20, 0x6d, 0x73, 0x67,
  0x74, 0x65, 0x73, 0x74, 0x20, 0x6d, 0x73, 0x67,
  0x74, 0x65, 0x73, 0x74, 0x20, 0x6d, 0x73, 0x67,
  0x74, 0x65, 0x73, 0x74, 0x20, 0x6d, 0x73, 0x67,
  0x74, 0x65, 0x73, 0x74, 0x20, 0x6d, 0x73, 0x67,
  0x74, 0x65, 0x73, 0x74, 0x20, 0x6d, 0x73, 0x67,
  0x74, 0x65, 0x73, 0x74, 0x20, 0x6d, 0x73, 0x67,
  0x74, 0x65, 0x73, 0x74, 0x20, 0x6d, 0x73, 0x67,
  0x74, 0x65, 0x73, 0x74, 0x20, 0x6d, 0x73, 0x67,
```

```
  0x74, 0x65, 0x73, 0x74, 0x20, 0x6d, 0x73, 0x67,
  0x74, 0x65, 0x73, 0x74, 0x20, 0x6d, 0x73, 0x67,
  0x74, 0x65, 0x73, 0x74, 0x20, 0x6d, 0x73, 0x67
};
```

DECLARACIÓN DE FUNCIONES

```
/* Declaración de Funciones */

static void ethernet_interrupt_handler(void);
void Enviar_Mensaje(char* );
static unsigned short Calcular_Checksum(char* );
void Setear_Longitud_Paquete(unsigned int );
void Setear_ID_Paquete(void);
void Setear_Checksum(char* , unsigned int );

void LCD_cursor( int, int );
void LCD_text( char * );
void LCD_cursor_off( void );
void LCD_clear( void );

int EsNumero(char);
char leer_tecla(void);
void ingreso_direccion_ip_tarjeta(void);
void ingreso_direccion_ip_servidor(void);
void generar_llave_nueva(void);
char leer_tecla_tiempo(void);
void ingreso_usuario_clave(void);
void cifrar_clave(void);
void clave_cifrada_caracter(void);
void clave_cifrada_entero(void);
void entero_binario(int);
void desplazamiento_binario_entero(int);
void cifrar_clave_nueva(void);
```

CÓDIGO DE LAS FUNCIONES

```
/*********************************************************************
 * Subrutina para recibir paquetes UDP
*********************************************************************/
static void ethernet_interrupt_handler() {
  unsigned int receive_status;

  if(imprimir_consola==1){
     receive_buffer_length=0;
     receive_status = Recibir_Paquete(receive_buffer, &
                                      receive_buffer_length);
```

```
if (receive_status == DMFE_SUCCESS) {
  if (receive_buffer_length >= 14) {
     // Es un Paquete Ethernet
    if ((receive_buffer[12] == 8 && receive_buffer[13] == 0 &&
    receive_buffer_length >= 34)) {
       // Es un paquete IP
       if ((receive_buffer[23] == 0x11)){
          // Es un paquete UDP
          if (receive_buffer_length >= UDP_PACKET_PAYLOAD_OFFSET) {
             receivedMessageFlag = 1;
             int i;
             for(i = 0; i < receive_buffer_length; i++){
                if ((i+1) % 8 == 0){
                }
             }
             char* message_received = receive_buffer +
                                 UDP_PACKET_PAYLOAD_OFFSET;
             mensaje=message_received;

             if(strcmp(message_received,"autorizado")==0){
               estado_sistema=1;
                printf("\n  Mensaje AUTORIZADO recibida desde el
                                      servidor\n");
             }
             else if(strcmp(mensaje,"denegado")==0){
               estado_sistema=2;
                printf("\n  Mensaje DENEGADO recibida desde el
                                 servidor\n");
             }
             else if(strcmp(mensaje,"bloquear")==0){
             estado_sistema=3;
                printf("\n  Mensaje BLOQUEAR recibida desde el
                                 servidor\n");
             }
             else if(strcmp(mensaje,"desbloquear")==0){
             estado_sistema=4;
                printf("\n  Mensaje DESBLOQUEAR recibida desde el
                                 servidor\n");
             }
             else if(strcmp(mensaje,"conexion")==0){
             estado_sistema=5;
                printf("\n  Mensaje CONEXION recibida desde el
                                 servidor\n");
             }
             else if(strcmp(mensaje,"idincorrecto")==0){
             estado_sistema=6;
                printf("\n  Mensaje IDINCORRECTO recibida desde el
                                 servidor\n");
             }
             else if(strcmp(mensaje,"cambiado")==0){
             estado_sistema=7;
                printf("\n  Mensaje CAMBIADO recibida desde el
                                 servidor\n");
             }
             else if(strcmp(mensaje,"ack")==0){
```

```
                    estado_sistema=8;
                   printf("\n  Mensaje ACK recibida desde el
                                   servidor\n");
                }

                pointer_to_receive_buffer = &receive_buffer[0];
                receive_checksum = Calcular_Checksum(

      pointer_to_receive_buffer);

                if( ((receive_checksum >> 8) == (receive_buffer[
                                IP_HEADER_CHECKSUM_OFFSET]))
                      && ((receive_checksum & 0x00ff) == (receive_buffer[
                                    IP_HEADER_CHECKSUM_OFFSET+1])) )
                {
                   //printf("\nChecksum OK");
                }
                else {
                   //printf("\nChecksum test failed. Message ignored.\n");
                }

             }
          } else {
             //printf("\nReceived non-UDP packet\n");
          }
        } else {
          //printf("\nReceived non-IP packet\n");
          Transmitir_Paquete(arp_buffer, 60);
          //printf("\nARP reply sent.\n");
        }
      } else {
        //printf("Malformed Ethernet packet\n");
      }
    } else {
      //printf("Error recibiendo paquete\n");
    }

    /* Limpiar el registro ISR del DM9000A ISR: PRS, PTS, ROS, ROOS
          4   bits, by RW/C1 */
    iow(ISR, 0x3F);

    /* Re-Habilitar las interrupciones del DM9000A */
    iow(IMR, INTR_set);
}
    /* Limpiar el registro ISR del DM9000A ISR: PRS, PTS, ROS, ROOS 4
       bits, by RW/C1 */
    iow(ISR, 0x3F);

    /* Re-Habilitar las interrupciones del DM9000A */
    iow(IMR, INTR_set);
}
```

```
/*********************************************************************
 * Subrutina para Enviar un mensaje
*********************************************************************/
void Enviar_Mensaje (char* message) {

       unsigned int longitud_mensaje = 0;
         unsigned int packet_checksum;

          while(message[longitud_mensaje] != '\0'){

         transmit_buffer [UDP_PACKET_PAYLOAD_OFFSET + longitud_mensaje]
         = message [longitud_mensaje];
      longitud_mensaje++;
  }
  transmit_buffer [UDP_PACKET_PAYLOAD_OFFSET + longitud_mensaje]
  = '\0';
  longitud_mensaje++;

  // Setear el ID del Paquete
  Setear_ID_Paquete ();

  // Set la Longitud Del Paquete
  Setear_Longitud_Paquete (longitud_mensaje);
  // Obtener el checksum para la cabecera ip
  pointer_to_transmit_buffer = &transmit_buffer [0];
  packet_checksum = Calcular_Checksum (pointer_to_transmit_buffer);
  Setear_Checksum (pointer_to_transmit_buffer, packet_checksum);

  if (Transmitir_Paquete (transmit_buffer, 0x6c) ==DMFE_SUCCESS) {
        //printf ("\nMensaje enviado exitosamente");
  } else {
        //printf ("\nOcurrio un falla enviando el mensaje\n");
  }
}

/*********************************************************************
 * Subrutina para Calcular Checksum
*********************************************************************/
static unsigned short Calcular_Checksum (char* buffer) {
    int access;
    unsigned int checksum = 0x00000000;

    // Cabecera IP desde el byte 14 al 33
    for (access = 14; access < 34; access++) {
        if (access != IP_HEADER_CHECKSUM_OFFSET) {
             checksum = checksum + ((((int) buffer [access]) << 8) &
                                    0x0000ff00); //msb
            access++;
            checksum = checksum + (((int) buffer [access]) &
                                    0x000000ff); //lsb
            /* Esto añade al checksum lo sisguiente:
            Se asume que buffer [22] = 0x80 and buffer [23] = 0x11
            Luego buffer [22] << 8) = 0x8000 */
        } else {
             access++; //Ya que el checksum tiene 2 bytes de longitud,
```

```
                                //éste obvia
                            //checksum_offset + 1 también
            }
            // Ignoramos los campos del checksum Del 24 al 25
        }
        checksum = (checksum & 0x0FFFF) + ((checksum & 0xF0000) >> 16);
        checksum = ~checksum;

        return (short) checksum;
}

/*****************************************************************
 * Subrutina para setear longitud del paquete
*****************************************************************/
void Setear_Longitud_Paquete (unsigned int packet_length) {

  // Setear el campo longitud de la trama Ethernet apropiadamente
  transmit_buffer [IP_LENGTH_OFFSET] = (20 + 8 + packet_length) >> 8;
  transmit_buffer [IP_LENGTH_OFFSET + 1] = (20 + 8 + packet_length) &
  0xff;

  // Setear el campo longitud Del UDP apropiadamente
  transmit_buffer [UDP_PACKET_LENGTH_OFFSET] = (8 + packet_length) >>8;

transmit_buffer [UDP_PACKET_LENGTH_OFFSET + 1] = (8 + packet_length) &

0xff;

  return;
}

/*****************************************************************
 * Subrutina para setear el ID de un paquete
*****************************************************************/
void Setear_ID_Paquete () {

   // Incluir el ID del paquete
   // Mantener la longitud en 2 bytes
   packet_id = (packet_id + 1) & 0xffff;
   transmit_buffer [IP_PACKET_ID_OFFSET] = packet_id >> 8;
   transmit_buffer [IP_PACKET_ID_OFFSET + 1] = packet_id & 0x00ff;
   return;
}

/*****************************************************************
 * Subrutina para setear el checksum de un paquete
*****************************************************************/
void Setear_Checksum (char* buffer, unsigned int checksum) {

    // Incluir el checksum
    // MSB
    buffer [IP_HEADER_CHECKSUM_OFFSET] = (checksum & 0x0000ffff) >> 8;
    // LSB
    buffer [IP_HEADER_CHECKSUM_OFFSET + 1] = checksum & 0x000000ff;
```

```
    return;
}

/*********************************************************************
 * Subrutina para mover el cursor del LCD
*********************************************************************/
void LCD_cursor(int x, int y)
{
	volatile char * LCD_display_ptr = (char *) 0x10003050;
	char instruction;

	instruction = x;
	if (y != 0) instruction |= 0x40;
	instruction |= 0x80;
	*(LCD_display_ptr) = instruction;
}

/*********************************************************************
 * Subrutina para enviar una cadena de texto al LCD
*********************************************************************/
void LCD_text( char * text_ptr)
{
	volatile char * LCD_display_ptr = (char *) 0x10003050;

	while ( *(text_ptr) )
	{
		*(LCD_display_ptr + 1) = *(text_ptr); // escribir en el

//registro de datos del
                                                                    //LCD
		++text_ptr;
	}
}

/*********************************************************************
 * Subrutina para apagar el cuursor del LCD
*********************************************************************/
void LCD_cursor_off(void)
{
	volatile char * LCD_display_ptr = (char *) 0x10003050;
	*(LCD_display_ptr) = 0x0C;
}

/*********************************************************************
 * Subrutina para limpiar la pantalla del LCD
*********************************************************************/
void LCD_clear(void)
{
	LCD_cursor (0,0);
	LCD_text ("                ");
	LCD_cursor (0,1);
	LCD_text ("                ");
}
```

```
/*********************************************************************
 * Subrutina para convertir un numero entero a caracter
*********************************************************************/
void itoa(int f){
	int x,dig;
	int var=100000000;
	for(x=0;x<9;x++){
		dig=f/var;
		if(dig==0){
			temporal[x]='0';
		}else if(dig==1){
			temporal[x]='1';
		}else if(dig==2){
			temporal[x]='2';
		}else if(dig==3){
			temporal[x]='3';
		}else if(dig==4){
			temporal[x]='4';
		}else if(dig==5){
			temporal[x]='5';
		}else if(dig==6){
			temporal[x]='6';
		}else if(dig==7){
			temporal[x]='7';
		}else if(dig==8){
			temporal[x]='8';
		}else if(dig==9){
			temporal[x]='9';
		}
		f=f%var;
		var=var/10;
	}
}

/*********************************************************************
 * Subrutina que indica si una caracter es un numero del 0 al 9
*********************************************************************/
int EsNumero(char tec){

	if(tec=='1' || tec=='2' || tec=='3' || tec=='4' || tec=='5'
	|| tec=='6' || tec=='7' || tec=='8' || tec=='9' || tec=='0')
		return 1;
	else
		return 0;
}

/*********************************************************************
 * Subrutina para cifrar clave
*********************************************************************/
void cifrar_clave(void){

	int arreglo_binario[36]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
```

```
int proceso1[9]={10,10,10,10,10,10,10,10,10};
int proceso2[9]={0,0,0,0,0,0,0,0,0};
int llave_binario[12]={0,0,0,0,0,0,0,0,0,0,0,0};
int x,dig,q,r,bandera=0;
int var=100000,digito_llave=0;
int clave_entero=0,desplazamiento_binario;
clave_entero=atoi(clave);

for(x=0;x<6;x++){
	dig=clave_entero/var;
	if(dig==0){
		clave_num[x]=0;
	}else if(dig==1){
		clave_num[x]=1;
	}else if(dig==2){
		clave_num[x]=2;
	}else if(dig==3){
		clave_num[x]=3;
	}else if(dig==4){
		clave_num[x]=4;
	}else if(dig==5){
		clave_num[x]=5;
	}else if(dig==6){
		clave_num[x]=6;
	}else if(dig==7){
		clave_num[x]=7;
	}else if(dig==8){
		clave_num[x]=8;
	}else if(dig==9){
		clave_num[x]=9;
	}
	clave_entero=clave_entero%var;
	var=var/10;
}

var=100;
int llave_entero=0;
llave_entero=atoi(llave_actual);
for(x=0;x<6;x++){
	dig=llave_entero/var;
	if(dig==0){
		llave_actual_num[x]=0;
	}else if(dig==1){
		llave_actual_num[x]=1;
	}else if(dig==2){
		llave_actual_num[x]=2;
	}else if(dig==3){
		llave_actual_num[x]=3;
	}else if(dig==4){
		llave_actual_num[x]=4;
	}else if(dig==5){
		llave_actual_num[x]=5;
	}else if(dig==6){
		llave_actual_num[x]=6;
	}else if(dig==7){
```

```
                llave_actual_num[x]=7;
        }else if(dig==8){
                llave_actual_num[x]=8;
        }else if(dig==9){
                llave_actual_num[x]=9;
        }
        llave_entero=llave_entero%var;
        var=var/10;
}
for(q=0;q<9;q++){
        for(r=0;r<3;r++){
                if(q==llave_actual_num[r]){
                        proceso1[q]=llave_actual_num[r];
                }
        }
}
for(q=0;q<6;q++){
        bandera=0;
        for(r=0;r<9;r++){
                if(proceso1[r]==10&&bandera==0){
                        proceso1[r]=clave_num[q];
                        bandera=1;
                }
        }
}
digito_llave=0;
for (q=0;q<12;q++){
        if(q==0){
                entero_binario(llave_actual_num[digito_llave]);
        }
        if(q==4||q==8){
                digito_llave++;
                entero_binario(llave_actual_num[digito_llave]);
        }
        if(q<4){
                llave_binario[q]=binario_llave[q];
        }
        else if(q>=4&&q<8){
                llave_binario[q]=binario_llave[q-4];
        }
        else if(q>=8){
                llave_binario[q]=binario_llave[q-8];
        }
}
desplazamiento_binario=(llave_binario[3]*100)+(llave_binario[7]
                        *10)+llave_binario[11];
desplazamiento_binario_entero(desplazamiento_binario);
desplazamiento++;

for (q=0;q<9;q++){
        if(desplazamiento==9){
                desplazamiento=0;
        }
        proceso2[q]=proceso1[desplazamiento];
        desplazamiento++;
```

```
}

digito_llave=0;
for(q=0;q<36;q++){
	if(q==0){
		entero_binario(proceso2[digito_llave]);
	}
	if(q==4||q==8||q==12||q==16||q==20||q==24||q==28||q==32){
		digito_llave++;
		entero_binario(proceso2[digito_llave]);
	}
	if(q<4){
		arreglo_binario[q]=binario_llave[q];
	}
	else if(q>=4&&q<8){
		arreglo_binario[q]=binario_llave[q-4];
	}
	else if(q>=8&&q<12){
		arreglo_binario[q]=binario_llave[q-8];
	}
	else if(q>=12&&q<16){
		arreglo_binario[q]=binario_llave[q-12];
	}
	else if(q>=16&&q<20){
		arreglo_binario[q]=binario_llave[q-16];
	}
	else if(q>=20&&q<24){
		arreglo_binario[q]=binario_llave[q-20];
	}
	else if(q>=24&&q<28){
		arreglo_binario[q]=binario_llave[q-24];
	}
	else if(q>=28&&q<32){
		arreglo_binario[q]=binario_llave[q-28];
	}
	else if(q>=32){
		arreglo_binario[q]=binario_llave[q-32];
	}
}
clave_cifrada_binario[0]=(arreglo_binario[3]*100)+(arreglo_binario
                          [7]*10)+arreglo_binario[11];
clave_cifrada_binario[1]=(arreglo_binario[2]*100)+(arreglo_binario
                          [6]*10)+arreglo_binario[10];
clave_cifrada_binario[2]=(arreglo_binario[1]*100)+(arreglo_binario
                          [5]*10)+arreglo_binario[9];
clave_cifrada_binario[3]=(arreglo_binario[0]*100)+(arreglo_binario
                          [4]*10)+arreglo_binario[8];
clave_cifrada_binario[4]=(arreglo_binario[15]*100)+(arreglo_binario
                          [19]*10)+arreglo_binario[23];
clave_cifrada_binario[5]=(arreglo_binario[14]*100)+(arreglo_binario
                          [18]*10)+arreglo_binario[22];
clave_cifrada_binario[6]=(arreglo_binario[13]*100)+(arreglo_binario
                          [17]*10)+arreglo_binario[21];
clave_cifrada_binario[7]=(arreglo_binario[12]*100)+(arreglo_binario
                          [16]*10)+arreglo_binario[20];
```

```
	clave_cifrada_binario[8]=(arreglo_binario[27]*100)+(arreglo_binario
				[31]*10)+arreglo_binario[35];
	clave_cifrada_binario[9]=(arreglo_binario[26]*100)+(arreglo_binario
				[30]*10)+arreglo_binario[34];
	clave_cifrada_binario[10]=(arreglo_binario[25]*100)+(arreglo_binario
				[29]*10)+arreglo_binario[33];
	clave_cifrada_binario[11]=(arreglo_binario[24]*100)+(arreglo_binario
				[28]*10)+arreglo_binario[32];

	clave_cifrada_entero();
	clave_cifrada_caracter();
}

/*********************************************************************
 * Subrutina para convertir de entero a caracter la clave cifrada
*********************************************************************/
void clave_cifrada_caracter(void){
	int w=0;

	for (w=0;w<12;w++){
		if(clave_cifrada[w]==0){
			clave_cifrada_car[w]='0';
		}
		else if(clave_cifrada[w]==1){
			clave_cifrada_car[w]='1';
		}
		else if(clave_cifrada[w]==2){
			clave_cifrada_car[w]='2';
		}
		else if(clave_cifrada[w]==3){
			clave_cifrada_car[w]='3';
		}
		else if(clave_cifrada[w]==4){
			clave_cifrada_car[w]='4';
		}
		else if(clave_cifrada[w]==5){
			clave_cifrada_car[w]='5';
		}
		else if(clave_cifrada[w]==6){
			clave_cifrada_car[w]='6';
		}
		else if(clave_cifrada[w]==7){
			clave_cifrada_car[w]='7';
		}
	}
}

/*********************************************************************
 * Subrutina para convertir de binario a decimal la clave cifrada
*********************************************************************/
void clave_cifrada_entero(void){
	int w=0;

	for (w=0;w<12;w++){
		if(clave_cifrada_binario[w]==0){
```

```
                        clave_cifrada[w]=0;
                }
                else if(clave_cifrada_binario[w]==1){
                        clave_cifrada[w]=1;
                }
                else if(clave_cifrada_binario[w]==10){
                        clave_cifrada[w]=2;
                }
                else if(clave_cifrada_binario[w]==11){
                        clave_cifrada[w]=3;
                }
                else if(clave_cifrada_binario[w]==100){
                        clave_cifrada[w]=4;
                }
                else if(clave_cifrada_binario[w]==101){
                        clave_cifrada[w]=5;
                }
                else if(clave_cifrada_binario[w]==110){
                        clave_cifrada[w]=6;
                }
                else if(clave_cifrada_binario[w]==111){
                        clave_cifrada[w]=7;
                }
        }
}

