ESCUELA POLITÉCNICA DEL EJÉRCITO

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

PROYECTO DE GRADO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERÍA

DISEÑO E IMPLEMENTACIÓN DE UN CONTROL DE ACCESO PARA ASCENSORES MITSUBISHI MEDIANTE UN SISTEMA DE SEGURIDAD MAGNÉTICO PERSONAL (PSSM *Duo*)

ALEJANDRO SEBASTIÁN MERA BALSECA

GIOVANNY PAUL PADILLA CAZAR

SANGOLQUÍ – ECUADOR

2007

CERTIFICACIÓN

Certificamos que el presente proyecto fue realizado en su totalidad por los señores Alejandro Sebastián Mera Balseca y Giovanny Paul Padilla Cazar, como requerimiento parcial a la obtención del título de INGENIEROS ELECTRÓNICOS.

Sr. Ing. Hugo Ortiz
DIRECTOR

Sr. Ing. Rodolfo Gordillo
CODIRECTOR

RESUMEN

La realización de este proyecto involucra el diseño e implementación de un sistema de control de acceso para ascensores Mitsubishi mediante lectores de tarjetas magnéticas para la empresa COHECO Cia. Ltda. El sistema maneja dos interfases de control de acceso simultáneas e independientes, que comparten una capacidad de memoria para 4 000 usuarios junto con una base de datos de 40 000 registros de acceso, con módulos de salida independientes, modulares y expandibles hasta 32 puntos para cada interfaz, todo esto manejado por el controlador embebido RCM 3700, cuya características de conectividad provee al sistema la capacidad de comunicarse con un PC de manera local o remota mediante el protocolo TCP/IP para crear, modificar y respaldar las variables de funcionamiento del sistema utilizando un software de administración amigable completamente desarrollado en el lenguaje de programación Visual Basic 2005 Express, el cual además permite generar reportes basados en consultas sobre registros de acceso recopilados por el controlador.

La gran capacidad de almacenamiento de registros de acceso y usuarios con sus respectivas combinaciones de horarios y pisos de acceso, el manejo de sistemas independientes, simultáneos, modulares, rápidos, flexibles y finalmente la conectividad resumen las ventajas del sistema que cumplen con las características que COHECO Cia. Ltda. busca en este proyecto.

DEDICATORIA

A mi padre cuyo esfuerzo y dedicación se ven reflejados en mi realización personal, a mi madre por su eterno sacrifico, a mi hermano por sus palabras y consejos, a mi hermana por su compañía y cariño. A todos los amigos con los que compartí buenos momentos.

Alejandro Mera

A mi madre por su constante soporte en toda mi vida, a mi hermana por su fé sincera en mí, a mis abuelos por sus sabios consejos y cariño sin límites, y especialmente a mi padre pues hubiese sido imposible alcanzar este momento en mi vida sin su esfuerzo desinteresado, que al sacrificar valiosos momentos de su vida pudo darme esta oportunidad de triunfar en el mundo.

Para todos ellos va dedicado este esfuerzo que es símbolo del profundo agradecimiento y admiración hacia sus personas.

Paul Padilla

AGRADECIMIENTO

La realización de este proyecto hubiera sido imposible sin el apoyo de COHECO Cia. Ltda., especialmente por la confianza que el Ing. Jóse Sáenz tuvo en nosotros; para nuestro director, codirector y todos aquellos profesores que compartieron sus conocimientos y nos guiaron mientras duró nuestro paso por la Escuela Politécnica del Ejército, una eterna gratitud de nuestra parte.

Un agradecimiento imposible de olvidar es para Juan Carlos Flores que estuvo a nuestro lado desde el inicio de este proyecto.

PRÓLOGO

Los sistemas de seguridad se han convertido en un elemento muy común tanto en el sector urbano como industrial, siendo una de sus principales aplicaciones el Control de Accesos. El sistema a diseñar e implementar constituye una de estas aplicaciones.

Actualmente COHECO Cía. Ltda. utiliza un sistema de este tipo, pero se han detectado ciertas deficiencias en su desempeño, las cuales limitan su rendimiento y calidad.