/*********************************************************************
 * Subrutina para determinar la cantidad de digitos a desplazar en
 * la clave cifrada
*********************************************************************/
void desplazamiento_binario_entero(int numero){
        if(numero==0){
                desplazamiento=0;
        }
        else if(numero==1){
                desplazamiento=1;
        }
        else if(numero==10){
                desplazamiento=2;
        }
        else if(numero==11){
                desplazamiento=3;
        }
        else if(numero==100){
                desplazamiento=4;
        }
        else if(numero==101){
                desplazamiento=5;
        }
        else if(numero==110){
                desplazamiento=6;
        }
        else{
                desplazamiento=7;
        }
```

```
}

/*********************************************************************
 * Subrutina para convertir un numero decimal a un numero binario
 * de 4 digitos en la clave cifrada
*********************************************************************/
void entero_binario(int numero){
	if(numero==0){
		binario_llave[0]=0;
		binario_llave[1]=0;
		binario_llave[2]=0;
		binario_llave[3]=0;
	}
	else if(numero==1){
		binario_llave[0]=0;
		binario_llave[1]=0;
		binario_llave[2]=0;
		binario_llave[3]=1;
	}
	else if(numero==2){
		binario_llave[0]=0;
		binario_llave[1]=0;
		binario_llave[2]=1;
		binario_llave[3]=0;
	}
	else if(numero==3){
		binario_llave[0]=0;
		binario_llave[1]=0;
		binario_llave[2]=1;
		binario_llave[3]=1;
	}
	else if(numero==4){
		binario_llave[0]=0;
		binario_llave[1]=1;
		binario_llave[2]=0;
		binario_llave[3]=0;
	}
	else if(numero==5){
		binario_llave[0]=0;
		binario_llave[1]=1;
		binario_llave[2]=0;
		binario_llave[3]=1;
	}
	else if(numero==6){
		binario_llave[0]=0;
		binario_llave[1]=1;
		binario_llave[2]=1;
		binario_llave[3]=0;
	}
	else if(numero==7){
		binario_llave[0]=0;
		binario_llave[1]=1;
		binario_llave[2]=1;
		binario_llave[3]=1;
	}
```

```
	else if(numero==8){
		binario_llave[0]=1;
		binario_llave[1]=0;
		binario_llave[2]=0;
		binario_llave[3]=0;
	}
}

/*********************************************************************
 * Subrutina para cifrar clave nueva
*********************************************************************/
void cifrar_clave_nueva(void){

	int arreglo_binario[36]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
	int proceso1[9]={10,10,10,10,10,10,10,10,10};
	int proceso2[9]={0,0,0,0,0,0,0,0,0};
	int llave_binario[12]={0,0,0,0,0,0,0,0,0,0,0,0};
	int x,dig,q,r,bandera=0;
	int var=100000,digito_llave=0;
	int clave_entero=0,desplazamiento_binario;
	clave_entero=atoi(clave_nueva);

	for(x=0;x<6;x++){
		dig=clave_entero/var;
		if(dig==0){
			clave_num_nuevo[x]=0;
		}else if(dig==1){
			clave_num_nuevo[x]=1;
		}else if(dig==2){
			clave_num_nuevo[x]=2;
		}else if(dig==3){
			clave_num_nuevo[x]=3;
		}else if(dig==4){
			clave_num_nuevo[x]=4;
		}else if(dig==5){
			clave_num_nuevo[x]=5;
		}else if(dig==6){
			clave_num_nuevo[x]=6;
		}else if(dig==7){
			clave_num_nuevo[x]=7;
		}else if(dig==8){
			clave_num_nuevo[x]=8;
		}else if(dig==9){
			clave_num_nuevo[x]=9;
		}
		clave_entero=clave_entero%var;
		var=var/10;
	}

	var=100;
	int llave_entero=0;
	llave_entero=atoi(llave_actual);
	for(x=0;x<6;x++){
```

```
        dig=llave_entero/var;
        if(dig==0){
                llave_actual_num_nuevo[x]=0;
        }else if(dig==1){
                llave_actual_num_nuevo[x]=1;
        }else if(dig==2){
                llave_actual_num_nuevo[x]=2;
        }else if(dig==3){
                llave_actual_num_nuevo[x]=3;
        }else if(dig==4){
                llave_actual_num_nuevo[x]=4;
        }else if(dig==5){
                llave_actual_num_nuevo[x]=5;
        }else if(dig==6){
                llave_actual_num_nuevo[x]=6;
        }else if(dig==7){
                llave_actual_num_nuevo[x]=7;
        }else if(dig==8){
                llave_actual_num_nuevo[x]=8;
        }else if(dig==9){
                llave_actual_num_nuevo[x]=9;
        }
        llave_entero=llave_entero%var;
        var=var/10;
}
for(q=0;q<9;q++){
        for(r=0;r<3;r++){
                if(q==llave_actual_num_nuevo[r]){
                        proceso1[q]=llave_actual_num_nuevo[r];
                }
        }
}
for(q=0;q<6;q++){
        bandera=0;
        for(r=0;r<9;r++){
                if(proceso1[r]==10&&bandera==0){
                        proceso1[r]=clave_num_nuevo[q];
                        bandera=1;
                }
        }
}
digito_llave=0;
for (q=0;q<12;q++){
        if(q==0){
                entero_binario_nuevo(llave_actual_num_nuevo[digito_llave]);
        }
        if(q==4||q==8){
                digito_llave++;
                entero_binario_nuevo(llave_actual_num_nuevo[digito_llave]);
        }
        if(q<4){
                llave_binario[q]=binario_llave_nuevo[q];
        }
        else if(q>=4&&q<8){
                llave_binario[q]=binario_llave_nuevo[q-4];
```

```
        }
        else if(q>=8){
                llave_binario[q]=binario_llave_nuevo[q-8];
        }
}
desplazamiento_binario=(llave_binario[3]*100)+(llave_binario[7]*10)
                                        +llave_binario[11];
desplazamiento_binario_entero_nuevo(desplazamiento_binario);
desplazamiento_nuevo++;

for (q=0;q<9;q++){
        if(desplazamiento_nuevo==9){
                desplazamiento_nuevo=0;
        }
        proceso2[q]=proceso1[desplazamiento_nuevo];
        desplazamiento_nuevo++;
}

digito_llave=0;
for(q=0;q<36;q++){
        if(q==0){
                entero_binario_nuevo(proceso2[digito_llave]);
        }
        if(q==4||q==8||q==12||q==16||q==20||q==24||q==28||q==32){
                digito_llave++;
                entero_binario_nuevo(proceso2[digito_llave]);
        }
        if(q<4){
                arreglo_binario[q]=binario_llave_nuevo[q];
        }
        else if(q>=4&&q<8){
                arreglo_binario[q]=binario_llave_nuevo[q-4];
        }
        else if(q>=8&&q<12){
                arreglo_binario[q]=binario_llave_nuevo[q-8];
        }
        else if(q>=12&&q<16){
                arreglo_binario[q]=binario_llave_nuevo[q-12];
        }
        else if(q>=16&&q<20){
                arreglo_binario[q]=binario_llave_nuevo[q-16];
        }
        else if(q>=20&&q<24){
                arreglo_binario[q]=binario_llave_nuevo[q-20];
        }
        else if(q>=24&&q<28){
                arreglo_binario[q]=binario_llave_nuevo[q-24];
        }
        else if(q>=28&&q<32){
                arreglo_binario[q]=binario_llave_nuevo[q-28];
        }
        else if(q>=32){
                arreglo_binario[q]=binario_llave_nuevo[q-32];
        }
}
```

```
	clave_cifrada_binario_nuevo[0]=(arreglo_binario[3]*100)+(
				arreglo_binario[7]*10)+arreglo_binario[11];

	clave_cifrada_binario_nuevo[1]=(arreglo_binario[2]*100)+(
				arreglo_binario[6]*10)+arreglo_binario[10];

	clave_cifrada_binario_nuevo[2]=(arreglo_binario[1]*100)+(
				arreglo_binario[5]*10)+arreglo_binario[9];

	clave_cifrada_binario_nuevo[3]=(arreglo_binario[0]*100)+(
				arreglo_binario[4]*10)+arreglo_binario[8];

	clave_cifrada_binario_nuevo[4]=(arreglo_binario[15]*100)+(
				arreglo_binario[19]*10)+arreglo_binario[23];

	clave_cifrada_binario_nuevo[5]=(arreglo_binario[14]*100)+(
				arreglo_binario[18]*10)+arreglo_binario[22];

	clave_cifrada_binario_nuevo[6]=(arreglo_binario[13]*100)+(
				arreglo_binario[17]*10)+arreglo_binario[21];

	clave_cifrada_binario_nuevo[7]=(arreglo_binario[12]*100)+(
				arreglo_binario[16]*10)+arreglo_binario[20];

	clave_cifrada_binario_nuevo[8]=(arreglo_binario[27]*100)+(
				arreglo_binario[31]*10)+arreglo_binario[35];

	clave_cifrada_binario_nuevo[9]=(arreglo_binario[26]*100)+(
				arreglo_binario[30]*10)+arreglo_binario[34];

	clave_cifrada_binario_nuevo[10]=(arreglo_binario[25]*100)+(
				arreglo_binario[29]*10)+arreglo_binario[33];

	clave_cifrada_binario_nuevo[11]=(arreglo_binario[24]*100)+(
				arreglo_binario[28]*10)+arreglo_binario[32];

	clave_cifrada_entero_nuevo();
	clave_cifrada_caracter_nuevo();
}

/*********************************************************************
 * Subrutina para convertir de entero a caracter la clave
 * nueva cifrada
*********************************************************************/
void clave_cifrada_caracter_nuevo(void){
	int w=0;

	for (w=0;w<12;w++){
		if(clave_cifrada_nuevo[w]==0){
			clave_cifrada_car_nuevo[w]='0';
		}
		else if(clave_cifrada_nuevo[w]==1){
			clave_cifrada_car_nuevo[w]='1';
		}
```

```
                else if(clave_cifrada_nuevo[w]==2){
                        clave_cifrada_car_nuevo[w]='2';
                }
                else if(clave_cifrada_nuevo[w]==3){
                        clave_cifrada_car_nuevo[w]='3';
                }
                else if(clave_cifrada_nuevo[w]==4){
                        clave_cifrada_car_nuevo[w]='4';
                }
                else if(clave_cifrada_nuevo[w]==5){
                        clave_cifrada_car_nuevo[w]='5';
                }
                else if(clave_cifrada_nuevo[w]==6){
                        clave_cifrada_car_nuevo[w]='6';
                }
                else if(clave_cifrada_nuevo[w]==7){
                        clave_cifrada_car_nuevo[w]='7';
                }
        }

}

/*********************************************************************
 * Subrutina para convertir de binario a decimal la clave
 * nueva cifrada
*********************************************************************/
void clave_cifrada_entero_nuevo(void){
        int w=0;

        for (w=0;w<12;w++){
                if(clave_cifrada_binario_nuevo[w]==0){
                        clave_cifrada_nuevo[w]=0;
                }
                else if(clave_cifrada_binario_nuevo[w]==1){
                        clave_cifrada_nuevo[w]=1;
                }
                else if(clave_cifrada_binario_nuevo[w]==10){
                        clave_cifrada_nuevo[w]=2;
                }
                else if(clave_cifrada_binario_nuevo[w]==11){
                        clave_cifrada_nuevo[w]=3;
                }
                else if(clave_cifrada_binario_nuevo[w]==100){
                        clave_cifrada_nuevo[w]=4;
                }
                else if(clave_cifrada_binario_nuevo[w]==101){
                        clave_cifrada_nuevo[w]=5;
                }
                else if(clave_cifrada_binario_nuevo[w]==110){
                        clave_cifrada_nuevo[w]=6;
                }
                else if(clave_cifrada_binario_nuevo[w]==111){
                        clave_cifrada_nuevo[w]=7;
                }
        }
```

```
}

/*********************************************************************
 * Subrutina para convertir un numero decimal a un numero binario
 * de 4 digitos en la clave nueva cifrada
*********************************************************************/
void entero_binario_nuevo(int numero){
	if(numero==0){
		binario_llave_nuevo[0]=0;
		binario_llave_nuevo[1]=0;
		binario_llave_nuevo[2]=0;
		binario_llave_nuevo[3]=0;
	}
	else if(numero==1){
		binario_llave_nuevo[0]=0;
		binario_llave_nuevo[1]=0;
		binario_llave_nuevo[2]=0;
		binario_llave_nuevo[3]=1;
	}
	else if(numero==2){
		binario_llave_nuevo[0]=0;
		binario_llave_nuevo[1]=0;
		binario_llave_nuevo[2]=1;
		binario_llave_nuevo[3]=0;
	}
	else if(numero==3){
		binario_llave_nuevo[0]=0;
		binario_llave_nuevo[1]=0;
		binario_llave_nuevo[2]=1;
		binario_llave_nuevo[3]=1;
	}
	else if(numero==4){
		binario_llave_nuevo[0]=0;
		binario_llave_nuevo[1]=1;
		binario_llave_nuevo[2]=0;
		binario_llave_nuevo[3]=0;
	}
	else if(numero==5){
		binario_llave_nuevo[0]=0;
		binario_llave_nuevo[1]=1;
		binario_llave_nuevo[2]=0;
		binario_llave_nuevo[3]=1;
	}
	else if(numero==6){
		binario_llave_nuevo[0]=0;
		binario_llave_nuevo[1]=1;
		binario_llave_nuevo[2]=1;
		binario_llave_nuevo[3]=0;
	}
	else if(numero==7){
		binario_llave_nuevo[0]=0;
		binario_llave_nuevo[1]=1;
		binario_llave_nuevo[2]=1;
		binario_llave_nuevo[3]=1;
	}
```

```
	else if(numero==8){
		binario_llave_nuevo[0]=1;
		binario_llave_nuevo[1]=0;
		binario_llave_nuevo[2]=0;
		binario_llave_nuevo[3]=0;
	}
}

/*********************************************************************
 * Subrutina para determinar la cantidad de digitos a desplazar en
 * la clave nueva cifrada
*********************************************************************/
void desplazamiento_binario_entero_nuevo(int numero){
	if(numero==0){
		desplazamiento_nuevo=0;
	}
	else if(numero==1){
		desplazamiento_nuevo=1;
	}
	else if(numero==10){
		desplazamiento_nuevo=2;
	}
	else if(numero==11){
		desplazamiento_nuevo=3;
	}
	else if(numero==100){
		desplazamiento_nuevo=4;
	}
	else if(numero==101){
		desplazamiento_nuevo=5;
	}
	else if(numero==110){
		desplazamiento_nuevo=6;
	}
	else{
		desplazamiento_nuevo=7;
	}
}

/*********************************************************************
 * Subrutina para leer una tecla en el teclado matricial
*********************************************************************/
char leer_tecla(void){
	int i=0,j=0,valor;
	volatile int *p_gpio_in=gpio_in;
	volatile int *p_gpio_out=gpio_out;
	unsigned char fila[]={0x07,0x0b,0x0d,0x0e};
	char tecla='w';

	while(1){
		*p_gpio_out=fila[i];
		if(*p_gpio_in != 15){
			switch (*p_gpio_in)
			{
				case 7:
```

```
switch(i)
{
	case 0:
		tecla='1'; j=1;
		break;
	case 1:
		tecla='4';j=1;
		break;
	case 2:
		tecla='7';j=1;
		break;
	case 3:
		tecla='*';j=1;
		break;
	default:
		break;
}
break;
case 11:
switch(i)
{
	case 0:
		tecla='2';j=1;
		break;
	case 1:
		tecla='5';j=1;
		break;
	case 2:
		tecla='8';j=1;
		break;
	case 3:
		tecla='0';j=1;
		break;
	default:
		break;
}
break;
case 13:
switch(i)
{
	case 0:
		tecla='3';j=1;
		break;
	case 1:
		tecla='6';j=1;
		break;
	case 2:
		tecla='9';j=1;
		break;
	case 3:
		tecla='#';j=1;
		break;
	default:
		break;
}
```

```
                                break;
                                case 14:
                                switch(i)
                                {
                                        case 0:
                                                tecla='a';j=1;
                                                break;
                                        case 1:
                                                tecla='b';j=1;
                                                break;
                                        case 2:
                                                tecla='c';j=1;
                                                break;
                                        case 3:
                                                tecla='d';j=1;
                                                break;
                                        default:
                                                break;
                                }
                                break;
                                default:
                                        case 15:
                                        }
                        usleep(300000);
                        }
                        i=i+1;
                        if(i==4){i=0;}
                        if(j==1){break;}
                }
        return tecla;
}

/*********************************************************************
 * Subrutina para ingresar la direccion IP de la Tarjeta DE2
*********************************************************************/
void ingreso_direccion_ip_tarjeta(){
        int dig=0,x=0,p=0,j=0,siguiente=0,m=0,puntos=0;
        char tecla='q';

        char direccionip_user_int[16]="qqqqqqqqqqqqqqqq";
        char octeto_int1[4]="qqqq";
        char octeto_int2[4]="qqqq";
        char octeto_int3[4]="qqqq";
        char octeto_int4[4]="qqqq";

        char direccionip_user[16]=" ";
        char octeto1[4]=" ";
        char octeto2[4]=" ";
        char octeto3[4]=" ";
        char octeto4[4]=" ";

        LCD_clear();
        LCD_cursor (0,0);
        LCD_text ("Dir.IP(Tarjeta):");
```

```
while(1){
	tecla=leer_tecla();
	if(EsNumero(tecla) && dig<15){
		direccionip_user_int[dig]=tecla;
		dig=dig+1;
		for(p=0;p<15;p++){
			if(direccionip_user_int[p]!='q'){
				direccionip_user[p]=direccionip_user_int[p];
			}
		}
		//Mostrar en pantalla
		LCD_cursor (0,1);
		LCD_text (direccionip_user);
	}else if(tecla=='#'){
		// Mostrar Punto
		puntos++;
		direccionip_user_int[dig]='.';
		dig=dig+1;
		for(p=0;p<15;p++){
			if(direccionip_user_int[p]!='q'){
				direccionip_user[p]=direccionip_user_int[p];
			}
		}
		LCD_cursor (0,1);
		LCD_text (direccionip_user);
	}
	else if(tecla=='a'){
		// Borrar ultimo digito
		if(dig>0){
			if(direccionip_user_int[dig-1]=='.'){
				puntos--;
			}
			direccionip_user_int[dig-1]='q';
			direccionip_user[dig-1]=' ';
			dig=dig-1;
			LCD_cursor (0,1);
			LCD_text (direccionip_user);
		}
	}else if(tecla=='b'&& (puntos<3||puntos>3)){
		LCD_clear();
		LCD_cursor (0,0);
		LCD_text ("  Estructura IP ");
		LCD_cursor (0,1);
		LCD_text ("   Incorrecta   ");
		usleep(2000000);
		LCD_clear();
		LCD_cursor (0,0);
		LCD_text ("Dir.IP(Tarjeta):");
		LCD_cursor (0,1);
		LCD_text (direccionip_user);
	}else if(tecla=='b'&& puntos==3){
		if(dig>0)
			x=10;
	}
	if(x==10)
```

```
        break;
}
m=0;
j=0;
while(j<15){
        while(siguiente==0){
                octeto_int1[m]=direccionip_user[j];
                for (p=0;p<3;p++){
                        if(octeto_int1[p]!='q'){
                                octeto1[p]=octeto_int1[p];
                        }
                }
                if(direccionip_user[j]=='.')
                        siguiente=1;
                j++;
                m++;
        }
        siguiente=0;
        m=0;
        while(siguiente==0){
                octeto_int2[m]=direccionip_user[j];
                for (p=0;p<3;p++){
                        if(octeto_int2[p]!='q'){
                                octeto2[p]=octeto_int2[p];
                        }
                }
                if(direccionip_user[j]=='.')
                        siguiente=1;
                j++;
                m++;
        }
        siguiente=0;
        m=0;
        while(siguiente==0){
                octeto_int3[m]=direccionip_user[j];
                for (p=0;p<3;p++){
                        if(octeto_int3[p]!='q'){
                                octeto3[p]=octeto_int3[p];
                        }
                }
                if(direccionip_user[j]=='.')
                        siguiente=1;
                j++;
                m++;
        }
        siguiente=0;
        m=0;
        while(siguiente==0){
                octeto_int4[m]=direccionip_user[j];
                for (p=0;p<3;p++){
                        if(octeto_int4[p]!='q'){
                                octeto4[p]=octeto_int4[p];
                        }
                }
                if(direccionip_user[j]=='.')
```