COHECO se encuentra en una continua búsqueda para mejorar servicios y brinda su apoyo a la tecnología nacional, dando lugar a la generación de proyectos que aportan al desarrollo de la empresa y de la sociedad. Por estos motivos el proyecto propone un nuevo sistema de control de acceso para ascensores Mitsubishi mediante lectores de tarjetas magnéticas, con una amplia capacidad de almacenamiento de usuarios, manejo de horarios y salidas, comunicación con un PC mediante protocolo TCP/IP, y recopilación de registro de accesos realizados por los usuarios.

ÍNDICE DE CONTENIDO

1. CAPÍTULO 1 INTRODUCCIÓN 16
1.1 JUSTIFICACIÓN 18
1.2 ALCANCE DEL PROYECTO 19
1.3 CARACTERÍSTICAS PROPUESTAS PARA EL SISTEMA 19

2 CAPÍTULO 2 MARCO TEÓRICO 21
2.1 FORMATO WIEGAND 21
2.1.1 Temporización 22
2.1.2 Formato estándar de tarjetas de 26-bits 22
2.2 PROTOCOLO DE RED 24
2.2.1 Modelo de Capas 24
2.3 PROTOCOLO TCP/IP 26
2.3.1 IP 26
2.3.2 TCP 26
2.4 ETHERNET 30
2.5 RCM3700 31
2.5.1 Rabbit 3000 34
2.5.2 Descripción de Pines del Módulo RCM 3700 35
2.5.3 Reloj de Tiempo Real (RTC, Real-Time Clock) 38
2.6 MEMORIA FLASH SERIAL 42
2.7 PANTALLA DE CRISTAL LÍQUIDO 43
2.7.1 Características principales 44
2.7.2 La memoria del LCD 44
2.7.3 Asignación de pines 45
2.7.4 Interfaz del LCD con el medio exterior 46

2.7.5 Operaciones 49
2.7.6 Comandos del LCD 51
2.8 PROGRAMACIÓN ORIENTADA A OBJETOS 52
2.8.1 Diferencias entre la POO y la programación tradicional. 53
2.8.2 Ventajas y desventajas de los lenguajes orientados a objetos 54
2.8.3 Conceptos de la Programación orientada a Objetos 54
2.9 TECNOLOGÍA .NET 56
2.9.1 POO, base de Microsoft .NET Framework 2.0 56
2.9.2 Elementos de Microsoft .NET Framework 57
2.10 MANEJO DE ARCHIVOS 59
2.10.1 Conceptos de Entrada- Salida por archivos en .NET Framework 60
2.10.2 Definición de un Stream 60
2.10.3 Acceso a Archivos y Atributos 61
2.11 BASE DE DATOS 63
2.11.1 Tipos de bases de datos 63
2.11.2 Modelos de bases de datos 64

3 CAPÍTULO 3 DISEÑO DE HARDWARE 67
3.1 DIAGRAMA DE BLOQUES DEL SISTEMA 67
3.2 DIAGRAMAS Y ESPECIFICACIONES DEL MÓDULO DE CONTROL 69
3.3 DIAGRAMAS Y ESPECIFICACIONES DEL MÓDULO DE SALIDA 73

4 CAPÍTULO 4 DESARROLLO DE SOFTWARE 78
4.1 DESARROLLO DEL SOFTWARE PARA EL CONTROLADOR 78
4.1.1 Inicialización del sistema 83
4.1.2 Inicialización LCD 86
4.1.3 Mensaje LCD 87
4.1.4 Hora 87
4.1.5 Lectura de tarjetas 88
4.1.6 Manejo de TCP 89
4.1.7 Pila TCP 93

4.1.8 Salidas 93
4.2 DISTRIBUCIÓN DE LA MEMORIA DEL SISTEMA 93
4.2.1 Base de datos de usuarios 94
4.2.2 Variables críticas de funcionamiento del sistema 95
4.2.3 Base de datos de acceso al sistema 96
4.3 DESARROLLO DEL SOFTWARE DE LA INTERFAZ DE USUARIO 96
4.3.1 Conexión tcp. 97
4.3.2 Archivos. 100
4.3.3 Usuarios y Claves. 106
4.3.4 Ambiente Gráfico 107

5 CAPÍTULO 5 IMPLEMENTACIÓN 113
5.1 HARDWARE 113
5.1.1 Implementación del Módulo de Control. 114
5.1.2 Implementación de Módulo de Salida. 116
5.2 SOFTWARE 120
5.2.1 Requerimientos de hardware de computador 120
5.2.2 Requerimientos de software de computador. 121
5.2.3 Publicación. 121

6 CAPÍTULO 6 PRUEBAS Y RESULTADOS 122
6.1 PRUEBAS Y RESULTADOS ELÉCTRICOS 122
6.1.1 Módulo de Control 122
6.1.2 Módulo de Salida. 123
6.1.3 Sistema Completo 124
6.2 PRUEBAS Y RESULTADOS DE SOFTWARE 125
6.2.1 Condiciones de prueba 127

7 CAPÍTULO 7 ANÁLISIS TÉCNICO COMPARATIVO 128
7.1 ANÁLISIS TÉCNICO COMPARATIVO CON LA PRIMERA VERSIÓN DEL SISTEMA 128

7.1.1 Control de acceso. 128
7.1.2 Registros de acceso al sistema. 129
7.1.3 Comunicación y mantenimiento. 130

ÍNDICE DE TABLAS

Tabla. 2.1. Capas del Modelo OSI 24
Tabla. 2.2. Modelo de capas TCP/IP..... 25
Tabla. 2.3. Características estándar de una red Ethernet..... 30
Tabla. 2.4. Descripción de pines RCM 3700 36
Tabla. 2.5. Registros de Datos (RTCxR) del Reloj de Tiempo Real..... 40
Tabla. 2.6. Registro de Control (RTCCR dir = 0x01) del Reloj de Tiempo Real..... 41
Tabla. 2.7. Características de la memoria flash serial del Controlador 43
Tabla. 2.8. Asignación de pines del LCD. 45
Tabla. 2.9. Tipos de registros del LCD 48
Tabla. 2.10. Resumen de Comandos del LCD..... 51

Tabla. 4.1. Inicialización de interfases para módulos de salida..... 83
Tabla. 4.2. Inicialización de interfases para LCD. 84
Tabla. 4.3. Configuración de interfases para lectura Wiegand. 85
Tabla. 4.4. Protocolo de comunicación PC – Módulo de Control. 89
Tabla. 4.5. Distribución de memoria para la Base de Datos de Usuarios 95
Tabla. 4.6. Distribución de memoria para las Variables Críticas de Funcionamiento del Sistema..... 95
Tabla. 4.7. Distribución de memoria para la Base de Datos de Acceso al Sistema .. 96
Tabla. 4.8. Estructura de archivo de usuarios 102
Tabla. 4.9. Estructura de registros de ingreso..... 103

Tabla. 5.1. Lista de materiales del módulo de control 116
Tabla. 5.2. Lista de materiales del módulo de salida..... 119
Tabla. 5.3. Usuarios y claves de acceso 121

Tabla. 6.1. Pruebas eléctricas y resultados del módulo de control..... 123
Tabla. 6.2. Pruebas y resultados eléctricos del modulo de salida..... 124
Tabla. 6.3. Pruebas eléctricas y resultados del sistema..... 124
Tabla. 6.4. Pruebas y resultados de software 126

Tabla. 7.1. Comparación de control de acceso 129
Tabla. 7.2. Registros de acceso al sistema..... 130
Tabla. 7.3. Comunicación y mantenimiento 130