```
                        siguiente=1;
                j++;
                m++;
        }

        for (p=0;p<3;p++){
                if(octeto1[p]=='.'){
                        octeto1[p]=' ';
                }
        }

        for (p=0;p<3;p++){
                if(octeto2[p]=='.'){
                        octeto2[p]=' ';
                }
        }

        for (p=0;p<3;p++){
                if(octeto3[p]=='.'){
                        octeto3[p]=' ';
                }
        }

        for (p=0;p<3;p++){
                if(octeto4[p]=='.'){
                        octeto4[p]=' ';
                }
        }
        printf(" %s",octeto1);
        printf(".%s",octeto2);
        printf(".%s",octeto3);
        printf(".%s\n",octeto4);
    }
    transmit_buffer[26]=atoi(octeto1);
    transmit_buffer[27]=atoi(octeto2);
    transmit_buffer[28]=atoi(octeto3);
    transmit_buffer[29]=atoi(octeto4);

    arp_buffer[28]=atoi(octeto1);
    arp_buffer[29]=atoi(octeto2);
    arp_buffer[30]=atoi(octeto3);
    arp_buffer[31]=atoi(octeto4);
}

/*********************************************************************
 * Subrutina para ingresar la direccion IP del Servidor
*********************************************************************/
void ingreso_direccion_ip_servidor(){

    int dig=0,x=0,p=0,j=0,siguiente=0,m=0,puntos=0;
    char tecla='q';

    char direccionip_servidor_int[16]="qqqqqqqqqqqqqqqq";
    char octeto_int1[4]="qqqq";
    char octeto_int2[4]="qqqq";
```

```
char octeto_int3[4]="qqqq";
char octeto_int4[4]="qqqq";

char direccionip_servidor[16]=" ";
char octeto1[4]=" ";
char octeto2[4]=" ";
char octeto3[4]=" ";
char octeto4[4]=" ";

LCD_clear();
LCD_cursor (0,0);
LCD_text ("Dir.IP(Servidor)");

while(1){
	tecla=leer_tecla();
	if(EsNumero(tecla) && dig<15){
		direccionip_servidor_int[dig]=tecla;
		dig=dig+1;
		for(p=0;p<15;p++){
			if(direccionip_servidor_int[p]!='q'){

direccionip_servidor[p]=direccionip_servidor_int[p];
			}
		}
		//Mostrar en pantalla
		LCD_cursor (0,1);
		LCD_text (direccionip_servidor);
	}else if(tecla=='#'){
		// Mostrar Punto
		puntos++;
		direccionip_servidor_int[dig]='.';
		dig=dig+1;
		for(p=0;p<15;p++){
			if(direccionip_servidor_int[p]!='q'){

direccionip_servidor[p]=direccionip_servidor_int[p];
			}
		}
		LCD_cursor (0,1);
		LCD_text (direccionip_servidor);
	}
	else if(tecla=='a'){
		// Borrar ultimo digito
		if(dig>0){
			if(direccionip_servidor_int[dig-1]=='.'){
				puntos--;
			}
			direccionip_servidor_int[dig-1]='q';
			direccionip_servidor[dig-1]=' ';
			dig=dig-1;
			LCD_cursor (0,1);
			LCD_text (direccionip_servidor);
		}
	}else if(tecla=='b'&& (puntos<3||puntos>3)){
		LCD_clear();
```

```
                LCD_cursor (0,0);
                LCD_text ("  Estructura IP ");
                LCD_cursor (0,1);
                LCD_text ("   Incorrecta   ");
                usleep(1000000);
                LCD_clear();
                LCD_cursor (0,0);
                LCD_text ("Dir.IP(Servidor)");
                LCD_cursor (0,1);
                LCD_text (direccionip_servidor);
        }else if(tecla=='b'&& puntos==3){
                if(dig>0)
                        x=10;
        }
        if(x==10)
        break;
}
m=0;
j=0;
while(j<15){
        while(siguiente==0){
                octeto_int1[m]=direccionip_servidor[j];
                for (p=0;p<3;p++){
                        if(octeto_int1[p]!='q'){
                                octeto1[p]=octeto_int1[p];
                        }
                }
                if(direccionip_servidor[j]=='.')
                        siguiente=1;
                j++;
                m++;
        }
        siguiente=0;
        m=0;
        while(siguiente==0){
                octeto_int2[m]=direccionip_servidor[j];
                for (p=0;p<3;p++){
                        if(octeto_int2[p]!='q'){
                                octeto2[p]=octeto_int2[p];
                        }
                }
                if(direccionip_servidor[j]=='.')
                        siguiente=1;
                j++;
                m++;
        }
        siguiente=0;
        m=0;
        while(siguiente==0){
                octeto_int3[m]=direccionip_servidor[j];
                for (p=0;p<3;p++){
                        if(octeto_int3[p]!='q'){
                                octeto3[p]=octeto_int3[p];
                        }
                }
```

```
                if(direccionip_servidor[j]=='.')
                        siguiente=1;
                j++;
                m++;
        }
        siguiente=0;
        m=0;
        while(siguiente==0){
                octeto_int4[m]=direccionip_servidor[j];
                for (p=0;p<3;p++){
                        if(octeto_int4[p]!='q'){
                                octeto4[p]=octeto_int4[p];
                        }
                }
                if(direccionip_servidor[j]=='.')
                        siguiente=1;
                j++;
                m++;
        }

        for (p=0;p<3;p++){
                if(octeto1[p]=='.'){
                        octeto1[p]=' ';
                }
        }

        for (p=0;p<3;p++){
                if(octeto2[p]=='.'){
                        octeto2[p]=' ';
                }
        }

        for (p=0;p<3;p++){
                if(octeto3[p]=='.'){
                        octeto3[p]=' ';
                }
        }

        for (p=0;p<3;p++){
                if(octeto4[p]=='.'){
                        octeto4[p]=' ';
                }
        }
        printf("%s",octeto1);
        printf(".%s",octeto2);
        printf(".%s",octeto3);
        printf(".%s",octeto4);
    }
    transmit_buffer[30]=atoi(octeto1);
    transmit_buffer[31]=atoi(octeto2);
    transmit_buffer[32]=atoi(octeto3);
    transmit_buffer[33]=atoi(octeto4);
}
```

```
/*********************************************************************
 * Subrutina para generar una llave nueva
*********************************************************************/
void generar_llave_nueva(){
	int a=0,b=0,numero,repetido=0;
	char numero_generado;
	srand((unsigned)time(NULL));
	llave_actual[0]=10;
	llave_actual[1]=10;
	llave_actual[2]=10;

	for(a=0;a<3;a++){
		do{
			repetido=0;
			numero=rand()%9;
			if(numero==0){
				numero_generado='0';
			}else if(numero==1){
				numero_generado='1';
			}else if(numero==2){
				numero_generado='2';
			}else if(numero==3){
				numero_generado='3';
			}else if(numero==4){
				numero_generado='4';
			}else if(numero==5){
				numero_generado='5';
			}else if(numero==6){
				numero_generado='6';
			}else if(numero==7){
				numero_generado='7';
			}else{
				numero_generado='8';
			}
			for(b=0;b<3;b++){
				if(llave_actual[b]==numero_generado){
					repetido=1;
				}
			}
		}while(repetido==1);

		if(numero==0){
			llave_actual[a]='0';
		}else if(numero==1){
			llave_actual[a]='1';
		}else if(numero==2){
			llave_actual[a]='2';
		}else if(numero==3){
			llave_actual[a]='3';
		}else if(numero==4){
			llave_actual[a]='4';
		}else if(numero==5){
			llave_actual[a]='5';
		}else if(numero==6){
			llave_actual[a]='6';
```

```
                }else if(numero==7){
                        llave_actual[a]='7';
                }else if(numero==8){
                        llave_actual[a]='8';
                }else if(numero==9){
                        llave_actual[a]='9';
                }
        }
        printf("\nLlave generada: %s",llave_actual);
}


/*********************************************************************
 * Subrutina para leer una tecla en el teclado matricial, probando
 * Conexion con el servidor cada 80 segundos
*********************************************************************/
char leer_tecla_tiempo(void){
        int i=0,j=0,valor,bandera=0,prueba=0;
        volatile int *p_gpio_in=gpio_in;
        volatile int *p_gpio_out=gpio_out;
        volatile int *p_pushbutton=pushbutton;
        volatile int *p_green_leds=green_leds;
        unsigned char fila[]={0x07,0x0b,0x0d,0x0e};
        char tecla='w';
        long pb_changes;

        (*p_green_leds)=0x00;
        while(1){
                contador_verificacion_conexion++;
                *p_gpio_out=fila[i];
                if(*p_gpio_in != 15){
                        switch (*p_gpio_in)
                        {
                                case 7:
                                switch(i)
                                {
                                        case 0:
                                                tecla='1'; j=1;
                                                break;
                                        case 1:
                                                tecla='4';j=1;
                                                break;
                                        case 2:
                                                tecla='7';j=1;
                                                break;
                                        case 3:
                                                tecla='*';j=1;
                                                break;
                                        default:
                                                break;
                                }
                                break;
                                case 11:
                                switch(i)
                                {
```

```
        case 0:
                tecla='2';j=1;
                break;
        case 1:
                tecla='5';j=1;
                break;
        case 2:
                tecla='8';j=1;
                break;
        case 3:
                tecla='0';j=1;
                break;
        default:
                break;
}
break;
case 13:
switch(i)
{
        case 0:
                tecla='3';j=1;
                break;
        case 1:
                tecla='6';j=1;
                break;
        case 2:
                tecla='9';j=1;
                break;
        case 3:
                tecla='#';j=1;
                break;
        default:
                break;
}
break;
case 14:
switch(i)
{
        case 0:
                tecla='a';j=1;
                break;
        case 1:
                tecla='b';j=1;
                break;
        case 2:
                tecla='c';j=1;
                break;
        case 3:
                tecla='d';j=1;
                break;
        default:
                break;
}
break;
default:
```

```
                break;
        }
usleep(300000);
}
i=i+1;
if(i==4){i=0;}
if(j==1){break;}
*p_pushbutton = 0;
pb_changes = *p_pushbutton;
if(pb_changes==2){
        (*p_green_leds)=0x01;
        *p_pushbutton = 0;
        usleep(4000000);
        contador_verificacion_conexion=
        contador_verificacion_conexion+800000;
}
(*p_green_leds)=0x00;
if(contador_verificacion_conexion>=16000000){
        estado_sistema=0;
        printf("\n\nVerificando conexion...");
        while(estado_sistema!=5){
                if(saltodes==1){
                        LCD_clear();
                        LCD_cursor (0,0);
                        LCD_text ("  Desconexion  ");
                        LCD_cursor (0,1);
                        LCD_text ("   Servidor    ");
                }
                if(bandera<1 && saltodes==0){
                        LCD_clear();
                        LCD_cursor (0,0);
                        LCD_text ("  Verificando   ");
                        LCD_cursor (0,1);
                        LCD_text ("   Conexion...  ");
                }

                Inicializacion_DM9000();
                usleep(2000000);
                if (!prueba_enlace_dm9000a())
                  {
                        printf("\n  Problema: Cable de red
                                desconectado");
                        LCD_clear();
                        LCD_cursor (0,0);
                        LCD_text ("  Cable de red ");
                        LCD_cursor (0,1);
                        LCD_text ("  Desconectado ");
                        LCD_cursor_off ();
                        prueba=0;
                                while(prueba==0){
                                  usleep(2000000);
                                  if (prueba_enlace_dm9000a())
                                  {
                                        prueba=1;
                                  }
```

```
                Inicializacion_DM9000();
            }
        usleep(2000000);
    }

Enviar_Mensaje("test");
printf("\n  Mensaje TEST enviado al
servidor");
usleep(1000000);

if(estado_sistema==5){
        contador_verificacion_conexion=0;
        Enviar_Mensaje("establecida");
        printf("  Mensaje ESTABLECIDA enviado
                    al servidor");
        printf("\n  Tarjeta DE2 conectada con el
                    servidor\n");
        printf("\nLlave actual:
        %s\n",llave_actual);
        printf("\nIngrese datos:\n");
        saltodes=0;
        if(estado_ingreso==1){
                LCD_clear();
                LCD_cursor (0,0);
                LCD_text ("Ingrese Usuario:");
                j=1;
        }else if(estado_ingreso==2){
                LCD_clear();
                LCD_cursor (0,0);
                LCD_text ("Ingrese clave: ");
                j=1;
        }else if(estado_ingreso==3){
                LCD_clear();
                LCD_cursor (0,0);
                LCD_text ("Clave anterior: ");
                j=1;
        }else if(estado_ingreso==4){
                LCD_clear();
                LCD_cursor (0,0);
                LCD_text ("Clave nueva: ");
                j=1;
        }else if(estado_ingreso==5){
                LCD_clear();
                LCD_cursor (0,0);
                LCD_text ("Confirme clave: ");
                j=1;
        }
        j=1;
        bandera=-1;
}
if(bandera==0){
    printf("\n  Error: Tiempo de espera para
            recibir una respuesta del servidor
            agotada (1 segundo)");
}
```

```
                                        if(bandera>=1){
                                               if(bandera==1){
                                                  printf("\n  Problema: Servidor
                                                           inalcanzable");
                                               }
                                               LCD_clear();
                                               LCD_cursor (0,0);
                                               LCD_text ("  Desconexion  ");
                                               LCD_cursor (0,1);
                                               LCD_text ("   Servidor    ");
                                        }
                                        bandera++;
                                 }
                          }
                   }
            return tecla;
}

/*********************************************************************
 * Subrutina para ingresar los datos del usuario
*********************************************************************/
void ingreso_usuario_clave(){
       iow(IMR, PAR_set);
       int dig=0,x=0,p=0,f=0, cambio_clave=0,contador_intentos_cambio=0,
              c=0;
       char tecla='q';

       temporal[9]="qqqqqqqqq";
       estado_sistema=0;

       // cadena de caracteres donde se almacenaran los digitos
       // ingresados que se manejaran en el programa
       char clave_int[7]="qqqqqqq";
       char clave_ext[7]=" ";
       char clave_cif[7]=" ";

       char clave_nueva_int[7]="qqqqqqq";
       char clave_nueva_ext[7]=" ";
       char clave_nueva_cif[7]=" ";

       char clave_nueva_confirmacion_int[7]="qqqqqqq";
       char clave_nueva_confirmacion_ext[7]=" ";
       char clave_nueva_confirmacion_cif[7]=" ";

       char usuario_int[4]="qqqq";
       char usuario_ext[4]=" ";

       estado_ingreso=0;
       if(salto==0){
       LCD_clear();
       LCD_cursor (0,0);
       LCD_text ("Ingrese Usuario:");
       while (1){
              LCD_cursor (0,1);
              LCD_text (usuario_ext);
```

```
        estado_ingreso=1;
        tecla=leer_tecla_tiempo();
        // solo se pueden ingresar hasta 4 digitos
        if(EsNumero(tecla) && dig<3){
                usuario_int[dig]=tecla;
                dig=dig+1;
                for (p=0;p<3;p++){
                        if(usuario_int[p]!='q')
                                usuario_ext[p]=usuario_int[p];
                }
                //Mostrar numero en pantalla
                LCD_cursor (0,1);
                LCD_text (usuario_ext);
        }else if(tecla=='a'){
                // Borrar ultimo digito
                if(dig>0){
                        usuario_int[dig-1]='q';
                        usuario_ext[dig-1]=' ';
                        dig=dig-1;
                        LCD_cursor (0,1);
                        LCD_text (usuario_ext);
                }
        }else if(tecla=='c' && dig==3){
                if(dig>0)
                        x=11;
        }else if(tecla=='b' && dig==3){
                if(dig>0)
                        x=10;
        }

        if(x==10)
                break;
        if(x==11){
                cambio_clave=1;
                break;
        }
}

usuario[0]=usuario_ext[0];
usuario[1]=usuario_ext[1];
usuario[2]=usuario_ext[2];
printf("  ID Usuario:");
for (c=0; c<3; c++){
   printf("%c",usuario[c]);
  }
}
if(salto==1){
        cambio_clave=1;
}
if(cambio_clave==0){

        dig=0; x=0; p=0;
        LCD_clear();
        LCD_cursor (0,0);
        LCD_text ("Ingrese clave: ");
```

```
while (1){
       LCD_cursor (0,1);
       LCD_text (clave_cif);
       estado_ingreso=2;
       tecla=leer_tecla_tiempo();
       // solo se pueden ingresar hasta 5 digitos
       if(EsNumero(tecla) && dig<6){
              clave_int[dig]=tecla;
              dig=dig+1;
              for (p=0;p<6;p++){
                     if(clave_int[p]!='q'){
                            clave_ext[p]=clave_int[p];
                            clave_cif[p]='*';
                     }
              }
              //Mostrar numero en pantalla
              LCD_cursor (0,1);
              LCD_text (clave_cif);
       }else if(tecla=='a'){
              // Borrar ultimo digito
              if(dig>0){
                     clave_int[dig-1]='q';
                     clave_cif[dig-1]=' ';
                     clave_ext[dig-1]=' ';
                     dig=dig-1;
                     LCD_cursor (0,1);
                     LCD_text (clave_cif);
              }
       }else if(tecla=='b'){
              if(dig>0)
                     x=10;
       }
       if(x==10)
       break;
}

clave[0]=clave_ext[0];
clave[1]=clave_ext[1];
clave[2]=clave_ext[2];
clave[3]=clave_ext[3];
clave[4]=clave_ext[4];
clave[5]=clave_ext[5];

printf("\n  Clave:");
for (c=0; c<6; c++){
   printf("%c",clave[c]);
}

cifrar_clave();

f=0;
f=(atoi(usuario)*1000000)+(atoi(clave));
itoa(f);

mensaje_enviar[0]=temporal[0];
```

```
        mensaje_enviar[1]=temporal[1];
        mensaje_enviar[2]=temporal[2];
        mensaje_enviar[3]=',';
        mensaje_enviar[4]=clave_cifrada_car[0];
        mensaje_enviar[5]=clave_cifrada_car[1];
        mensaje_enviar[6]=clave_cifrada_car[2];
        mensaje_enviar[7]=clave_cifrada_car[3];
        mensaje_enviar[8]=clave_cifrada_car[4];
        mensaje_enviar[9]=clave_cifrada_car[5];
        mensaje_enviar[10]=clave_cifrada_car[6];
        mensaje_enviar[11]=clave_cifrada_car[7];
        mensaje_enviar[12]=clave_cifrada_car[8];
        mensaje_enviar[13]=clave_cifrada_car[9];
        mensaje_enviar[14]=clave_cifrada_car[10];
        mensaje_enviar[15]=clave_cifrada_car[11];

        printf("\n  Clave cifrada:");
        for (c=0; c<12; c++){
           printf("%c",clave_cifrada_car[c]);
          }

        salir=0;

}else if(cambio_clave==1){

        if(salto==0){
        dig=0; x=0; p=0;
        LCD_clear();
        LCD_cursor (0,0);
        LCD_text ("Clave anterior: ");

        while (1){
               LCD_cursor (0,1);
               LCD_text (clave_cif);
               estado_ingreso=3;
               tecla=leer_tecla_tiempo();
               // solo se pueden ingresar hasta 5 digitos
               if(EsNumero(tecla) && dig<6){
                      clave_int[dig]=tecla;
                      dig=dig+1;
                      for (p=0;p<6;p++){
                             if(clave_int[p]!='q'){
                                    clave_ext[p]=clave_int[p];
                                    clave_cif[p]='*';
                             }
                      }
                      //Mostrar numero en pantalla
                      LCD_cursor (0,1);
                      LCD_text (clave_cif);
               }else if(tecla=='a'){
                      // Borrar ultimo digito
                      if(dig>0){
                             clave_int[dig-1]='q';
                             clave_cif[dig-1]=' ';
                             clave_ext[dig-1]=' ';
```

```
                        dig=dig-1;
                        LCD_cursor (0,1);
                        LCD_text (clave_cif);
                }
        }else if(tecla=='b'){
                if(dig>0)
                        x=10;
        }
        if(x==10)
        break;
}

clave[0]=clave_ext[0];
clave[1]=clave_ext[1];
clave[2]=clave_ext[2];
clave[3]=clave_ext[3];
clave[4]=clave_ext[4];
clave[5]=clave_ext[5];

printf("\n  Clave anterior:");
for (c=0; c<6; c++){
   printf("%c",clave[c]);
}

cifrar_clave();

f=0;
f=(atoi(usuario)*1000000)+(atoi(clave));
itoa(f);
printf("\n  Clave cifrada:");
for (c=0; c<12; c++){
   printf("%c",clave_cifrada_car[c]);
  }
}

dig=0; x=0; p=0;
LCD_clear();
LCD_cursor (0,0);
LCD_text ("Clave nueva: ");

while (1){
        LCD_cursor (0,1);
        LCD_text (clave_nueva_cif);
        estado_ingreso=4;
        tecla=leer_tecla_tiempo();
        // solo se pueden ingresar hasta 5 digitos
        if(EsNumero(tecla) && dig<6){
                clave_nueva_int[dig]=tecla;
                dig=dig+1;
                for (p=0;p<6;p++){
                        if(clave_nueva_int[p]!='q'){
                                clave_nueva_ext[p]=clave_nueva_int[p];
                                clave_nueva_cif[p]='*';
                        }
                }
```