ÍNDICE DE FIGURAS

Figura. 2.1. Temporización 23
Figura. 2.2. Formato de 26 bits. 23
Figura. 2.3. Configuración de paridad 24
Figura. 2.4. Flujo del Protocolo TCP/IP 28
Figura. 2.5. Pila protocolar TCP/IP 29
Figura. 2.6. RCM 3700 31
Figura. 2.7. Diagrama de bloques RCM 3700 33
Figura. 2.8. Descripción de pines RCM 3700 (vista inferior) 35
Figura. 2.9. Puertos de E/S RCM 3700 35
Figura. 2.10. Conexión del LCD utilizando un bus de 8 bits y de 4 bits 47
Figura. 2.11. Control de contraste en el LCD 49
Figura. 2.12. Códigos a enviar para inicializar el LCD, tanto a 8 bits como a 4 50
Figura. 2.13. Stream 60

Figura. 3.1. Diagrama de bloques del sistema 68
Figura. 3.2. Diagrama de bloques del módulo de control 70
Figura. 3.3. Diagrama esquemático del controlador principal RCM 3700 71
Figura. 3.4. Diagrama esquemático de los lectores, interfases de salida y LCD 73
Figura. 3.5. Diagrama de bloques del módulo de salida 74
Figura. 3.6. Diagrama esquemático del módulo de salida 76
Figura. 3.7. Diagrama esquemático del arreglo Darlington ULN2803 77

Figura. 4.1. Diagrama de bloques del software para el controlador 80
Figura. 4.2. Diagrama de bloques del software para el controlador 81
Figura. 4.3. Diagrama de bloques del software para el controlador 82

Figura. 4.4. Distribución de la memoria flash serial 94
Figura. 4.5. Tramas de Petición y Respuesta..... 92
Figura. 4.6. Diagrama de flujo, Conexión TCP/IP..... 98
Figura. 4.7. Propiedades del proyecto, espacio settings 107
Figura. 4.8. Diagrama Función Conectar 108
Figura. 4.9. Diagrama de flujo agregar usuario 109

Figura. 5.1. Diagrama de conexión módulo principal 114
Figura. 5.2. Capa superior del módulo de control..... 115
Figura. 5.3. Capa inferior del módulo de control..... 115
Figura. 5.4. Posición de los elementos en el módulo de control..... 116
Figura. 5.5. Diagrama de conexión módulo de expansión 117
Figura. 5.6. Capa superior del módulo de salida..... 118
Figura. 5.7. Capa inferior del módulo de salida..... 118
Figura. 5.8. Posición de los elementos en el módulo de salida. 119

GLOSARIO

OSI	Organización de Internacional para la Estandarización.
CSMA/CD	Detección de portadora y colisiones por múltiple acceso (Carrier Sense Múltiple Access Colision Detection).
RAM	Memoria de acceso aleatorio (Random Acces Memory).
RTC	Reloj de tiempo real (Real Time-Clock).
RCM	Módulo Rabbit (Rabbit Core Module).
LCD	Pantalla de cristal líquido (Liquid Crystal Display).
TTL	Lógica transistor – transistor (Transistor- Transistor Logia)
ROM	Memoria de solo lectura (Rad Only Memory)
EEPROM	ROM eléctricamente borrable y programable (Electrical Eraseable Programable ROM)
POO	Programación orientada a objetos.
SDK	Conjunto de herramientas para desarrollo de software (Software Development Kit).
SQL	Lenguaje estructurado de consultas (Structured Query Language)

CAPÍTULO 1

INTRODUCCIÓN

COHECO Cía. Ltda., ha logrado cubrir la mayor parte del mercado relacionado a la instalación de ascensores, ocupando aproximadamente el 60% del mercado en la provincia de Pichincha y el 80% en la provincia del Guayas.

El éxito notable que tiene COHECO Cía. Ltda. frente a la competencia radica en el hecho que la empresa proporciona mantenimiento total y de por vida a todos los equipos instalados.

En la actualidad han tomado mucho empuje los sistemas de seguridad, siendo una de sus aplicaciones más utilizadas el Control de Acceso. El sistema a diseñar e implementar constituye una de estas aplicaciones.

Desde hace 4 años en COHECO se ha utilizado una primera versión de este sistema, pero se han detectado ciertas deficiencias en su desempeño, las cuales limitan o dificultan la instalación del sistema cuando las demandas del cliente son exigentes. Las deficiencias identificadas son:

- El sistema está diseñado para manejar 1000 usuarios, pero tiene un retardo de 7 segundos de acceso para el último usuario almacenado, lo cual, para este

sistema es ineficiente. Apenas puede manejar 400 usuarios con un retardo aceptable (3 segundos) para el último usuario.

- Necesita las tarjetas magnéticas físicas para poder ser almacenadas en el controlador, lo cual constituye un problema técnico cuando se tiene que almacenar grupos grandes de usuarios, o cuando las tarjetas desaparecen ya sea por daño físico o pérdida de las mismas, impidiendo la liberación de memoria innecesaria, además de originar una falla de seguridad en el sistema.

- Lectura falsa de código Wiegand de las tarjetas magnéticas, ya que el sistema actual no diferencia los bits de paridad de los bits de código en la trama entregada por el módulo lector.

- Es posible manejar 31 salidas fijas, es decir, puede ser activado sólo un piso a la vez (de los 31 pisos disponibles), esto es ineficiente ya que existen muchos edificios en los cuales una misma persona tiene acceso a varios pisos, requiriendo una tarjeta diferente por cada piso al que tiene acceso.

- No se puede manejar horarios preestablecidos para cada usuario. Simplemente se acepta o se rechaza al usuario, sin tomar en cuenta el día y la hora en la que éste accede.

- No se proporciona una base datos que respalde información sobre quién y cuándo ingresó al sistema, lo cual constituye una deficiencia de seguridad.

- No se dispone de un respaldo de los usuarios del sistema en un computador personal (PC), lo cual sería útil si sufre daños la memoria del sistema.

- El sistema no es modular, por lo que en caso de sufrir un daño en cualquiera de sus elementos deberá ser reemplazado todo el sistema de control, esto hace que

el mantenimiento tenga un costo más elevado, tanto en materiales, como en mano de obra.

- No proporciona información a los técnicos sobre el estado de funcionamiento del sistema, que permita realizar un mantenimiento más eficiente.

1.1 JUSTIFICACIÓN

COHECO Cía. Ltda. se encuentra en una continua búsqueda para dar servicios que cubran las necesidades de sus clientes de manera efectiva. Además, la empresa brinda su apoyo a la tecnología nacional, en lugar de importarla, lo cual da lugar a la generación de proyectos que aportan al desarrollo de la empresa y de la sociedad. Por estos motivos, y tomando en cuenta los aspectos mencionados en la introducción, el proyecto propone un nuevo sistema de seguridad con las siguientes mejoras respecto al actualmente en uso:

- Incremento en el número de usuarios que puede manejar el sistema y disminución en el tiempo de respuesta del mismo, buscando la mayor comodidad de los usuarios mediante un sistema eficiente.

- Administración de los usuarios mediante un programa para PC que permita almacenar, modificar y borrar usuarios sin la necesidad de la existencia física de las tarjetas magnéticas, incrementando la seguridad del sistema.

- Control del acceso a todos los pisos de un edificio mediante el uso de salidas programables que le permitan al usuario tener acceso a cualquier combinación de pisos con una sola tarjeta magnética, minimizando los costos al usuario.

- Personalización del horario de acceso a cada usuario, individualmente, además de registro del usuario que ingresó, y en qué momento lo hizo, mejorando la característica de seguridad de éste.

- Provisión de un sistema de archivos que permita respaldar la base de datos de usuarios que posee el controlador, proveyendo al sistema de un nivel más alto de desempeño referente al manejo y protección de la información.

- Provisión de un sistema modular y expandible de manera que el módulo de control se encuentra separado de los módulos de salidas, reduciendo los costos y facilitando los procesos de instalación y mantenimiento.

- Provisión de mensajes visuales a los técnicos de mantenimiento, con el fin de asistirlos para optimizar su desempeño.

- Incremento de la capacidad del sistema al hacer uso de dos lectores de tarjetas y sus respectivas interfases de salida.