```
                //Mostrar numero en pantalla
                LCD_cursor (0,1);
                LCD_text (clave_nueva_cif);
        }else if(tecla=='a'){
                // Borrar ultimo digito
                if(dig>0){
                        clave_nueva_int[dig-1]='q';
                        clave_nueva_cif[dig-1]=' ';
                        clave_nueva_ext[dig-1]=' ';
                        dig=dig-1;
                        LCD_cursor (0,1);
                        LCD_text (clave_nueva_cif);
                }
        }else if(tecla=='b'){
                if(dig>0)
                        x=10;
        }
        if(x==10)
        break;
    }

    clave_nueva[0]=clave_nueva_ext[0];
    clave_nueva[1]=clave_nueva_ext[1];
    clave_nueva[2]=clave_nueva_ext[2];
    clave_nueva[3]=clave_nueva_ext[3];
    clave_nueva[4]=clave_nueva_ext[4];
    clave_nueva[5]=clave_nueva_ext[5];

    printf("\n  Clave nueva:");
    for (c=0; c<6; c++){
       printf("%c",clave_nueva[c]);
}
    cifrar_clave_nueva();

    f=0;
    f=(atoi(usuario)*1000000)+(atoi(clave_nueva));
    itoa(f);

    dig=0; x=0; p=0;
    LCD_clear();
    LCD_cursor (0,0);
    LCD_text ("Confirme clave: ");

    while (1){
        LCD_cursor (0,1);
        LCD_text (clave_nueva_confirmacion_cif);
        estado_ingreso=5;
        tecla=leer_tecla_tiempo();
        // solo se pueden ingresar hasta 5 digitos
        if(EsNumero(tecla) && dig<6){
                clave_nueva_confirmacion_int[dig]=tecla;
                dig=dig+1;
                for (p=0;p<6;p++){
                        if(clave_nueva_confirmacion_int[p]!='q'){
                                clave_nueva_confirmacion_ext[p]=
```

```
                                clave_nueva_confirmacion_int[p];
                                clave_nueva_confirmacion_cif[p]='*';
                        }
                }
                //Mostrar numero en pantalla
                LCD_cursor (0,1);
                LCD_text (clave_nueva_confirmacion_cif);
        }else if(tecla=='a'){
                // Borrar ultimo digito
                if(dig>0){
                        clave_nueva_confirmacion_int[dig-1]='q';
                        clave_nueva_confirmacion_cif[dig-1]=' ';
                        clave_nueva_confirmacion_ext[dig-1]=' ';
                        dig=dig-1;
                        LCD_cursor (0,1);
                        LCD_text (clave_nueva_confirmacion_cif);
                }
        }else if(tecla=='b'){
                if(dig>0)
                        x=10;
        }
        if(x==10)
        break;
}
clave_nueva_confirmacion[0]=clave_nueva_confirmacion_ext[0];
clave_nueva_confirmacion[1]=clave_nueva_confirmacion_ext[1];
clave_nueva_confirmacion[2]=clave_nueva_confirmacion_ext[2];
clave_nueva_confirmacion[3]=clave_nueva_confirmacion_ext[3];
clave_nueva_confirmacion[4]=clave_nueva_confirmacion_ext[4];
clave_nueva_confirmacion[5]=clave_nueva_confirmacion_ext[5];

f=0;
f=(atoi(usuario)*1000000)+(atoi(clave));
itoa(f);

mensaje_enviar_cambio_clave[0]=temporal[0];
mensaje_enviar_cambio_clave[1]=temporal[1];
mensaje_enviar_cambio_clave[2]=temporal[2];
mensaje_enviar_cambio_clave[3]=',';
mensaje_enviar_cambio_clave[4]=clave_cifrada_car[0];
mensaje_enviar_cambio_clave[5]=clave_cifrada_car[1];
mensaje_enviar_cambio_clave[6]=clave_cifrada_car[2];
mensaje_enviar_cambio_clave[7]=clave_cifrada_car[3];
mensaje_enviar_cambio_clave[8]=clave_cifrada_car[4];
mensaje_enviar_cambio_clave[9]=clave_cifrada_car[5];
mensaje_enviar_cambio_clave[10]=clave_cifrada_car[6];
mensaje_enviar_cambio_clave[11]=clave_cifrada_car[7];
mensaje_enviar_cambio_clave[12]=clave_cifrada_car[8];
mensaje_enviar_cambio_clave[13]=clave_cifrada_car[9];
mensaje_enviar_cambio_clave[14]=clave_cifrada_car[10];
mensaje_enviar_cambio_clave[15]=clave_cifrada_car[11];
mensaje_enviar_cambio_clave[16]=',';
mensaje_enviar_cambio_clave[17]=clave_cifrada_car_nuevo[0];
mensaje_enviar_cambio_clave[18]=clave_cifrada_car_nuevo[1];
mensaje_enviar_cambio_clave[19]=clave_cifrada_car_nuevo[2];
```

```
mensaje_enviar_cambio_clave[20]=clave_cifrada_car_nuevo[3];
mensaje_enviar_cambio_clave[21]=clave_cifrada_car_nuevo[4];
mensaje_enviar_cambio_clave[22]=clave_cifrada_car_nuevo[5];
mensaje_enviar_cambio_clave[23]=clave_cifrada_car_nuevo[6];
mensaje_enviar_cambio_clave[24]=clave_cifrada_car_nuevo[7];
mensaje_enviar_cambio_clave[25]=clave_cifrada_car_nuevo[8];
mensaje_enviar_cambio_clave[26]=clave_cifrada_car_nuevo[9];
mensaje_enviar_cambio_clave[27]=clave_cifrada_car_nuevo[10];
mensaje_enviar_cambio_clave[28]=clave_cifrada_car_nuevo[11];

if(clave_nueva[0]==clave_nueva_confirmacion[0]&&clave_nueva[1]==
       clave_nueva_confirmacion[1]&&clave_nueva[2]==
       clave_nueva_confirmacion[2]&&clave_nueva[3]==
       clave_nueva_confirmacion[3]&&clave_nueva[4]==
       clave_nueva_confirmacion[4]&&clave_nueva[5]==
       clave_nueva_confirmacion[5]){

       printf("\n  Confirmacion:");
       for (c=0; c<6; c++){
              printf("%c",clave_nueva_confirmacion[c]);
       }
       printf("\n  Clave nueva cifrada:");
       for (c=0; c<12; c++){
              printf("%c",clave_cifrada_car_nuevo[c]);
       }
       contador_intentos_cambio=0;
       estado_sistema=0;
       while(estado_sistema!=7 && contador_intentos_cambio<2){

              estado_sistema=0;
              LCD_clear();
              LCD_cursor (0,0);
              LCD_text ("  Verificando  ");
              LCD_cursor (0,1);
              LCD_text ("    Datos...   ");
              LCD_cursor_off ();
              Inicializacion_DM9000();
              usleep(2000000);
              Enviar_Mensaje(mensaje_enviar_cambio_clave);
              printf("\n  Mensaje ");
              for(c=0; c<29; c++){
                     printf("%c",mensaje_enviar_cambio_clave[c]);
              }
              printf(" enviado al servidor");
              usleep(1000000);

              if(estado_sistema!=0){

                     Enviar_Mensaje("confirmado");
                     printf("  Mensaje CONFIRMADO enviado al
                     servidor");
                     if(estado_sistema==7){
                            LCD_clear();
                            LCD_cursor (0,0);
                            LCD_text (" Clave cambiada ");
```

```
                                        LCD_cursor (0,1);
                                        LCD_text ("  Exitosamente  ");
                                        LCD_cursor_off ();
                                        printf("\n  Datos actualizados en el
                                                    servidor\n");
                                        usleep(2000000);
                                        contador_intentos_cambio=1;
                                        printf("\nIngreso de datos:\n");
                                }
                        }
                        if(contador_intentos_cambio==0){
                        printf("\n  Error: Tiempo de espera para recibir una
                                                respuesta del servidor agotada
                                                (1 segundo)");
                         }
                        contador_intentos_cambio++;
                }

                if(estado_sistema!=7){
                        printf("\n  Problema: ID_Clave de usuario ingresados
                                                incorrectamente\n");
                        LCD_clear();
                        LCD_cursor (0,0);
                        LCD_text ("ID_Clave usuario");
                        LCD_cursor (0,1);
                        LCD_text ("     Erroneo    ");
                        LCD_cursor_off ();
                        usleep(2000000);
                        printf("\nIngreso de datos:\n");
                }
                salto=0;

        }else{
                printf("\n  Error: Clave confirmada diferente");
                LCD_clear();
                LCD_cursor (0,0);
                LCD_text ("Clave confirmada");
                LCD_cursor (0,1);
                LCD_text ("    diferente   ");
                usleep(1000000);
                salto=1;
        }
    }
    iow(IMR, INTR_set);
}
```

CÓDIGO DEL PROGRAMA PRINCIPAL

```
int main(void)
{
  volatile int *p_green_leds=green_leds, *p_red_leds=red_leds;
  int i=0, intentos=0, bloqueo=0, desbloqueo=0,bandera1=0,prueba1=0,
```

```
    repeticion_envio=0, estado_conexion_ethernet=0,
    contador_intentos_conexion=0, c=0;
volatile int delay_count;
char tecla='q';

alt_irq_register(DM9000A_IRQ, NULL, (void*)
                                ethernet_interrupt_handler);

(*p_red_leds)=0x0;
(*p_green_leds)=0x0;
imprimir_consola=1;
contador_intentos_conexion=0;

printf("Configuracion Anterior:\n");
printf("  Tarjeta DE2:\n");
printf("    Direccion IP: %d",transmit_buffer[26]);
printf(".%d",transmit_buffer[27]);
printf(".%d",transmit_buffer[28]);
printf(".%d\n",transmit_buffer[29]);
printf("    Direccion MAC: %x",arp_buffer[6]);
printf(".%x",arp_buffer[7]);
printf(".%x",arp_buffer[8]);
printf(".%x",arp_buffer[9]);
printf(".%x",arp_buffer[10]);
printf(".%x\n",arp_buffer[11]);
printf("  Servidor:\n");
printf("    Direccion IP: %d",transmit_buffer[30]);
printf(".%d",transmit_buffer[31]);
printf(".%d",transmit_buffer[32]);
printf(".%d\n",transmit_buffer[33]);
printf("    Direccion MAC: %x",arp_buffer[0]);
printf(".%x",arp_buffer[1]);
printf(".%x",arp_buffer[2]);
printf(".%x",arp_buffer[3]);
printf(".%x",arp_buffer[4]);
printf(".%x\n",arp_buffer[5]);
printf("  Puerto Origen/destino: 5005\n");
printf("\nIniciando conexion con el servidor...");

while(estado_sistema!=5&&contador_intentos_conexion<2){
    estado_sistema=0;
    LCD_clear();
    LCD_cursor (0,0);
    LCD_text ("  Verificando ");
    LCD_cursor (0,1);
    LCD_text ("   Conexion... ");
    LCD_cursor_off ();
    Inicializacion_DM9000();
    usleep(2000000);

    if (!prueba_enlace_dm9000a()){
          LCD_clear();
          LCD_cursor (0,0);
          LCD_text ("  Cable de red ");
          LCD_cursor (0,1);
```

```
                LCD_text ("  Desconectado ");
                LCD_cursor_off ();
                printf("\n  Problema: Cable de red desconectado");
                estado_conexion_ethernet=0;

                while(estado_conexion_ethernet==0){
                        usleep(2000000);
                        if (prueba_enlace_dm9000a()){
                                estado_conexion_ethernet=1;
                         }
                         Inicializacion_DM9000();
                  }
                         usleep(2000000);
        }
                printf("\n  Mensaje TEST enviado al servidor");
                Enviar_Mensaje("test");
                usleep(1000000);

                if(estado_sistema==5){
                        printf("  Mensaje ACK enviado al servidor\n");
                        LCD_clear();
                        LCD_cursor (0,0);
                        LCD_text ("    Conexion    ");
                        LCD_cursor (0,1);
                        LCD_text ("   Establecida  ");
                        LCD_cursor_off ();
                        Enviar_Mensaje("establecida");
                        printf("  Tarjeta DE2 conectada con servidor\n");

                 }else{
                        if(contador_intentos_conexion==0){
                        printf("\n  Error: Tiempo de espera para recibir una
                                respuesta del servidor agotada (1 segundo)");
                        }

                        if(contador_intentos_conexion==1){
                        printf("\n  Problema: Servidor inalcanzable\n");
                        LCD_clear();
                        LCD_cursor (0,0);
                        LCD_text ("   Configure    ");
                        LCD_cursor (0,1);
                        LCD_text (" Direcciones IP ");
                        LCD_cursor_off ();
                        printf("\nConfiguracion de red:\n");
                        usleep(2000000);
                        printf("  Direccion IP Tarjeta DE2: ");
                        ingreso_direccion_ip_tarjeta();
                        printf("  Direccion IP Servidor:     ");
                        ingreso_direccion_ip_servidor();
                        contador_intentos_conexion=-1;
                        }
                  }
                contador_intentos_conexion++;
}
```

```
contador_verificacion_conexion=0;
while(1){
     estado_conexion_ethernet=0;
     salir=1;
     mensaje_enviar[9]="qqqqqqqqq";
     clave[6]="qqqqqq";
     usuario[3]="qqq";
     i=0;

     estado_sistema=0;
     if(saltodes==0){
     generar_llave_nueva();
     }

     estado_sistema=0;
     repeticion_envio_llave=0;
     while(estado_sistema==0 && repeticion_envio_llave<2 &&
                  saltodes==0){
            Inicializacion_DM9000();
            usleep(2000000);
            Enviar_Mensaje(llave_actual);
            printf("\n  Mensaje %s enviado al servidor",llave_actual);
            usleep(1000000);

            if(estado_sistema==8){
                  printf("  Llave enviada correctamente al servidor\n");
                  printf("\nIngreso de datos:\n");
             }else{
                  printf("\n  Error: Tiempo de espera para recibir una
                        respuesta del servidor agotada (1 segundo)");
                }

            repeticion_envio_llave++;
            if(repeticion_envio_llave==2 && estado_sistema==0){
            printf("\n\nProblema: Servidor sin llave de cifrado
                        actual\n");
            printf("\nIngreso de datos:\n");
            }
      }

     (*p_red_leds)=0x0;
     (*p_green_leds)=0x0;
     while(salir==1){
            ingreso_usuario_clave();
     }

     estado_sistema=0;
     repeticion_envio=0;
     while(estado_sistema==0 && repeticion_envio<2){

            if(repeticion_envio==1){
                  printf("\n  Error: Tiempo de espera para recibir una
                        respuesta del servidor agotada (1 segundo)");
                  LCD_cursor (0,0);
                  LCD_text (" Autenticando...");
```

```
                LCD_cursor (0,1);
                LCD_text ("                ");
                LCD_cursor_off ();
        }
        Inicializacion_DM9000();
        usleep(2000000);
        Enviar_Mensaje(mensaje_enviar);
        printf("\n  Mensaje ");
        for (c=0; c<16; c++){
        printf("%c",mensaje_enviar[c]);
        }
        printf(" enviado al servidor ");
        usleep(1000000);
        repeticion_envio++;
}

if(estado_sistema!=0){
        Enviar_Mensaje("ack");
        printf("  Mensaje ACK enviado al servidor\n");

        if(estado_sistema==1){
                (*p_green_leds)=0x01;
                printf("  Puerta desbloqueada\n");
                LCD_clear();
                LCD_cursor (0,0);
                LCD_text ("    Usuario    ");
                LCD_cursor (0,1);
                LCD_text ("  Autorizado  ");
                LCD_cursor_off ();
                usleep(2000000);

                LCD_clear();
                LCD_cursor (0,0);
                LCD_text ("     Puerta     ");
                LCD_cursor (0,1);
                LCD_text ("  Desbloqueada  ");
                LCD_cursor_off ();

        }else if(estado_sistema==2){
                (*p_red_leds)=0x20000;
                printf("  Puerta bloqueada\n");
                LCD_clear();
                LCD_cursor (0,0);
                LCD_text ("     Clave    ");
                LCD_cursor (0,1);
                LCD_text ("   Incorrecta  ");
                LCD_cursor_off ();
                usleep(2000000);
           }
        else if(estado_sistema==3){
                printf("\nTarjeta DE2 bloqueada...");

                while(estado_sistema!=4){
                        (*p_red_leds)=0x20000;
                        LCD_clear();
```

```
LCD_cursor (0,0);
LCD_text ("    OFICINA  ");
LCD_cursor (0,1);
LCD_text ("   BLOQUEADA  ");
LCD_cursor_off ();
estado_sistema=0;
Inicializacion_DM9000();
usleep(2000000);

if (!prueba_enlace_dm9000a()){
       LCD_clear();
       LCD_cursor (0,0);
       LCD_text ("  Cable de red ");
       LCD_cursor (0,1);
       LCD_text ("  Desconectado ");
       LCD_cursor_off ();
              prueba1=0;
              while(prueba1==0){
                     usleep(2000000);
                     if (prueba_enlace_dm9000a()){
                            prueba1=1;
                     }
                     Inicializacion_DM9000();
               }
       LCD_clear();
       LCD_cursor (0,0);
       LCD_text ("    OFICINA  ");
       LCD_cursor (0,1);
       LCD_text ("   BLOQUEADA  ");
       LCD_cursor_off ();
       usleep(2000000);
}

if(contador_verificacion_conexion1>=10){
       while(estado_sistema!=5 && estado_sistema!=4){

              if(bandera1>=1){
                     Inicializacion_DM9000();
                     usleep(2000000);
              }
              Enviar_Mensaje("test");
              printf("\n  Mensaje TEST enviado al
                                   servidor");
              usleep(1000000);

              if(estado_sistema==5){

                     contador_verificacion_conexion1=0;
                     Enviar_Mensaje("bloqueado");
                     printf("  Mensaje BLOQUEADO
                     enviado al servidor\n");
                     bandera1=0;
              }

              if(estado_sistema==4){
```

```
                                        contador_verificacion_conexion1=0;
                                              bandera1=0;
                                        }

                                        if(bandera1>=1){
                                              LCD_clear();
                                              LCD_cursor (0,0);
                                              LCD_text ("  Desconexion  ");
                                              LCD_cursor (0,1);
                                              LCD_text ("   Servidor    ");
                                        }
                                        bandera1++;
                                  }
                                  bandera1=0;
                             }
                            if(estado_sistema==0){
                               usleep(1000000);
                             }
                            contador_verificacion_conexion1=
                            contador_verificacion_conexion1+4;
                      }
                      LCD_clear();
                      LCD_cursor (0,0);
                      LCD_text ("    OFICINA  ");
                      LCD_cursor (0,1);
                      LCD_text (" DESBLOQUEADA ");
                      Enviar_Mensaje("desbloqueado");
                      printf("  Mensaje DESBLOQUEADO enviado al servidor");
                      printf("\n  Tarjeta DE2 desbloqueada\n");
                      usleep(2000000);
                }
                else if(estado_sistema==6){
                      printf("\n  Problema: ID de usuario ingresado no
                                        es valido\n");
                      LCD_clear();
                      LCD_cursor (0,0);
                      LCD_text ("   ID Usuario   ");
                      LCD_cursor (0,1);
                      LCD_text ("   Incorrecto   ");
                      LCD_cursor_off ();
                      usleep(2000000);
                }
          }else{
                printf("\n  Problema: Autenticacion no recibida
                                  del servidor");
                saltodes=1;
                contador_verificacion_conexion=16000000;
            }

    }
    return 0;
}
```

CÓDIGO DEL DM9000A.C

```
#include <stdio.h>
#include "dm9000a.h"
#include "basic_io.h"

unsigned char ether_addr[6]={ 0x01, 0x60, 0x6E, 0x11, 0x02, 0x0F  };

int prueba_enlace_dm9000a()
{
    // Leer el estatus del bit de red
    iow( NSR, 0xff );
    IOWR(DM9000A_BASE,IO_addr,NSR);
        usleep(STD_DELAY);
        unsigned char tmp = IORD(DM9000A_BASE,IO_data);
    // Devuelve el bit del estatus del enlace
    return !!(tmp & (1<<6));
}