1.2 ALCANCE DEL PROYECTO

La realización de este proyecto busca solucionar las deficiencias existentes en el sistema de control de acceso que comercializa actualmente COHECO Cía. Ltda. Para ello se diseña e implementa un nuevo sistema en el que se manejan dos interfases de control de acceso simultáneas e independientes, que comparten una capacidad de memoria para 4 000 usuarios y una base de datos de 40 000 registros de acceso, cada una con salidas independientes, modulares y expandibles hasta 32 puntos. Se incluye la capacidad de conectar el sistema a un PC de manera local o remota mediante el protocolo TCP/IP para modificar y respaldar las variables de funcionamiento del sistema.

1.3 CARACTERÍSTICAS PROPUESTAS PARA EL SISTEMA

El sistema de seguridad magnético personal esta enfocado a realizar un control de acceso para ascensores Mitsubishi con las siguientes características:

- Sistema de control de acceso doble (dos lectores y dos interfases de salida, los cuales funcionan de manera independiente y simultánea), utilizando como controlador principal el sistema embebido RabbitCore RCM3700.

- Interfases de acceso mediante tarjetas magnéticas y sus respectivos lectores bajo el protocolo Wiegand de 26-bits.

- Manejo de horarios y combinaciones de pisos individuales para cada tarjeta registrada en el sistema.

- Interfases modulares que permitan la expansión del sistema hasta un máximo de 32 salidas, con 4 módulos expandibles individuales para cada interfase.

- Conectividad entre el controlador principal y el computador mediante el protocolo TCP/IP.

- Administración de una base de datos para la información de 4000 usuarios, y 40 000 registros de acceso al sistema.

- Tiempo de respuesta del sistema a un valor inferior a los cuatro segundos para el último usuario almacenado.

- Mensajes de información de alerta y funcionamiento del sistema mostrados en un LCD.

- Software para PC que permita realizar todas las operaciones de inicialización, edición y administración de bases de datos relacionadas con usuarios y registros de acceso.

CAPÍTULO 2

MARCO TEÓRICO

2.1 FORMATO WIEGAND

El término Wiegand se aplica de manera muy común a las características relacionadas con sistemas de control de acceso, lectores e incluso tarjetas. Sin embargo, esta palabra se usa descuidadamente y puede llevar a la confusión innecesaria.

Así se puede decir que Wiegand es:

- Una interface lector-tarjeta específica.
- Una interface binaria específica lector-controlador
- Una señal electrónica portadora de datos.
- Un formato binario de 26-bits para tarjetas.
- Un efecto electromagnético.
- Una tecnología para tarjetas.

Generalmente al referirse a “formato Wiegand”, se apunta al concepto de codificación de datos de tarjeta de seguridad. Aunque se necesita estar consciente de que el término, formato Wiegand, también se refiere a menudo al formato estándar de 26-bits el cual es un arreglo específico de datos binarios. Por esto se debe tener en cuenta lo siguiente:

- Un formato describe lo que un numero significa, o como un número es usado, el formato no es el numero por si mismo.

- El número de bits no indica el formato, excepto por el estándar de 26-bits.

- Para una longitud de bits dada (34-bits, 37-bits, etc.), el tamaño y posición de cada elemento puede variar.

- Cada sistema de control de acceso indica cuales formatos es capaz de reconocer y cuales no.

2.1.1 Temporización

El protocolo de transmisión Wiegand es asíncrono, posee dos líneas de transmisión que trabajan con niveles de voltaje TTL, la línea DATA-0 transporta los ceros del código y la línea DATA-1 realiza la misma función para los unos transmitidos.

Al bajar durante 50 microsegundos las líneas DATA-0 y DATA-1 se llega a indicar un 0 lógico o un 1 lógico respectivamente, como se indica en la **Figura. 2.1.**

2.1.2 Formato estándar de tarjetas de 26-bits

El formato en que una tarjeta esta programada esta determinado por el patrón compatible con el sistema de control de acceso. El formato estándar de 26-bits es un formato abierto, lo que implica que su descripción específica es de dominio público. Es ampliamente utilizado en la industria ya que prácticamente todos los sistemas de control de acceso lo aceptan.

El formato 26-bits se muestra en la **Figura. 2.2,** este de compone de 24 bits de datos, mas un bit inicial de paridad par que se calcula sobre los 12 primeros bits de

datos y por un bit final de paridad impar que se calcula en base a los últimos 12 bits de datos.

Los primeros 8 bits de datos, forman un número de 0-255 que se denomina código de factibilidad (Facility Code (FC)), el cual habitualmente identifica al lote de tarjetas. Y los últimos 16-bits de datos forman un número de 0-65535 que representa el código de la tarjeta.

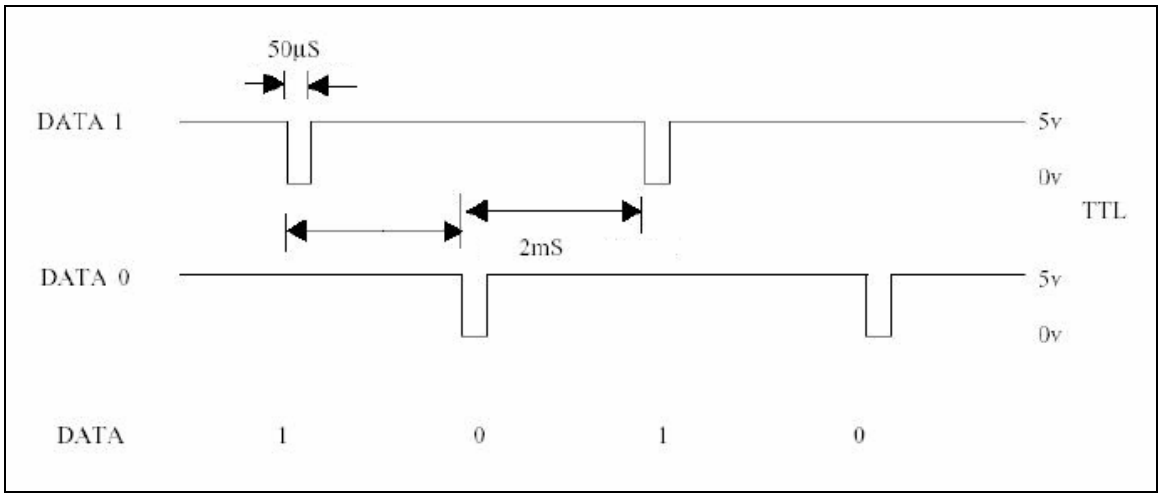


Figura. 2.1. Temporización

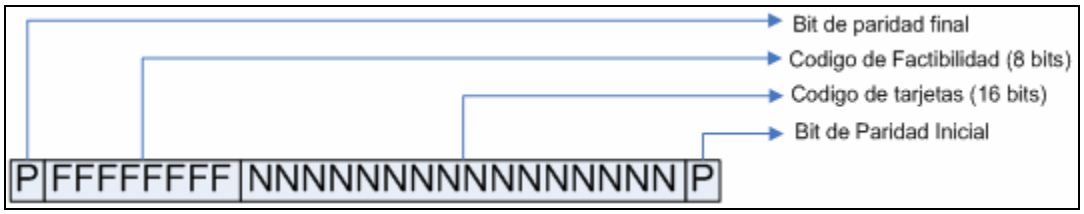


Figura. 2.2. Formato de tarjetas de 26 bits.

Los bits de paridad son utilizados como método de verificación de los datos binarios trasmitidos. La **Figura. 2.3** muestra la configuración de los bits de paridad dentro del formato de tarjetas de 26 bits.

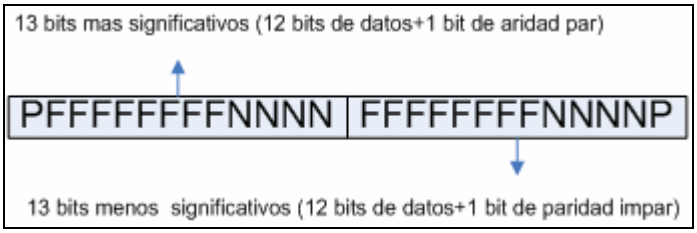


Figura. 2.3. Configuración de paridad

2.2 PROTOCOLO DE RED

Los computadores en una red se comunican de acuerdo a diferentes medios denominados protocolos. La complejidad del software para el protocolo de red viene dado por el hecho de ser dividido en muchas pequeñas piezas, con el fin de simplificar la conexión. Un modelo de capas ayuda a esta división y provee las bases conceptuales para comprender como el software y el hardware de una red conforman un poderoso sistema de comunicación.