//--------------------------------------------------------------------------
void iow(unsigned int reg, unsigned int data)
{
  IOWR(DM9000A_BASE,IO_addr,reg);
  usleep(STD_DELAY);
  IOWR(DM9000A_BASE,IO_data,data);
}

//--------------------------------------------------------------------------
unsigned int ior(unsigned int reg)
{
  IOWR(DM9000A_BASE,IO_addr,reg);
  usleep(STD_DELAY);
  return IORD(DM9000A_BASE,IO_data);
}

//--------------------------------------------------------------------------
void phy_write (unsigned int reg, unsigned int value)
{
  /* Setea la dirección del registro PHY en EPAR REG. 0CH */
  iow(0x0C, reg | 0x40); /* DM9000_PHY offset  = 0x40 */

  /* Llenar el dato en el registro PHY en EPDR REG. 0EH & REG. 0DH */
  iow(0x0E, ((value >> 8) & 0xFF));   /* byte msb de PHY */
  iow(0x0D, value & 0xFF);            /* byte msb de PHY */

  iow(0x0B, 0x8);                     /* Comando limpiar PHY */
  IOWR(DM9000A_BASE, IO_data, 0x0A);  /* Comando editar PHY + WRITE*/
  usleep(STD_DELAY);
  IOWR(DM9000A_BASE, IO_data, 0x08);  /* Comando limpiar PHY otra vez */
  usleep(50);  /* esperar 1~30 us (>20 us) para completar PHY + WRITE */
}


//--------------------------------------------------------------------------
```

```
unsigned int Inicializacion_DM9000 (void) /* Inicializar el chip DM9000 LAN */
{
  unsigned int  i;

  /* Setear el encendido del PHY interno (Seteando los GPIOs normal) */
  iow (0x1E, 0x01);  /* GPCR REG. 1EH = 1 */
  /* GPR  REG. 1FH GEPIO0 Bit [0] = 0 para activar el PHY interno */
  iow (0x1F, 0x00);
  msleep (5);    /* esperar > 2 ms para que el encendido del PHY este listo */

  /* Resetear la NIC */
  /* NCR REG. 00 RST Bit [0] = 1 para resetear, and LBK Bit [2:1] = 01b MAC
loopback encendido */
  iow (NCR, 0x03);
  usleep (20);/* esperar>10us para que el reseteo del software sea correcto */
  iow (NCR, 0x00);
  iow (NCR, 0x03);
  usleep (20);
  iow (NCR, 0x00);

  /* Setear GPIO0=1 luego GPIO0=0 para apagar y encender el PHY interno */
  iow (0x1F, 0x01);  /* GPR PHYPD Bit [0] = 1 apagar PHY */
  iow (0x1F, 0x00);  /* PHYPD Bit [0] = 0 encender PHY */
  msleep (10);       /* esperar >4 ms para que se encienda el PHY */

  /* Setear el modo de operación del PHY */
  phy_write(0,PHY_reset);   /* resetear PHY: registros apagados por defecto */
  usleep(50);               /* esperar >30 us para resetear el software */
  phy_write(16, 0x404);     /* apagar PHY */
  phy_write(4, PHY_txab); /* setear PHY TX capacidad: ALL + control de flujo */
  phy_write(0, 0x1200);     /* PHY auto-NEGO habilitado */
  msleep(5);    /* esperar >2 ms para PHY auto-sense se enlace con su vecino */

  /* Guardar MAC address en la NIC */
  for (i = 0; i < 6; i++)
       iow(16 + i, ether_addr[i]);

  /* Limpiar alguna interrupción pendiente */
  /* Limpiar el estatus del ISR: PRS, PTS, ROS, ROOS 4 bits, by RW/C1 */
  iow(ISR, 0x3F);
  /* Limpiar el estatus del TX: TX1END, TX2END, WAKEUP 3 bits, by RW/C1 */
  iow(NSR, 0x2C);

  /* Registros de operación del programa ~ */
  iow(NCR,  NCR_set);   /* NCR REG. 00 habilita las funciones del chip (y
deshabilita el modo loopback luego regresa al modo normal) */
  iow(0x08, BPTR_set);  /* BPTR  REG.08  (si es necesario) RX Back Pressure
Threshold solo en modo half dúplex: 3KB, 600 us */
  iow(0x09, FCTR_set);  /* FCTR  REG.09  (si es necesario) Flow Control
Threshold seteando el alto/ 5KB/ 10KB */
  iow(0x0A, RTFCR_set); /* RTFCR REG.0AH (si es necesario) RX/TX Flow Control
Register habilitar TXPEN, BKPM (TX_Half), FLCE (RX) */
  iow(0x0F, 0x00);      /* Limpiar todos los eventos */
  iow(0x2D, 0x80);      /* Cambiar el modo LED a 1 */
```

```
  /* Setear los otros registros dependiendo de la aplicacion */
  iow(ETXCSR, ETXCSR_set); /* Porcentaje de transmision 75% */

  /* Habilitar las interrupciones para activar el DM9000  */
  iow(IMR, INTR_set);   /* IMR REG. FFH PAR=1 */

  /* Habilitar el RX (Broadcast/ ALL_MULTICAST) */
  /* RCR REG. 05 RXEN Bit [0] = 1 */
  iow(RCR , RCR_set | RX_ENABLE | PASS_MULTICAST);

  return  (ior(0x2D)==0x80) ? DMFE_SUCCESS : DMFE_FAIL;
}


//------------------------------------------------------------------------
unsigned int  Transmitir_Paquete(unsigned char *data_ptr,unsigned int tx_len)
{
  unsigned int  i;

   /* Interrupt Mask Register IMR: PAR */
  iow(IMR, PAR_set);

  /* La longitud de emisión del paquete TX TXPLH REG. FDH & TXPLL REG. FCH */
  iow(0xFD, (tx_len >> 8) & 0xFF);  /* longitud de los bytes msb TXPLH */
  iow(0xFC, tx_len & 0xFF);         /* longitud de los bytes lsb TXPLL */

  /* Escribir el dato a transmitir a el chip SRAM */
  /* Setear MWCMD REG. F8H TX I/O puerto listo */
  IOWR(DM9000A_BASE, IO_addr, MWCMD);
  for (i = 0; i < tx_len; i += 2)
  {
    usleep(STD_DELAY);
    IOWR(DM9000A_BASE, IO_data, (data_ptr[i+1]<<8)|data_ptr[i] );
  }

  /* Comando de sondeo TX activado */
  /* TXCR Bit [0] TXREQ, limpiar después que la transmisión se completó*/
  iow(TCR , TCR_set | TX_REQUEST);

  /* Esperar que la transmisión se complete */
  usleep(STD_DELAY);

  /* Limpiar el registro NSR */
  iow (NSR,0x00);

  /* Rehabilitar la interrupciones de la NIC */
  iow(IMR, INTR_set);

  return  DMFE_SUCCESS;
}
```

```
//---------------------------------------------------------------------------
unsigned int  Recibir_Paquete (unsigned char *data_ptr,unsigned int *rx_len)
{
  unsigned char rx_READY,GoodPacket;
  unsigned int  Tmp, RxStatus, i;

  RxStatus = rx_len[0] = 0;
  GoodPacket=FALSE;

   /* Interrupt Mask Register IMR: PAR */
  iow(IMR, PAR_set);

  /* Simular leer un byte de MRCMDX REG. F0H */
  rx_READY = ior(MRCMDX);

  /* Obtener el byte más actualizado: rx_READY */
  rx_READY = /*0x01;*/IORD(DM9000A_BASE,IO_data)&0x03;
  usleep(STD_DELAY);

  /* Chequear si (rx_READY == 0x01): Listo para recibir paquete? */
  if (rx_READY == DM9000_PKT_READY)
  {
    /* Obtener el RX_Status & RX_Length de RX SRAM */
    /* Setear MRCMD REG. F2H RX I/O puerto listo */
    IOWR(DM9000A_BASE, IO_addr, MRCMD);
    usleep(STD_DELAY);
    RxStatus = IORD(DM9000A_BASE,IO_data);
    usleep(STD_DELAY);
    rx_len[0] = IORD(DM9000A_BASE,IO_data);

    /* Chequear el estatus de este paquete BUENO o MALO? */
    if ( !(RxStatus & 0xBF00) && (rx_len[0] < MAX_PACKET_SIZE) )
    {
      /* Leer un paquete recibido de la RX SRAM en el buffer RX */
      for (i = 0; i < rx_len[0]; i += 2)
      {
        usleep(STD_DELAY);
        Tmp = IORD(DM9000A_BASE, IO_data);
        data_ptr[i] = Tmp&0xFF;
        data_ptr[i+1] = (Tmp>>8)&0xFF;
      }
      GoodPacket=TRUE;
    } /* Finaliza si (GoodPacket) */
    else
    {
      /* Este paquete es malo, simula que está en la RX SRAM */
      for (i = 0; i < rx_len[0]; i += 2)
      {
        usleep(STD_DELAY);
        Tmp = IORD(DM9000A_BASE, IO_data);
      }
      rx_len[0] = 0;
    } /* Finaliza si (!GoodPacket) */
  } /* Finaliza si (rx_READY == DM9000_PKT_READY) */
```

```
  else if(rx_READY) /* El primer byte chequea el estatus: rx_READY Bit[1:0]
debería ser "00"b o "01"b */
  {
    /* Resetea la NIC */
    iow(NCR, 0x03); /* NCR REG. 00 RST Bit [0] = 1 reset on, and LBK Bit [2:1]
= 01b MAC loopback on */
    usleep(20);       /* espera > 10us para un correcto reseteo del software */
    iow(NCR, 0x00);
    iow(NCR, 0x03);
    usleep(20);
    iow(NCR, 0x00);

    /* Registros de operación del programa  */
    iow(NCR,  NCR_set);   /* NCR REG. 00 habilitar las funciones del chip (y
      cambiar el modo de operación de la NIC de loopback a modo normal) */
    iow(0x08, BPTR_set);  /* BPTR  REG.08  (si es necesario) RX Back Pressure
              Threshold solo en modo half dúplex: 3KB, 600 us */
     iow(0x09, FCTR_set);  /* FCTR  REG.09  (si es necesario) Flow Control
           Threshold setear en alto/ Low Water Overflow 5KB/ 10KB */
    iow(0x0A, RTFCR_set); /* RTFCR REG.0AH (si es necesario) RX/TX Flow Control
            Register habilitar TXPEN, BKPM (TX_Half), FLCE (RX) */
    iow(0x0F, 0x00);      /* Limpiar todos los eventos */
    iow(0x2D, 0x80);      /* Cambiar el modo del LED a 1 */

    /* Setear los otros registros dependiendo de las aplicaciones */
    iow(ETXCSR, ETXCSR_set); /* Porcentaje de transmisión 75% */
    /* Habilitar las interrupciones para activar el DM9000 */
    /* IMR REG. FFH PAR=1, or+PTM=1& PRM=1 habilitar las interrupciones RxTx */
    iow(IMR, INTR_set);
    /* Habilitar RX (Broadcast/ ALL_MULTICAST) */
    iow(RCR , RCR_set | RX_ENABLE | PASS_MULTICAST);
    /* RCR REG. 05 RXEN Bit [0] = 1 */
  }

  return GoodPacket ? DMFE_SUCCESS : DMFE_FAIL;
}
//------------------------------------------------------------------------------
```

ANEXO B

INTERFAZ GRÁFICA EN JAVA

VENTANAS DE LA APLICACIÓN

Figura B.1 – Ventana principal.

Figura B.2 – Agregar un usuario.

Figura B.3 – Agregar una oficina.

Figura B.4 – Modificar los datos y clave de un usuario.

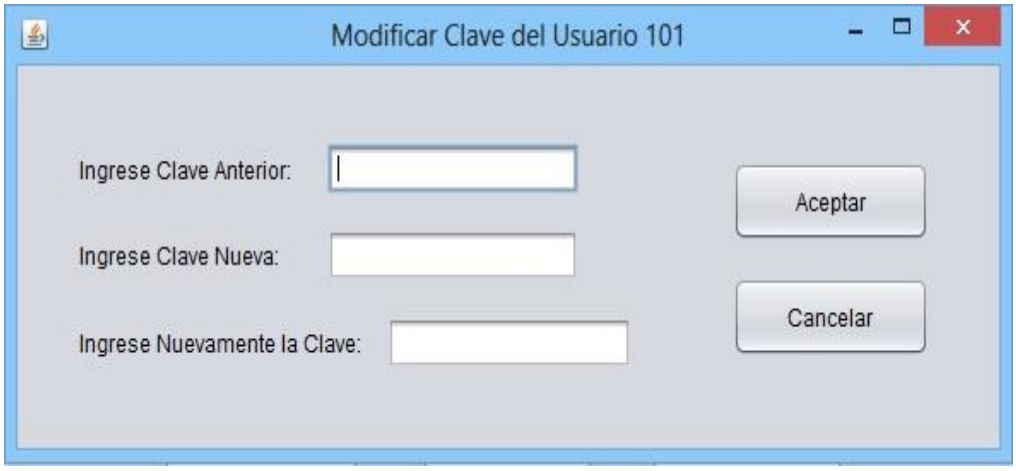


Figura B.5 – Cambiar la clave de un usuario.

Figura B.6 – Ventana con las opciones de una oficina.

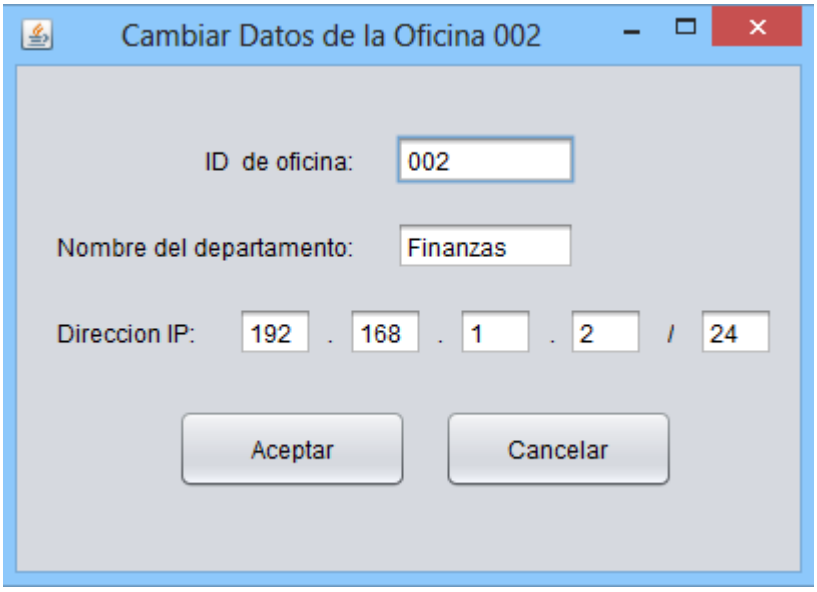


Figura B.7 – Cambiar los datos de una oficina.

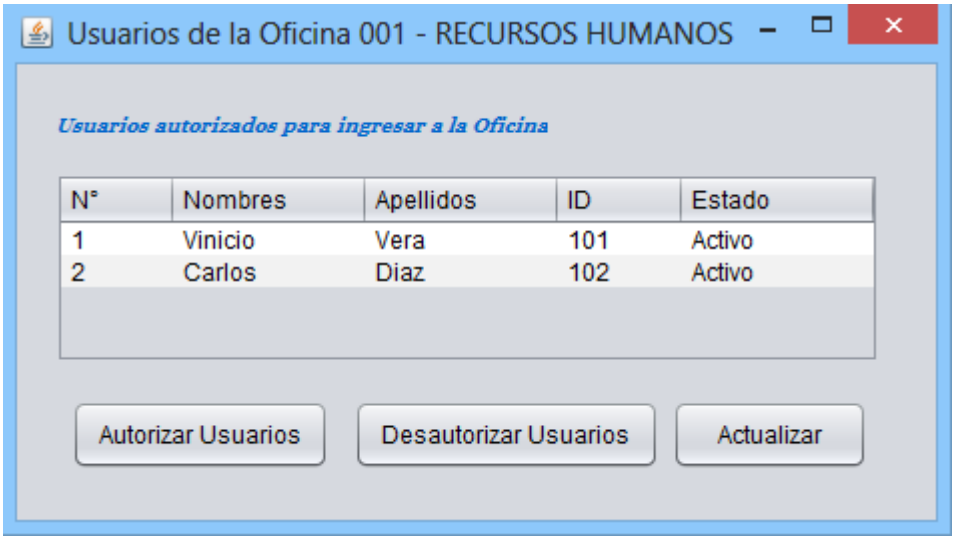


Figura B.8 – Lista de usuarios autorizados.

Figura B.9 – Autorizar el ingreso de nuevos usuarios.

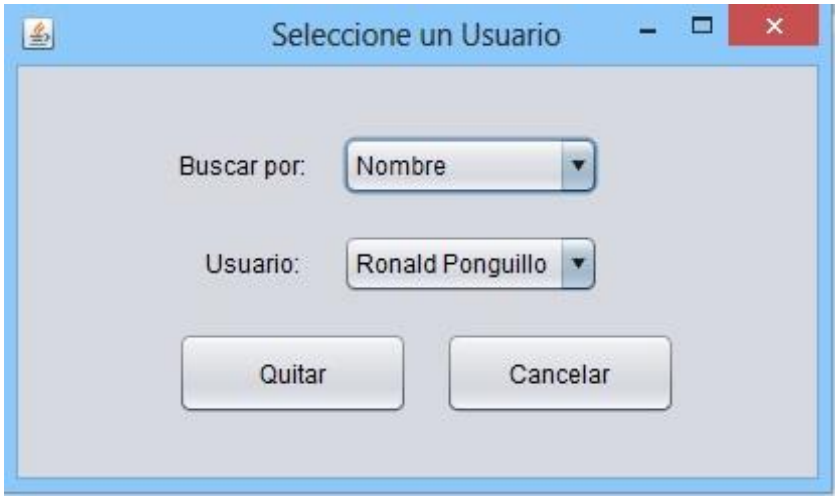


Figura B.10 – Desautorizar el ingreso de usuarios.

Figura B.11 – Ventana con las opciones de un usuario.

Figura B.12 – Ventana para desbloquear una oficina.

ANEXO C

CÓDIGO FUENTE EN LENGUAJE JAVA

LIBRERÍAS

```
package interfazgrafica;
import com.mysql.jdbc.EscapeTokenizer;
import interfazproyecto.Descifrar_Clave;
import interfazproyecto.RegistroHistorial;
import java.util.GregorianCalendar;
import interfazgrafica.Buscar_Usuarios;
import interfazproyecto.Network;
import interfazgrafica.Buscar_Oficinas;
import interfazgrafica.Agregar_Usuarios;
import interfazgrafica.Agregar_Oficinas;
import interfazproyecto.*;
import java.util.ArrayList;
import java.awt.Color;
import java.io.*;
import java.net.SocketException;
import java.net.UnknownHostException;
import java.sql.*;
import java.util.*;
import javax.sound.sampled.*;
import javax.swing.*;
```

PROGRAMA PRINCIPAL

```
public static void main(String args[]) throws UnknownHostException,
        SocketException, IOException, LineUnavailableException,
        UnsupportedAudioFileException {
```

DECLARACION DE VARIABLES

```
    String octeto1, octeto2, octeto3, octeto4, mascara="";
    Clip sonido;
    int port,i, idoficina=0, idusuario, intentos, numero_registro,comas=0,
            oficina_existente, usuario_existente, clave_correcta,
            usuario_autorizado_oficina, estado_usuario, veces=0, oct1=0,
            oct2=0, oct3=0,oct4=0,i1, mes, idest=0, existe_sistema=0,
```

```
        id_oficina_actual=0, id_usuario_actual=0, es_llave=1,
        i_estado=0,i_autorizado=0, i_existe_base=0;

String mensaje_recibido, m, oficina="", usuario_recibido,
        usuario_recibido_cambio="", clave_recibida="",
        clave_nueva_cambio="", clave_cifrada, clave_anterior="",
        clave_anterior_cifrada="", clave_nueva="", n="1", llave = null,
        nombre_presentar="",apellido_presentar="", llave_base="",
        mensaje_imprimir="";;

ArrayList<Usuario> usuarios_existentes;
ArrayList<String> primeros_octetos, ultimos_octetos;
StringTokenizer tokens,tokens1,tk,tk1;
Object[] fila=new Object[8];
Object[] fila_bloquear=new Object[6];
Calendar fecha, tiempo_recepcion, tiempo_respuesta;
RegistroHistorial registro;
oficinas_sistema=new ArrayList<Oficina>();
Database db_ofi1 = new Database();
CallableStatement prcProcedimientoAlmacenadocambio;
```

DESARROLLO DE LA APLICACION

```
intentos=0;
numero_registro=1;
numero_registro2=1;
Verificar_Tiempo_Test hilo_verificar_tiempo_test=new
        Verificar_Tiempo_Test();

hilo_verificar_tiempo_test.start();

while(true){

    existe_sistema=0;
    comas=0;
    es_llave=1;
    id_oficina_actual=0;
    oficinas_existentes=new ArrayList<Oficina>();
    Database db_ofi = new Database();

    port = args.length == 0 ? 5005: Integer.parseInt(args[0]);
    m=Network.Recibir_Paquetes(port);
    mensaje_recibido=m.substring(0, m.length()-1);
    tiempo_recepcion=new GregorianCalendar();

    Object[][] data_ofi = db_ofi.select("TodasOficina", "Numero, "
            + "DireccionIp, idOficina, Estado,Nombre", null);

    for(i=0;i<data_ofi.length;i++){
        primeros_octetos=new ArrayList<String>();
        ultimos_octetos=new ArrayList<String>();
        tokens=new StringTokenizer(data_ofi[i][1].toString(),".");
```

```
            while(tokens.hasMoreTokens())
                primeros_octetos.add(tokens.nextToken());

        octeto1=primeros_octetos.get(0);
        octeto2=primeros_octetos.get(1);
        octeto3=primeros_octetos.get(2);

        tokens1=new StringTokenizer(primeros_octetos.get(3),"/");
            while(tokens1.hasMoreTokens())
                ultimos_octetos.add(tokens1.nextToken());

        octeto4=ultimos_octetos.get(0);
        mascara=ultimos_octetos.get(1);

        oficinas_existentes.add(new Oficina(data_ofi[i][0].toString(),
                Integer.parseInt(octeto1),Integer.parseInt(octeto2),
                Integer.parseInt(octeto3), Integer.parseInt(octeto4),
                Integer.parseInt(mascara),Integer.parseInt(data_ofi[i]
                [2].toString()),data_ofi[i][3].toString(),null,
                data_ofi[i][4].toString()));
    }

    for(i=0;i<mensaje_recibido.length();i++){
        if(mensaje_recibido.charAt(i)==','){
            es_llave=0;
        }
    }

    for(i=0;i<mensaje_recibido.length();i++){
        if(mensaje_recibido.charAt(i)==','){
            comas++;
        }
    }

    if(es_llave==1&&!mensaje_recibido.equals("ack")&&!mensaje_recibido.
            equals("test")&&!mensaje_recibido.equals("desbloqueado")&&
            !mensaje_recibido.equals("establecida")&&
            !mensaje_recibido.equals("confirmado")&&!mensaje_recibido.
            equals("bloqueado")){

        llave=mensaje_recibido;
        for(i=0;i<oficinas_sistema.size();i++){
            if(Network.ip_envio.getHostAddress().equalsIgnoreCase(
                    oficinas_sistema.get(i).getOcteto1()+"."+
                oficinas_sistema.get(i).getOcteto2()+"."+
                    oficinas_sistema.get(i).getOcteto3()+"."+
                oficinas_sistema.get(i).getOcteto4())){

                id_oficina_actual=i;
            }
        }
        try {
            Connection conexion = null;
            Class.forName("com.mysql.jdbc.Driver");
            conexion = DriverManager.getConnection("jdbc:mysql://"
```

```
                + "localhost/ethernet","root","123456");
            conexion.setAutoCommit(false);
            Statement s=conexion.createStatement();
            CallableStatement prcProcedimientoAlmacenado = conexion.
                    prepareCall("{ call SetearLLave(?,?) }");
            try{
                prcProcedimientoAlmacenado.setString(1,
                oficinas_sistema.get(id_oficina_actual).getNumero());
            }catch(Exception e){}
            prcProcedimientoAlmacenado.setString(2,llave);
            prcProcedimientoAlmacenado.execute();
            conexion.commit();

        } catch (SQLException ex) {
        } catch (Exception e1) {
        }
        finally{

        }
        System.out.println("Actualizando llave...");
        System.out.println("  Llave "+'\"'+llave+'\"'+" recibida "
                + "desde la oficina "+oficinas_sistema.get(
                id_oficina_actual));
        Network.Enviar_Paquetes("ack",IP);
        System.out.println("  Mensaje ACK enviado a la oficina "+
                oficinas_sistema.get(id_oficina_actual));
        System.out.println("  Llave guardada correctamente");
        try{
            oficinas_sistema.get(id_oficina_actual).setLlave(llave);
            oficinas_sistema.get(id_oficina_actual).setTiempo(0);
        }catch(Exception e){}

    }else if(mensaje_recibido.equals("establecida")){

        for(i=0;i<oficinas_sistema.size();i++){
            if(Network.ip_envio.getHostAddress().equalsIgnoreCase(
                    oficinas_sistema.get(i).getOcteto1()+"."+
                oficinas_sistema.get(i).getOcteto2()+"."+
                    oficinas_sistema.get(i).getOcteto3()+"."+
                oficinas_sistema.get(i).getOcteto4())){

                id_oficina_actual=i;
            }
        }

        System.out.println("  Mensaje ESTABLECIDA recibido desde la "
                + "oficina "+oficinas_sistema.get(id_oficina_actual));
        if(Verificar_Tiempo_Test.fila_conectado!=null){
            System.out.println("  Oficina "+oficinas_sistema.get(
            id_oficina_actual).getNumero()+" conectada con la IP: "+
            oficinas_sistema.get(id_oficina_actual).getOcteto1()+"."+
            oficinas_sistema.get(id_oficina_actual).getOcteto2()+"."+
            oficinas_sistema.get(id_oficina_actual).getOcteto3()+"."+
            oficinas_sistema.get(id_oficina_actual).getOcteto4()+"\n");
```

```
            Principal.modelo1.addRow(Verificar_Tiempo_Test.
                    fila_conectado);
            Principal.numero_registro2++;
            Verificar_Tiempo_Test.fila_conectado=null;
        }

        try{
            oficinas_sistema.get(id_oficina_actual).setTiempo(0);
            oficinas_sistema.get(id_oficina_actual).
                    setBloqueado_actual(0);

        }catch(Exception e){}



    }else if(mensaje_recibido.equals("bloqueado")){

        for(i=0;i<oficinas_sistema.size();i++){
            if(Network.ip_envio.getHostAddress().equalsIgnoreCase(
                    oficinas_sistema.get(i).getOcteto1()+"."+
                oficinas_sistema.get(i).getOcteto2()+"."+
                    oficinas_sistema.get(i).getOcteto3()+"."+
                oficinas_sistema.get(i).getOcteto4())){

                id_oficina_actual=i;
            }
        }


        try{
            oficinas_sistema.get(id_oficina_actual).setTiempo(0);
            System.out.println("Mensaje BLOQUEADO recibido de la "
                    + "oficina "+oficinas_sistema.get(id_oficina_actual));

        }catch(Exception e){}


        if(Verificar_Tiempo_Test.fila_conectado!=null){
            System.out.println("\n  Oficina "+oficinas_sistema.get(
            id_oficina_actual).getNumero()+" conectada con la IP: "+
            oficinas_sistema.get(id_oficina_actual).getOcteto1()+"."+
            oficinas_sistema.get(id_oficina_actual).getOcteto2()+"."+
            oficinas_sistema.get(id_oficina_actual).getOcteto3()+"."+
            oficinas_sistema.get(id_oficina_actual).getOcteto4()+"\n");

            Principal.modelo1.addRow(Verificar_Tiempo_Test.
                    fila_conectado);
            Principal.numero_registro2++;
            Verificar_Tiempo_Test.fila_conectado=null;

        }

        fecha=new GregorianCalendar();
        mes=fecha.get(Calendar.MONTH);
        mes=mes+1;
```

```
            fila_bloquear=new Object[6];
            fila_bloquear[0]=numero_registro2;
            fila_bloquear[1]=fecha.get(Calendar.YEAR)+"-0"+mes+"-"+fecha.
                    get(Calendar.DAY_OF_MONTH);
            fila_bloquear[2]=fecha.get(Calendar.HOUR_OF_DAY)+":"+fecha.
                    get(Calendar.MINUTE)+":"
                    +fecha.get(Calendar.SECOND);
            try{
            fila_bloquear[3]=oficinas_sistema.get(id_oficina_actual).
                    getNumero();
            }catch(Exception e){}
            fila_bloquear[4]=oficinas_sistema.get(id_oficina_actual).
                    getNombre();
            fila_bloquear[5]="Oficina Bloqueada";

            if(oficinas_sistema.get(id_oficina_actual).getVeces() == 0){
                Principal.modelo1.addRow(fila_bloquear);
                numero_registro2++;
            }

            try{
                oficinas_sistema.get(id_oficina_actual).
                        setBloqueado_actual(1);
                oficinas_sistema.get(id_oficina_actual).setVeces(1);
                if(oficinas_sistema.get(id_oficina_actual).
                        getBloqueado_actual()==1){
                    oficinas_sistema.get(id_oficina_actual).
                            setImprimir_bloqueo(1);
                }
            }catch(Exception e){}

        }else if(mensaje_recibido.equals("confirmado")){

            for(i=0;i<oficinas_sistema.size();i++){
                if(Network.ip_envio.getHostAddress().equalsIgnoreCase(
                        oficinas_sistema.get(i).getOcteto1()+"."+
                    oficinas_sistema.get(i).getOcteto2()+"."+
                        oficinas_sistema.get(i).getOcteto3()+"."+
                    oficinas_sistema.get(i).getOcteto4())){

                    id_oficina_actual=i;
                }
            }

            oficinas_sistema.get(id_oficina_actual).setTiempo(0);

            System.out.println("  Mensaje CONFIRMADO recibido desde la "
                    + "oficina "
                    +oficinas_sistema.get(id_oficina_actual));
            System.out.println("  Datos actualizados con exito\n");

            if(Verificar_Tiempo_Test.fila_conectado!=null){

                System.out.println("  Oficina "+oficinas_sistema.get(
                id_oficina_actual).getNumero()+" conectada con la IP: "+
```

```
            oficinas_sistema.get(id_oficina_actual).getOcteto1()+"."+
            oficinas_sistema.get(id_oficina_actual).getOcteto2()+"."+
            oficinas_sistema.get(id_oficina_actual).getOcteto3()+"."+
            oficinas_sistema.get(id_oficina_actual).getOcteto4()+"\n");

            Principal.modelo1.addRow(Verificar_Tiempo_Test.
                    fila_conectado);
            Principal.numero_registro2++;
            Verificar_Tiempo_Test.fila_conectado=null;

        }

        try {

            Connection conexion = null;
            Class.forName("com.mysql.jdbc.Driver");
            conexion = DriverManager.getConnection("jdbc:mysql://local"
                    + "host/ethernet","root","123456");
            conexion.setAutoCommit(false);
            Statement s=conexion.createStatement();
            prcProcedimientoAlmacenadocambio = conexion.prepareCall(
                    "{ call CambiarClave(?,?) }");

            prcProcedimientoAlmacenadocambio.setString(1,
                    usuario_recibido_cambio);
            prcProcedimientoAlmacenadocambio.setString(2,
                    clave_nueva_cambio);
            prcProcedimientoAlmacenadocambio.execute();
            conexion.commit();

            } catch (SQLException ex) {
                ex.printStackTrace();
            } catch (Exception e1) {
                e1.printStackTrace();
            }
            finally{

            }
    }else if(mensaje_recibido.equals("ack")){

        for(i=0;i<oficinas_sistema.size();i++){
            if(Network.ip_envio.getHostAddress().equalsIgnoreCase(
                    oficinas_sistema.get(i).getOcteto1()+"."+
                oficinas_sistema.get(i).getOcteto2()+"."+
                    oficinas_sistema.get(i).getOcteto3()+"."+
                oficinas_sistema.get(i).getOcteto4())){

                id_oficina_actual=i;
            }
        }
        System.out.println("  Mensaje ACK recibido desde la oficina "+
                oficinas_sistema.get(id_oficina_actual));
        oficinas_sistema.get(id_oficina_actual).setTiempo(0);
        System.out.println("  Solicitud respondida "+
                "con exito\n");
```

```
oficinas_sistema.get(id_oficina_actual).setTiempo(0);
if(Verificar_Tiempo_Test.fila_conectado!=null){
    Principal.modelo1.addRow(Verificar_Tiempo_Test.
            fila_conectado);
    Principal.numero_registro2++;
    Verificar_Tiempo_Test.fila_conectado=null;
    System.out.println("  Oficina "+oficinas_sistema.get(
    id_oficina_actual).getNumero()+" conectada con la IP: "+
    oficinas_sistema.get(id_oficina_actual).getOcteto1()+"."+
    oficinas_sistema.get(id_oficina_actual).getOcteto2()+"."+
    oficinas_sistema.get(id_oficina_actual).getOcteto3()+"."+
    oficinas_sistema.get(id_oficina_actual).getOcteto4()+"\n");
}

if(oficinas_sistema.get(id_oficina_actual).getFilas()!=null){
    if(oficinas_sistema.get(id_oficina_actual).getFilas()[7].
            equals("Autorizado")){
        modelo.addRow(oficinas_sistema.get(id_oficina_actual).
                getFilas());
        oficinas_sistema.get(id_oficina_actual).setIntentos(0);
        numero_registro++;
        sonido = AudioSystem.getClip();
        sonido.open(AudioSystem.getAudioInputStream(new File(
                "src/imagenes_sonidos/exito.wav")));
        sonido.start();

        try {
            Thread.sleep (1000);
        } catch (Exception e) {
        }
        sonido.close();

    }else if(oficinas_sistema.get(id_oficina_actual).getFilas()
            [7].equals("Denegado")){
        modelo.addRow(oficinas_sistema.get(id_oficina_actual).
                getFilas());
        oficinas_sistema.get(id_oficina_actual).setIntentos(
                oficinas_sistema.get(id_oficina_actual).
                getIntentos()+1);
        numero_registro++;
        sonido = AudioSystem.getClip();
        sonido.open(AudioSystem.getAudioInputStream(new File(
                "src/imagenes_sonidos/fallido.wav")));
        sonido.start();

        try {
            Thread.sleep (1000);
        } catch (Exception e) {
        }
        sonido.close();
    }

    if(oficinas_sistema.get(id_oficina_actual).getFilas()[7].
            equals("Oficina Bloqueada")){
        //numero_registro++;
```

```
                System.out.println("ent");
                sonido = AudioSystem.getClip();
                sonido.open(AudioSystem.getAudioInputStream(new File(
                        "src/imagenes_sonidos/bloqueo.wav")));
                sonido.start();

                try {
                    Thread.sleep (2000);
                } catch (Exception e) {
                }
                sonido.close();
                BloquearOficina bloquearoficina=new BloquearOficina(
                oficinas_sistema.get(id_oficina_actual).getNumero());
                numero_registro2++;
            }

            registro=new RegistroHistorial(oficinas_sistema.get(
                    id_oficina_actual).getFilas(),oficinas_sistema.
                    get(id_oficina_actual).getIdOficina());
            oficinas_sistema.get(id_oficina_actual).setFilas(null);
        }

        if(oficinas_sistema.get(id_oficina_actual).getFilas_bloquear()
                !=null){
            modelo1.addRow(oficinas_sistema.get(id_oficina_actual).
                    getFilas_bloquear());
            oficinas_sistema.get(id_oficina_actual).
                    setFilas_bloquear(null);
            oficinas_sistema.get(id_oficina_actual).
                    setBloqueado_actual(1);
            oficinas_sistema.get(id_oficina_actual).setVeces(1);
        }


    }else if(mensaje_recibido.equals("test")){

        StringTokenizer to=new StringTokenizer(Network.ip_envio.
                getHostAddress(),".");
        oct1=Integer.parseInt(to.nextToken());
        oct2=Integer.parseInt(to.nextToken());
        oct3=Integer.parseInt(to.nextToken());
        oct4=Integer.parseInt(to.nextToken());
        IP[0]=(byte)oct1;
        IP[1]=(byte)oct2;
        IP[2]=(byte)oct3;
        IP[3]=(byte)oct4;

        for(i=0;i<oficinas_existentes.size();i++){
            if(Network.ip_envio.getHostAddress().equalsIgnoreCase(
                    oficinas_existentes.get(i).getOcteto1()+"."+
                oficinas_existentes.get(i).getOcteto2()+"."+
                    oficinas_existentes.get(i).getOcteto3()+"."+
                oficinas_existentes.get(i).getOcteto4())){
                Network.Enviar_Paquetes("conexion",IP);
            }
```

```
}

if(oficinas_sistema.size()==0){

    for(i=0;i<oficinas_existentes.size();i++){
        if(Network.ip_envio.getHostAddress().equals(
                oficinas_existentes.get(i).getOcteto1()+"."+
                oficinas_existentes.get(i).getOcteto2()+"."+
                oficinas_existentes.get(i).getOcteto3()+
                "."+oficinas_existentes.get(i).getOcteto4())){

            oficinas_sistema.add(new Oficina(oficinas_existentes.
            get(i).getNumero(),oct1,oct2,oct3,oct4,
            oficinas_existentes.get(i).getIdOficina(),0,null,null,
            0,"",0,0,0,0,null,oficinas_existentes.get(i).
            getEstado(),oficinas_existentes.get(i).getNombre()));
        }
    }
}else{
    for(i=0;i<oficinas_sistema.size();i++){
        if(Network.ip_envio.getHostAddress().equals(
                oficinas_sistema.get(i).getOcteto1()+"."+
                oficinas_sistema.get(i).getOcteto2()+"."
                +oficinas_sistema.get(i).getOcteto3()+
                "."+oficinas_sistema.get(i).getOcteto4())){
            id_oficina_actual=i;
            existe_sistema=1;
        }
    }
    if(existe_sistema==0){
        for(i=0;i<oficinas_existentes.size();i++){
            if(Network.ip_envio.getHostAddress().equals(
                    oficinas_existentes.get(i).getOcteto1()+"."+
                    oficinas_existentes.get(i).getOcteto2()+"."+
                    oficinas_existentes.get(i).getOcteto3()+
                    "."+oficinas_existentes.get(i).getOcteto4())){

                oficinas_sistema.add(new Oficina(
                oficinas_existentes.get(i).getNumero(),oct1,
                oct2,oct3,oct4,oficinas_existentes.get(i).
                getIdOficina(),0,null,null,0,"",0,0,0,0,null,
                oficinas_existentes.get(i).getEstado(),
                oficinas_existentes.get(i).getNombre()));
            }
        }
    }
}
    for(i=0;i<oficinas_sistema.size();i++){
    if(Network.ip_envio.getHostAddress().equalsIgnoreCase(
        oficinas_sistema.get(i).getOcteto1()+"."+
        oficinas_sistema.get(i).getOcteto2()+"."+
        oficinas_sistema.get(i).getOcteto3()+"."+
        oficinas_sistema.get(i).getOcteto4())){

        id_oficina_actual=i;
```

```
                }
            }
            try{
                oficinas_sistema.get(id_oficina_actual).setConexion_actual(1);
                oficinas_sistema.get(id_oficina_actual).setTiempo(0);
            }catch(Exception e){

            }
                System.out.println("\nVerificando conexion...");
                System.out.println("  Mensaje TEST recibida desde la "
                        + "oficina "
                        +oficinas_sistema.get(id_oficina_actual));
                System.out.println("  Mensaje CONEXION enviado a la "
                        + "oficina "
                        +oficinas_sistema.get(id_oficina_actual));
        }else if(mensaje_recibido.equals("desbloqueado")){

            for(i=0;i<oficinas_sistema.size();i++){
                if(Network.ip_envio.getHostAddress().equalsIgnoreCase(
                    oficinas_sistema.get(i).getOcteto1()+"."+
                    oficinas_sistema.get(i).getOcteto2()+"."+
                    oficinas_sistema.get(i).getOcteto3()+"."+
                    oficinas_sistema.get(i).getOcteto4())){

                    id_oficina_actual=i;
                }
            }
            try{
                oficinas_sistema.get(id_oficina_actual).setTiempo(0);
                oficinas_sistema.get(id_oficina_actual).
                        setBloqueado_actual(0);
            }catch (Exception e){
                StringTokenizer to=new StringTokenizer(Network.ip_envio.
                        getHostAddress(),".");
                oct1=Integer.parseInt(to.nextToken());
                oct2=Integer.parseInt(to.nextToken());
                oct3=Integer.parseInt(to.nextToken());
                oct4=Integer.parseInt(to.nextToken());
                IP[0]=(byte)oct1;
                IP[1]=(byte)oct2;
                IP[2]=(byte)oct3;
                IP[3]=(byte)oct4;

                if(oficinas_sistema.size()==0){

                    for(i=0;i<oficinas_existentes.size();i++){
                        if(Network.ip_envio.getHostAddress().equals(
                                oficinas_existentes.get(i).getOcteto1()+"."+
                                oficinas_existentes.get(i).getOcteto2()+"."+
                                oficinas_existentes.get(i).getOcteto3()+
                                "."+oficinas_existentes.get(i).getOcteto4())){

                            oficinas_sistema.add(new Oficina(
                            oficinas_existentes.get(i).getNumero(),oct1,
                            oct2,oct3,oct4,oficinas_existentes.get(i).
```

```
                    getIdOficina(),0,null,null,0,"",0,0,0,0,null,
                    oficinas_existentes.get(i).getEstado(),
                    oficinas_existentes.get(i).getNombre()));
                }
            }
        }else{
            for(i=0;i<oficinas_sistema.size();i++){
                if(Network.ip_envio.getHostAddress().equals(
                    oficinas_sistema.get(i).getOcteto1()+"."+
                    oficinas_sistema.get(i).getOcteto2()+"."+
                    oficinas_sistema.get(i).getOcteto3()+
                    "."+oficinas_sistema.get(i).getOcteto4())){
                    id_oficina_actual=i;
                    existe_sistema=1;
                }
            }
            if(existe_sistema==0){
                for(i=0;i<oficinas_existentes.size();i++){
                    if(Network.ip_envio.getHostAddress().equals(
                        oficinas_existentes.get(i).getOcteto1()+"."+
                        oficinas_existentes.get(i).getOcteto2()+"."+
                        oficinas_existentes.get(i).getOcteto3()+
                        "."+oficinas_existentes.get(i).
                        getOcteto4())){

                        oficinas_sistema.add(new Oficina(
                        oficinas_existentes.get(i).getNumero(),oct1,
                        oct2,oct3,oct4,oficinas_existentes.get(i).
                        getIdOficina(),0,null,null,0,"",0,0,0,0,null,
                        oficinas_existentes.get(i).getEstado(),
                        oficinas_existentes.get(i).getNombre()));
                    }
                }
            }
        }
        for(i=0;i<oficinas_sistema.size();i++){
            if(Network.ip_envio.getHostAddress().equalsIgnoreCase(
                oficinas_sistema.get(i).getOcteto1()+"."+
                oficinas_sistema.get(i).getOcteto2()+"."+
                oficinas_sistema.get(i).getOcteto3()+"."+
                oficinas_sistema.get(i).getOcteto4())){

                id_oficina_actual=i;
            }
        }
        oficinas_sistema.get(id_oficina_actual).setTiempo(0);
        oficinas_sistema.get(id_oficina_actual).
                setBloqueado_actual(0);
    }

    DesbloquearOficinas.hilo1.stop();
    oficinas_sistema.get(id_oficina_actual).setIntentos(0);
    DesbloquearOficina desbloquear_Oficina=new DesbloquearOficina(
            oficinas_sistema.get(id_oficina_actual).getNumero());
```

```
            fila_bloquear=new Object[6];
            fecha=new GregorianCalendar();
            mes=fecha.get(Calendar.MONTH);
            mes=mes+1;
            fila_bloquear[0]=numero_registro2;
            fila_bloquear[1]=fecha.get(Calendar.YEAR)+"-0"+mes+"-"+fecha.
                    get(Calendar.DAY_OF_MONTH);
            fila_bloquear[2]=fecha.get(Calendar.HOUR_OF_DAY)+":"+fecha.
                    get(Calendar.MINUTE)+":"+fecha.get(Calendar.SECOND);
            fila_bloquear[3]=oficinas_sistema.get(id_oficina_actual).
                    getNumero();
            fila_bloquear[4]=oficinas_sistema.get(id_oficina_actual).
                    getNombre();
            fila_bloquear[5]="Oficina Desbloqueada";

            fila=new Object[8];
            fila[0]=numero_registro2;
            fila[1]=fecha.get(Calendar.YEAR)+"-0"+mes+"-"+fecha.get(
                    Calendar.DAY_OF_MONTH);
            fila[2]=fecha.get(Calendar.HOUR_OF_DAY)+":"+fecha.get(
                    Calendar.MINUTE)+":"+fecha.get(Calendar.SECOND);
            fila[3]="Adm";
            fila[4]="--";
            fila[5]="--";
            fila[6]=oficinas_sistema.get(id_oficina_actual).getNumero();
            fila[7]="Oficina Desbloqueada";

            modelo1.addRow(fila_bloquear);


            registro=new RegistroHistorial(fila,oficinas_sistema.get(
                    id_oficina_actual).getIdOficina());
            oficinas_sistema.get(id_oficina_actual).setFilas_bloquear(
                    null);
            oficinas_sistema.get(id_oficina_actual).setFilas(null);
            System.out.println("Mensaje DESBLOQUEADO recibido de la ofic"
                    + "ina "+oficinas_sistema.get(id_oficina_actual));
            System.out.println("Oficina "+oficinas_sistema.get(
                    id_oficina_actual)+" Desbloqueada\n");
            oficinas_sistema.get(id_oficina_actual).setImprimir_bloqueo(0);
            numero_registro2++;

        }else if(comas==2){

            for(i=0;i<oficinas_sistema.size();i++){

            try {

                    Connection conexion = null;
                    Class.forName("com.mysql.jdbc.Driver");
                    conexion = DriverManager.getConnection("jdbc:mysql://"
                            + "localhost/ethernet","root","123456");
                    conexion.setAutoCommit(false);
                    Statement s=conexion.createStatement();
```

```
            ////////////////////////////////////////
            CallableStatement prcProcedimientoAlmacenado3 =
                conexion.prepareCall("{ call ConsulLLave(?,?) }");
            prcProcedimientoAlmacenado3.setString(1,
                oficinas_sistema.get(i).getNumero());
            prcProcedimientoAlmacenado3.registerOutParameter(2,
                    Types.CHAR);
            prcProcedimientoAlmacenado3.execute();
            llave_base = prcProcedimientoAlmacenado3.getString(2);

            conexion.commit();

        } catch (SQLException ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(null, "Hubo un problema "
                    + "en la base de datos ", "Error SQL",
                    JOptionPane.ERROR_MESSAGE);
        } catch (Exception e1) {
            e1.printStackTrace();
            JOptionPane.showMessageDialog(null, "Existe un proble"
                + "ma interno", "Error",JOptionPane.ERROR_MESSAGE);
        }
        finally{

        }
        oficinas_sistema.get(i).setLlave(llave_base);
    }

    usuarios_existentes=new ArrayList<Usuario>();
    oficina_existente=0;
    usuario_existente=0;
    id_oficina_actual=-1;
    clave_correcta=0;
    usuario_autorizado_oficina=0;
    estado_usuario=0;
    idoficina=0;
    idusuario=0;
    i_estado=0;
    i_autorizado=0;
    i_existe_base=0;

    mensaje_imprimir=m.substring(0, 29);

    for(i=0;i<oficinas_sistema.size();i++){
        if(Network.ip_envio.getHostAddress().equalsIgnoreCase(
            oficinas_sistema.get(i).getOcteto1()+"."+
            oficinas_sistema.get(i).getOcteto2()+"."+
            oficinas_sistema.get(i).getOcteto3()+"."+
            oficinas_sistema.get(i).getOcteto4())){

            id_oficina_actual=i;
        }
    }

    tk=new StringTokenizer(mensaje_recibido,",");
```

```
usuario_recibido_cambio=tk.nextToken();

clave_anterior_cifrada=tk.nextToken();

try{
    clave_anterior=Descifrar_Clave.descifrar_clave(
            clave_anterior_cifrada,oficinas_sistema.get(
            id_oficina_actual).getLlave());
}catch (Exception e){
    StringTokenizer to=new StringTokenizer(Network.ip_envio.
            getHostAddress(),".");
    oct1=Integer.parseInt(to.nextToken());
    oct2=Integer.parseInt(to.nextToken());
    oct3=Integer.parseInt(to.nextToken());
    oct4=Integer.parseInt(to.nextToken());
    IP[0]=(byte)oct1;
    IP[1]=(byte)oct2;
    IP[2]=(byte)oct3;
    IP[3]=(byte)oct4;

    if(oficinas_sistema.size()==0){

        for(i=0;i<oficinas_existentes.size();i++){
            if(Network.ip_envio.getHostAddress().equals(
                oficinas_existentes.get(i).getOcteto1()+"."+
                oficinas_existentes.get(i).getOcteto2()+"."+
                oficinas_existentes.get(i).getOcteto3()+
                "."+oficinas_existentes.get(i).getOcteto4())){

                oficinas_sistema.add(new Oficina(
                oficinas_existentes.get(i).getNumero(),oct1,oct2
                ,oct3,oct4,oficinas_existentes.get(i).
                getIdOficina(),0,null,null,0,"",0,0,0,0,null,
                oficinas_existentes.get(i).getEstado(),
                oficinas_existentes.get(i).getNombre()));
            }
        }
    }else{
        for(i=0;i<oficinas_sistema.size();i++){
            if(Network.ip_envio.getHostAddress().equals(
                    oficinas_sistema.get(i).getOcteto1()+"."+
                    oficinas_sistema.get(i).getOcteto2()+"."+
                    oficinas_sistema.get(i).getOcteto3()+
                    "."+oficinas_sistema.get(i).getOcteto4())){
                id_oficina_actual=i;
                existe_sistema=1;
            }
        }
        if(existe_sistema==0){
            for(i=0;i<oficinas_existentes.size();i++){
                if(Network.ip_envio.getHostAddress().equals(
                    oficinas_existentes.get(i).getOcteto1()+"."+
                    oficinas_existentes.get(i).getOcteto2()+"."+
                    oficinas_existentes.get(i).getOcteto3()+
                    "."+oficinas_existentes.get(i).getOcteto4())){
```

```
                    oficinas_sistema.add(new Oficina(
                    oficinas_existentes.get(i).getNumero(),oct1,
                    oct2,oct3,oct4,oficinas_existentes.get(i).
                    getIdOficina(),0,null,null,0,"",0,0,0,0,null,
                    oficinas_existentes.get(i).getEstado(),
                    oficinas_existentes.get(i).getNombre()));
                }
            }
        }
    }
    for(i=0;i<oficinas_sistema.size();i++){
        if(Network.ip_envio.getHostAddress().equalsIgnoreCase(
            oficinas_sistema.get(i).getOcteto1()+"."+
            oficinas_sistema.get(i).getOcteto2()+"."+
            oficinas_sistema.get(i).getOcteto3()+"."+
            oficinas_sistema.get(i).getOcteto4())){

            id_oficina_actual=i;
        }
    }
    oficinas_sistema.get(id_oficina_actual).
            setConexion_actual(1);
    try {

        Connection conexion = null;
        Class.forName("com.mysql.jdbc.Driver");
        conexion = DriverManager.getConnection("jdbc:mysql://"
                + "localhost/ethernet","root","123456");
        conexion.setAutoCommit(false);
        Statement s=conexion.createStatement();

        ////////////////////////////////////////
        CallableStatement prcProcedimientoAlmacenado3 =
            conexion.prepareCall("{ call ConsulLLave(?,?) }");
        prcProcedimientoAlmacenado3.setString(1,
            oficinas_sistema.get(id_oficina_actual).
            getNumero());
        prcProcedimientoAlmacenado3.registerOutParameter(2,
            Types.CHAR);
        prcProcedimientoAlmacenado3.execute();
        llave_base = prcProcedimientoAlmacenado3.getString(2);

        conexion.commit();

    } catch (SQLException ex) {
        ex.printStackTrace();
        JOptionPane.showMessageDialog(null, "Hubo un problema "
                + "en la base de datos ", "Error SQL",
                JOptionPane.ERROR_MESSAGE);
    } catch (Exception e1) {
        e1.printStackTrace();
        JOptionPane.showMessageDialog(null, "Existe un "
                + "problema interno", "Error",
                JOptionPane.ERROR_MESSAGE);
```

```
    }
    finally{

    }
    oficinas_sistema.get(id_oficina_actual).setLlave(
            llave_base);
    oficinas_sistema.get(id_oficina_actual).setTiempo(0);
    clave_anterior=Descifrar_Clave.descifrar_clave(
            clave_anterior_cifrada,oficinas_sistema.get(
            id_oficina_actual).getLlave());

}
System.out.println("\nSolicitud de cambio de clave iniciada."
        + "..");
System.out.println("  Mensaje: "+mensaje_imprimir+" recibida "
            +"desde la oficina "
            +oficinas_sistema.get(id_oficina_actual).getNumero());
System.out.println("  ID: "+usuario_recibido_cambio);
System.out.println("  Clave anterior: "+clave_anterior);

clave_nueva_cambio=tk.nextToken().substring(0, 12);

clave_nueva_cambio=Descifrar_Clave.descifrar_clave(
    clave_nueva_cambio,oficinas_sistema.get(id_oficina_actual).
    getLlave());
System.out.println("  Clave nueva: "+
    clave_nueva_cambio);


for(i=0;i<oficinas_sistema.size();i++){
    if(Network.ip_envio.getHostAddress().equalsIgnoreCase(
        oficinas_sistema.get(i).getOcteto1()+"."+
        oficinas_sistema.get(i).getOcteto2()+"."+
        oficinas_sistema.get(i).getOcteto3()+"."+
        oficinas_sistema.get(i).getOcteto4())){

        //System.out.println("Oficina existente");
        oficina_existente=1;
        id_oficina_actual=i;
        idoficina=oficinas_sistema.get(i).getIdOficina();
        oficina=oficinas_sistema.get(i).getNumero();

        IP[0]=(byte)oficinas_sistema.get(i).getOcteto1();
        IP[1]=(byte)oficinas_sistema.get(i).getOcteto2();
        IP[2]=(byte)oficinas_sistema.get(i).getOcteto3();
        IP[3]=(byte)oficinas_sistema.get(i).getOcteto4();
    }
}
oficinas_sistema.get(id_oficina_actual).setTiempo(0);

Database db1 = new Database();

Object[][] data1 = db1.select("TodosUsuarios", "User, Clave, "
        + "idUsuario, Estado,Nombres, Apellidos", null);
```

```
        for(i=0;i<data1.length;i++){
            usuarios_existentes.add(new Usuario(data1[i][0].toString(),
                data1[i][1].toString(),Integer.parseInt(data1[i][2].
                toString()),data1[i][3].toString(),data1[i][4].
                toString(),data1[i][5].toString()));
        }

        for(i=0;i<usuarios_existentes.size();i++){
            if(usuarios_existentes.get(i).getUser().equals(
                usuario_recibido_cambio)){
                usuario_existente=1;
                idusuario=usuarios_existentes.get(i).getIdUsuario();

                i_existe_base=1;
            }
            else{
            }
        }
        if(i_existe_base==1){
            System.out.println("  Usuario existe en la base de datos");
        }else{
            System.out.println("  Usuario no existe en la base "
                    + "de datos");
        }

        if(usuario_existente==1){
            for(i=0;i<usuarios_existentes.size();i++){
                if(usuarios_existentes.get(i).getUser().equals(
                    usuario_recibido_cambio)){
                    if(usuarios_existentes.get(i).getClave().equals(
                        clave_anterior)){
                        System.out.println("  Clave anterior CORRE"
                                + "CTA");
                        clave_correcta=1;
                    }
                    else if(!usuarios_existentes.get(i).getClave().
                        equals(clave_anterior)){
                        System.out.println("  Clave anterior INCORREC"
                                + "TA");
                    }
                }
            }
        }

        if(usuario_existente==1&&clave_correcta==1){
            Network.Enviar_Paquetes("cambiado",IP);
            System.out.println("  Mensaje CAMBIADO enviado a la "
                + "oficina "+oficinas_sistema.get(id_oficina_actual));
        }

    }else{

    usuarios_existentes=new ArrayList<Usuario>();

    oficina_existente=0;
```

```
mensaje_imprimir=m.substring(0,16);
id_oficina_actual=-1;
id_usuario_actual=-1;
usuario_existente=0;
clave_correcta=0;
usuario_autorizado_oficina=0;
estado_usuario=0;
idoficina=0;
idusuario=0;
i_estado=0;
i_autorizado=0;
i_existe_base=0;

for(i=0;i<oficinas_sistema.size();i++){
    if(Network.ip_envio.getHostAddress().equalsIgnoreCase(
        oficinas_sistema.get(i).getOcteto1()+"."+
        oficinas_sistema.get(i).getOcteto2()+"."+oficinas_sistema.
        get(i).getOcteto3()+"."+
        oficinas_sistema.get(i).getOcteto4())){

        id_oficina_actual=i;
    }
}

for(i=0;i<oficinas_sistema.size();i++){

    try {

        Connection conexion = null;
        Class.forName("com.mysql.jdbc.Driver");
        conexion = DriverManager.getConnection("jdbc:mysql://"
                + "localhost/ethernet","root","123456");
        conexion.setAutoCommit(false);
        Statement s=conexion.createStatement();

        ////////////////////////////////////////
        CallableStatement prcProcedimientoAlmacenado3 = conexion.
                prepareCall("{ call ConsulLLave(?,?) }");
        prcProcedimientoAlmacenado3.setString(1,oficinas_sistema.
                get(i).getNumero());
        prcProcedimientoAlmacenado3.registerOutParameter(2,
                Types.CHAR);
        prcProcedimientoAlmacenado3.execute();
        llave_base = prcProcedimientoAlmacenado3.getString(2);

        conexion.commit();

    } catch (SQLException ex) {
        ex.printStackTrace();
        JOptionPane.showMessageDialog(null, "Hubo un problema en "
                + "la base de datos ", "Error SQL",
                JOptionPane.ERROR_MESSAGE);
    } catch (Exception e1) {
        e1.printStackTrace();
        JOptionPane.showMessageDialog(null, "Existe un problema "
```

```
                + "interno", "Error",JOptionPane.ERROR_MESSAGE);
    }
    finally{

    }
    oficinas_sistema.get(i).setLlave(llave_base);
}

tk=new StringTokenizer(mensaje_recibido,",");
usuario_recibido=tk.nextToken();


clave_cifrada=tk.nextToken().substring(0, 12);

try{
    clave_recibida=Descifrar_Clave.descifrar_clave(clave_cifrada,
            oficinas_sistema.get(id_oficina_actual).getLlave());
}catch(Exception e){
        StringTokenizer to=new StringTokenizer(Network.ip_envio.
                getHostAddress(),".");
        oct1=Integer.parseInt(to.nextToken());
        oct2=Integer.parseInt(to.nextToken());
        oct3=Integer.parseInt(to.nextToken());
        oct4=Integer.parseInt(to.nextToken());
        IP[0]=(byte)oct1;
        IP[1]=(byte)oct2;
        IP[2]=(byte)oct3;
        IP[3]=(byte)oct4;

        if(oficinas_sistema.size()==0){

            for(i=0;i<oficinas_existentes.size();i++){
                if(Network.ip_envio.getHostAddress().equals(
                        oficinas_existentes.get(i).getOcteto1()+"."+
                        oficinas_existentes.get(i).getOcteto2()+"."+
                        oficinas_existentes.get(i).getOcteto3()+
                        "."+oficinas_existentes.get(i).getOcteto4())){

                    oficinas_sistema.add(new Oficina(
                    oficinas_existentes.get(i).getNumero(),oct1,oct2,
                    oct3,oct4,oficinas_existentes.get(i).
                    getIdOficina(),0,null,null,0,"",0,0,0,0,null,
                    oficinas_existentes.get(i).getEstado(),
                    oficinas_existentes.get(i).getNombre()));
                }
            }
        }else{
            for(i=0;i<oficinas_sistema.size();i++){
                if(Network.ip_envio.getHostAddress().equals(
                        oficinas_sistema.get(i).getOcteto1()+"."+
                        oficinas_sistema.get(i).getOcteto2()+"."+
                        oficinas_sistema.get(i).getOcteto3()+
                        "."+oficinas_sistema.get(i).getOcteto4())){
                    id_oficina_actual=i;
                    existe_sistema=1;
```

```
            }
        }
        if(existe_sistema==0){
            for(i=0;i<oficinas_existentes.size();i++){
                if(Network.ip_envio.getHostAddress().equals(
                    oficinas_existentes.get(i).getOcteto1()+"."+
                    oficinas_existentes.get(i).getOcteto2()+"."+
                    oficinas_existentes.get(i).getOcteto3()+
                    "."+oficinas_existentes.get(i).getOcteto4())){

                    oficinas_sistema.add(new Oficina(
                    oficinas_existentes.get(i).getNumero(),oct1,
                    oct2,oct3,oct4,oficinas_existentes.get(i).
                    getIdOficina(),0,null,null,0,"",0,0,0,0,null,
                    oficinas_existentes.get(i).getEstado(),
                    oficinas_existentes.get(i).getNombre()));
                }
            }
        }
    }
    for(i=0;i<oficinas_sistema.size();i++){
        if(Network.ip_envio.getHostAddress().equalsIgnoreCase(
            oficinas_sistema.get(i).getOcteto1()+"."+
            oficinas_sistema.get(i).getOcteto2()+"."+
            oficinas_sistema.get(i).getOcteto3()+"."+
            oficinas_sistema.get(i).getOcteto4())){

            id_oficina_actual=i;
        }
    }
    oficinas_sistema.get(id_oficina_actual).
            setConexion_actual(1);
    try {

        Connection conexion = null;
        Class.forName("com.mysql.jdbc.Driver");
        conexion = DriverManager.getConnection("jdbc:mysql://"
                + "localhost/ethernet","root","123456");
        conexion.setAutoCommit(false);
        Statement s=conexion.createStatement();

        ////////////////////////////////////////
        CallableStatement prcProcedimientoAlmacenado3 =
            conexion.prepareCall("{ call ConsulLLave(?,?) }");
        prcProcedimientoAlmacenado3.setString(1,
            oficinas_sistema.get(id_oficina_actual).getNumero());
        prcProcedimientoAlmacenado3.registerOutParameter(2,
            Types.CHAR);
        prcProcedimientoAlmacenado3.execute();
        llave_base = prcProcedimientoAlmacenado3.getString(2);

        conexion.commit();

    } catch (SQLException ex) {
        ex.printStackTrace();
```

```
                JOptionPane.showMessageDialog(null, "Hubo un problema"
                        + " en la base de datos ", "Error SQL",
                        JOptionPane.ERROR_MESSAGE);
            } catch (Exception e1) {
                e1.printStackTrace();
                JOptionPane.showMessageDialog(null, "Existe un probl"
                        + "ema interno", "Error",
                        JOptionPane.ERROR_MESSAGE);
            }
            finally{

            }
            oficinas_sistema.get(id_oficina_actual).setLlave(
                llave_base);
            oficinas_sistema.get(id_oficina_actual).setTiempo(0);
                clave_recibida=Descifrar_Clave.descifrar_clave(
                clave_cifrada,oficinas_sistema.get(id_oficina_actual).
                getLlave());
    }
    System.out.println("\nSolicitud de autenticacion iniciada...");
    System.out.println("  Mensaje: "+mensaje_imprimir
        + " recibida desde la oficina "+
            oficinas_sistema.get(id_oficina_actual));
    System.out.println("  ID usuario: "+usuario_recibido);
    System.out.println("  Clave descifrada: "+clave_recibida);

    for(i=0;i<oficinas_sistema.size();i++){
        if(Network.ip_envio.getHostAddress().equalsIgnoreCase(
            oficinas_sistema.get(i).getOcteto1()+"."+
            oficinas_sistema.get(i).getOcteto2()+"."+
            oficinas_sistema.get(i).getOcteto3()+"."+
            oficinas_sistema.get(i).getOcteto4())){

            oficina_existente=1;
            id_oficina_actual=i;
            idoficina=oficinas_sistema.get(i).getIdOficina();
            oficina=oficinas_sistema.get(i).getNumero();
            IP[0]=(byte)oficinas_sistema.get(i).getOcteto1();
            IP[1]=(byte)oficinas_sistema.get(i).getOcteto2();
            IP[2]=(byte)oficinas_sistema.get(i).getOcteto3();
            IP[3]=(byte)oficinas_sistema.get(i).getOcteto4();
        }
    }
    oficinas_sistema.get(id_oficina_actual).setTiempo(0);

    Database db1 = new Database();

    Object[][] data1 = db1.select("TodosUsuarios", "User, Clave,"
            + " idUsuario, Estado,Nombres,Apellidos", null);

    for(i=0;i<data1.length;i++){
        usuarios_existentes.add(new Usuario(data1[i][0].toString(),
            data1[i][1].toString(),Integer.parseInt(data1[i][2].
            toString()),data1[i][3].toString(),data1[i][4].toString(),
            data1[i][5].toString()));
```

```
}

for(i=0;i<usuarios_existentes.size();i++){
    if(usuarios_existentes.get(i).getUser().equals(
        usuario_recibido)){

        usuario_existente=1;
        id_usuario_actual=i;
        nombre_presentar=usuarios_existentes.get(i).
                getNombres_usuario();
        apellido_presentar=usuarios_existentes.get(i).
                getApellidos_usuario();
        idusuario=usuarios_existentes.get(i).getIdUsuario();

        for(i1=0;i1<oficinas_sistema.size();i1++){
            if(Network.ip_envio.getHostAddress().equalsIgnoreCase(
                oficinas_sistema.get(i1).getOcteto1()+"."+
                oficinas_sistema.get(i1).getOcteto2()+"."+
                oficinas_sistema.get(i1).getOcteto3()+"."+
                oficinas_sistema.get(i1).getOcteto4())){

                oficinas_sistema.get(i1).setIdusuario_paquete(
                    usuarios_existentes.get(i1).getIdUsuario());
                oficinas_sistema.get(i1).setUser_paquete(
                    usuario_recibido);
            }
        }
        i_existe_base=1;
    }
    else{
    }
}

if(i_existe_base==1){
}else{
}

Database db2 = new Database();

Object[][] data2 = db2.select("TodosOficinasUsuarios", ""
        + "idOficina_Usuario, idOficina, idUsuario", null);

for(i=0;i<data2.length;i++){
    if(Integer.parseInt(data2[i][1].toString())==idoficina&&
            Integer.parseInt(data2[i][2].toString())==idusuario){

        usuario_autorizado_oficina=1;
        i_autorizado=1;
    }
    else{
    }
}

for(i=0;i<usuarios_existentes.size();i++){
    if(usuarios_existentes.get(i).getUser().equals(
```

```
            usuario_recibido)&&usuarios_existentes.get(i).
            getEstado().equals("Activo")){

        estado_usuario=1;
        i_estado=1;
    }
    else{
    }
}
    if(oficina_existente==1&&usuario_existente==0){
    System.out.println("  Usuario NO EXISTENTE en la base de "
            + "datos");
}
if(estado_usuario==1){
    System.out.println("  Estado del usuario ACTIVO");
}else{
    System.out.println("  Estado del usuario INACTIVO");
}
    if(i_autorizado==1){
    System.out.println("  Facultad de ingreso PERMITIDO");
}else{
    System.out.println("  Facultad de ingreso DENEGADO");
    }

if(oficina_existente==1&&usuario_existente==1){
    for(i=0;i<usuarios_existentes.size();i++){
        if(usuarios_existentes.get(i).getUser().equals(
            usuario_recibido)){

            if(usuarios_existentes.get(i).getClave().equals(
                    clave_recibida)){
                System.out.println("  Clave ingresada CORRECTA");
                clave_correcta=1;
            }
            else if(!usuarios_existentes.get(i).getClave().equals(
                    clave_recibida)){
                System.out.println("  Clave ingresada INCORRECTA");
            }
        }
    }
}


if(oficina_existente==1&&usuario_existente==0){
    //Mandar paquete a la tarjeta con un mensaje de autorizado
    Network.Enviar_Paquetes("idincorrecto",IP);
    System.out.println("  Mensaje IDINCORRECTO enviado a la "
            + "oficina"
            + " "+oficinas_sistema.get(id_oficina_actual));
}

if(oficina_existente==1&&usuario_existente==1&&clave_correcta==1&&
        usuario_autorizado_oficina==1&&estado_usuario==1){

    Network.Enviar_Paquetes("autorizado",IP);
```

```
        System.out.println("  Mensaje AUTORIZADO enviado a la oficina"
                + " "+oficinas_sistema.get(id_oficina_actual));
        tiempo_respuesta=new GregorianCalendar();

        fecha=new GregorianCalendar();
        mes=fecha.get(Calendar.MONTH);
        mes=mes+1;
        for(i=0;i<oficinas_sistema.size();i++){
            if(Network.ip_envio.getHostAddress().equalsIgnoreCase(
                    oficinas_sistema.get(i).getOcteto1()+"."+
                oficinas_sistema.get(i).getOcteto2()+"."+
                oficinas_sistema.get(i).getOcteto3()+"."+
                oficinas_sistema.get(i).getOcteto4())){

                fila=new Object[8];
                fila[0]=numero_registro;
                fila[1]=fecha.get(Calendar.YEAR)+"-0"+mes+"-"+fecha.
                        get(Calendar.DAY_OF_MONTH);
                fila[2]=fecha.get(Calendar.HOUR_OF_DAY)+":"+fecha.
                        get(Calendar.MINUTE)+":"
                        +fecha.get(Calendar.SECOND);
                fila[3]=usuario_recibido;
                fila[4]=nombre_presentar;
                fila[5]=apellido_presentar;
                fila[6]=oficinas_sistema.get(id_oficina_actual).
                        getNombre();
                fila[7]="Autorizado";
                oficinas_sistema.get(i).setFilas(fila);
            }
        }
    }
    else if(oficina_existente==1&&usuario_existente==1&&(
            clave_correcta==0||
            usuario_autorizado_oficina==0||estado_usuario==0)&&
            oficinas_sistema.get(id_oficina_actual).getIntentos()<2){

        Network.Enviar_Paquetes("denegado",IP);
        System.out.println("  Mensaje DENEGADO enviado a la oficina"
                + " "+oficinas_sistema.get(id_oficina_actual));
        tiempo_respuesta=new GregorianCalendar();

        fecha=new GregorianCalendar();
        mes=fecha.get(Calendar.MONTH);
        mes=mes+1;
        for(i=0;i<oficinas_sistema.size();i++){

            if(Network.ip_envio.getHostAddress().equalsIgnoreCase(

                oficinas_sistema.get(i).getOcteto1()+"."+
                oficinas_sistema.get(i).getOcteto2()+"."+
                oficinas_sistema.get(i).getOcteto3()+"."+
                oficinas_sistema.get(i).getOcteto4())){
                fila=new Object[8];
                fila[0]=numero_registro;
                fila[1]=fecha.get(Calendar.YEAR)+"-0"+mes+"-"+fecha.
```

```
                        get(Calendar.DAY_OF_MONTH);
                fila[2]=fecha.get(Calendar.HOUR_OF_DAY)+":"+fecha.
                        get(Calendar.MINUTE)+":"
                        +fecha.get(Calendar.SECOND);
                fila[3]=usuario_recibido;
                fila[4]=nombre_presentar;
                fila[5]=apellido_presentar;
                fila[6]=oficinas_sistema.get(id_oficina_actual).
                        getNombre();
                fila[7]="Denegado";
                oficinas_sistema.get(i).setFilas(fila);
            }
        }
    }

    else if(oficina_existente==1&&usuario_existente==1&&(
            clave_correcta==0||
            usuario_autorizado_oficina==0||estado_usuario==0)&&
            oficinas_sistema.get(id_oficina_actual).getIntentos()==2){

        Network.Enviar_Paquetes("bloquear",IP);
        System.out.println("  Mensaje BLOQUEAR enviado a la oficina"
                + " "+oficinas_sistema.get(id_oficina_actual));
        tiempo_respuesta=new GregorianCalendar();

        fecha=new GregorianCalendar();
        mes=fecha.get(Calendar.MONTH);
        mes=mes+1;
        for(i=0;i<oficinas_sistema.size();i++){

            if(Network.ip_envio.getHostAddress().equalsIgnoreCase(

                oficinas_sistema.get(i).getOcteto1()+"."+
                oficinas_sistema.get(i).getOcteto2()+"."+
                oficinas_sistema.get(i).getOcteto3()+
                "."+oficinas_sistema.get(i).getOcteto4())){

                fila_bloquear=new Object[6];
                fila_bloquear[0]=numero_registro2;
                fila_bloquear[1]=fecha.get(Calendar.YEAR)+"-0"+mes+
                        "-"+fecha.get(Calendar.DAY_OF_MONTH);
                fila_bloquear[2]=fecha.get(Calendar.HOUR_OF_DAY)+":"
                        +fecha.get(Calendar.MINUTE)+":"
                        +fecha.get(Calendar.SECOND);
                fila_bloquear[3]=oficina;
                fila_bloquear[4]=oficinas_sistema.get(
                        id_oficina_actual).getNombre();
                fila_bloquear[5]="Oficina Bloqueada";
                oficinas_sistema.get(i).setFilas_bloquear(
                        fila_bloquear);

                fila=new Object[8];
                fila[0]=numero_registro2;
                fila[1]=fecha.get(Calendar.YEAR)+"-0"+mes+"-"+fecha.
                        get(Calendar.DAY_OF_MONTH);
```

```
                        fila[2]=fecha.get(Calendar.HOUR_OF_DAY)+":"+fecha.
                                get(Calendar.MINUTE)+":"
                                +fecha.get(Calendar.SECOND);
                        fila[3]=usuarios_existentes.get(id_usuario_actual).
                                getUser();
                        fila[4]=usuarios_existentes.get(id_usuario_actual).
                                getNombres_usuario();
                        fila[5]=usuarios_existentes.get(id_usuario_actual).
                                getApellidos_usuario();
                        fila[6]=oficina;
                        fila[7]="Oficina Bloqueada";
                        oficinas_sistema.get(i).setFilas(fila);
                    }
                }
            }
        }
        }
    }
```

ANEXO D

DISEÑO PCB

ADAPTADOR DEL TECLADO MATRICIAL 4X4

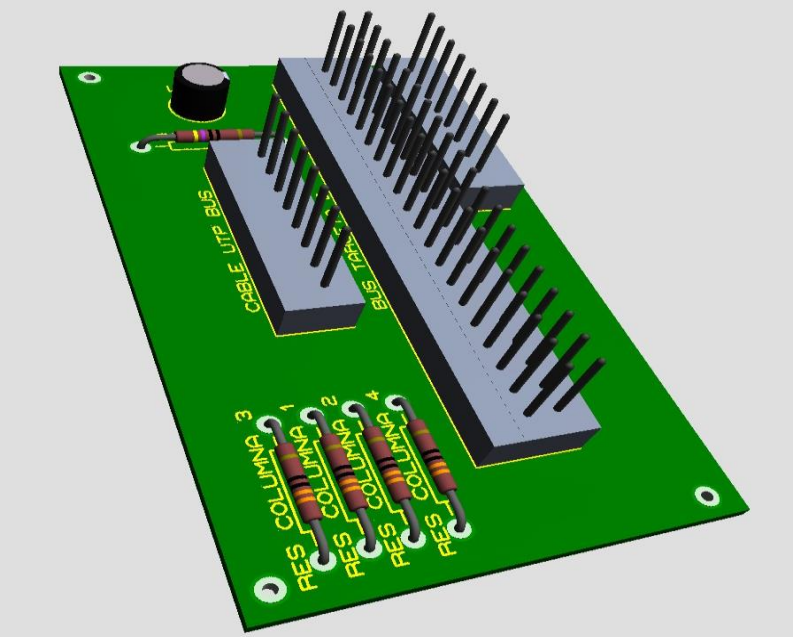


Figura D.1 – Adaptador del teclado matricial 4x4.

BIBLIOGRAFÍA

[1] Instituto Nacional de Tecnología Industrial, Simposio Argentino de Sistemas Embebidos, http://www.inti.gob.ar/noticias/slide_simposio.htm, Marzo 2014

[2] Biocomputación y Física de Sistemas Complejos, Las FPGA, http://oldwww.bifi.es/research/complexm_fundphysics/ssue/ssue_es.php, Marzo 2014

[3] Altera, Cyclone II FPGAs, http://www.altera.com/devices/fpga/cyclone2/cy2-index.jsp, Marzo 2014

[4] Altera Corporation, Altera University Program, Cyclone II Device Family Data Sheet – Página 5, Características del FPGA Cyclone II, Marzo 2014

[5] Genera, Sistemas Embebidos sobre FPGA, http://www.generatecnologias.es/sistemas_embebidos_fpga.html, Marzo 2014

[6] Hightech Letitbit, Procesador Emebido NIOS II, http://hightech.letitbit.net/archive.shtml?2006/0712, Marzo 2014

[7] Hightech Letitbit, Procesador Emebido NIOS II, http://hightech.letitbit.net/archive.shtml?2006/0712, Marzo 2014

[8] Altera, Arquitectura del Sistema NIOS II, http://www.altera.com.cn/support/examples/nios2/exm-system-architect.html, Marzo 2014

[9] Electronic Development Group, Tarjeta de Desarrollo DE2 de Altera, http://electronicdevelopmentgroup.wordpress.com/synchronous-generator-control-nios-ii-altera-de2/, Marzo 2014

[10] Electronic Development Group, Tarjeta de Desarrollo DE2 de Altera, http://electronicdevelopmentgroup.wordpress.com/synchronous-generator-control-nios-ii-altera-de2/, Marzo 2014

[11] Sistemas Digitales Electrónica, Programación de la tarjeta DE2 de Altera, http://digitales-itesi.blogspot.com/2010/08/ejemplo-2-vhdl-fpga-programacion-de-la.html, Marzo 2014

[12] Ebay, Módulo LCD 16x2, http://www.ebay.com/itm/DigiTron-SC162A-16-X-2-16X2-LCD-Module-Blue-Backlight-/200379391207, Marzo 2014

[13] Dacomwest, Chip Davicom DM9000A, http://beta.dacomwest.de/eng/e_etherics_mac_dm9000a.htm, Marzo 2014

[14] Altera Corporation, Altera University Program, Ethernet Controller DM9000A Data Sheet – Página 30, Diagramas de Bloques del DM9000A, Marzo 2014

[15] Electronilab, Conector GPIO, http://electronilab.co/tienda/kit-conector-gpio-raspberry-pi-para-protoboard-ensamblado/, Marzo 2014

[16] AppInformatica, Switch DLink de 8 puertos, http://www.appinformatica.com/hubs-switch-dlink-switch-8-puertos-10-100-(des1008d).php, Marzo 2014

[17] ICIDU, Cable UTP CAT 5e, http://www.icidu.com/en/cables/network/utp-cat5e-cable-1m.html, Marzo 2014

[18] Electrónica60Norte, Teclado Matricial 4x4, http://www.electronica60norte.com/detalle.php?sku=492, Marzo 2014

[19] El Rincón del Vago, Módelo OSI y TCP/IP, http://html.rincondelvago.com/modelo-osi-y-tcpip.html, Marzo 2014

[20] Redes y Telecomunicaciones, El módelo de referencia OSI, http://redes2010.wordpress.com/modelos/, Marzo 2014

[21] Redes de Computadoras, Formato de la Trama Ethernet, http://redesdecomputadores.umh.es/enlace/ethernet/Formato_Trama_ethernet.html, Marzo 2014

[22] Apuntes de Networking, El Protocolo ARP. Protocolo de resolución de direcciones, http://apuntesdenetworking.blogspot.com/2011/07/el-protocolo-arp-protocolo-de.html, Marzo 2014

[23] Universidad Politecnica de Valencia, Protocolo de Internet, http://personales.upv.es/rmartin/TcpIp/cap02s03.html, Marzo 2014

[24] Universidad Politecnica de Valencia, Protocolo de Datagrmas de Usuario, http://personales.upv.es/rmartin/TcpIp/cap02s11.html, Marzo 2014

[25] Wikispaces, Comunicación entre Sockets, http://sistemas-distribuidos-iucpr.wikispaces.com/7.1.1+SOCKETS, Marzo 2014

[26] Instituto Nacional de Tecnologías de la Comunicación, Secutity Issues, http://www.inteco.es/Training/security_issues/, Marzo 2014

[27] Laurier Canadian Excellence, Quartus II tutorial, http://denethor.wlu.ca/quartus/, Marzo 2014

[28] Application of Electronic Technique, Qsys tutorial, http://blog.chinaaet.com/detail/23329, Marzo 2014

[29] Realtime Embedded, Eclipse tutorial, http://www.rte.se/blog/blogg-modesty-corex/nios-ii-embeddedesign-suite-eds/3.4, Marzo 2014

[30] El blog de Arielb, Instalación de Netbeans, http://blog.arielb.com/, Marzo 2014

[31] Pedro Ventura, Crear un esquema EER desde el gestor de base de datos MySQL Workbench, http://www.pedroventura.com/gestion-de-proyecto/crear-un-esquema-eer-desde-el-gestor-de-base-de-datos-mysql-workbench/, Marzo 2014

[32] Tutoelectro, Simulación de circuitos con Proteus, http://tutoelectro.wikispaces.com/Simulaci%C3%B3n+con+Proteus, Marzo 2014

[33] Bloringaz, Wireshark para análisis de redes, http://bloringaz.com/descargar-wireshark-para-analisis-de-redes/, Marzo 2014

[34] Foros de Altera, Librerías y Código en C de las funciones para transmitir y recibir datos usando el chip DM9000A, http://http//www.alteraforum.com/forum/showthread.php?t=3947, Marzo 2014

[35] Universidad de Columbia, Stephen A. Edwards, Embedded System Design Spring 2011, Sección Projects - Networked Paddle Game, http://www.cs.columbia.edu/~sedwas/classes/2011/4840/index.html, Marzo 2014