2.2.1 Modelo de Capas

En los primeros días del trabajo con redes, la Organización de Internacional para la Estandarización (ISO) desarrolló un modelo de capas cuya terminología persiste hasta estos días.

Tabla. 2.1. Capas del Modelo OSI

	CAPA	PROPÓSITO DE LA CAPA
7	Aplicación	Indica como una aplicación particular hace uso de la red
6	Presentación	Especifica como se representan los datos
5	Sesión	Especifica como estabilizar una comunicación con el horario
4	Transporte	Especifica cómo ocuparse de transporte de datos

		fiablemente.
3	Red	Especifica las asignación de direcciones y como se remiten los paquetes.
2	Enlace	Especifica la organización de los datos en frames y como los frames se envían a la red.
1	Física	Especifica el hardware que se usa en la red

Este modelo de 7 capas, presentado en la **Tabla. 2.1** ha sido revisado y modificado a uno de solo 5 referido al modelo TCP/IP para acoplarse de mejor manera a las necesidades de los diseñadores de protocolos, el cual se muestra en la **Tabla. 2.2.**

Tabla. 2.2. Modelo de capas TCP/IP

	CAPA	**PROPÓSITO DE LA CAPA**
5	Aplicación	Indica como una aplicación particular hace uso de la red
4	Transporte	Especifica cómo ocuparse de transporte de datos fiablemente.
3	Internet	Especifica el formato del paquete y la ruta.
2	Red	Especifica la organización de los datos en frames y como los frames se envían a la red.
1	Física	Especifica el hardware que se usa en la red

2.3 PROTOCOLO TCP/IP

TCP/IP es una combinación de dos protocolos individuales: TCP e IP.

2.3.1 IP

IP es un protocolo de Capa 3 (modelo OSI), un servicio no orientado a conexión que brinda entrega de máximo esfuerzo a través de una red. Provee comunicación entre 2 terminales o hosts en diferentes tipos de redes. El hecho de no ser orientada a la conexión, implica que no existe un handshaking o acuerdo en la comunicación, cada paquete es independiente de los otros. Esto no es deseable ya que no existe garantía de que los paquetes hayan sido entregados, por lo cual los protocolos de niveles superiores tienen que lidiar con esto.

2.3.1.1 Dirección IP

Define un esquema de direccionamiento mediante un único número de 32 bits para cada terminal en una red. Este número se conoce como el la Dirección de Protocolo de Internet (Internet Protocol Address). Cada paquete enviado a internet contiene la dirección IP de destino y la de origen.

2.3.2 TCP

TCP es un protocolo de Capa 4 (modelo OSI): un servicio orientado a conexión que suministra control de flujo y confiabilidad. Ofrece un circuito virtual entre aplicaciones de usuario final. Sus características son las siguientes:

- Orientado a conexión
- Confiable
- Divide los mensajes salientes en segmentos
- Reensambla los mensajes en la estación destino
- Vuelve a enviar lo que no se ha recibido
- Reensambla los mensajes a partir de segmentos entrantes

2.3.2.1 TCP Socket

Una conexión TCP se realiza mediante un acuerdo de 3 vías entre el cliente y el servidor. El proceso se realiza de la siguiente manera:

- El cliente pide una conexión enviándole segmento TCP con el bit de control SYN activado.

- El servidor responde con su propio segmento SYN que incluye la información de identificación enviada por el cliente en el segmento SYN inicial.

- El cliente reconoce el segmento SYN del servidor.

Así la conexión es estabilizada y se identifica únicamente por el cuarteto denominado socket (dirección IP de destino, puerto de destino, dirección IP de origen, puerto de origen). Mientras dura la fase de inicialización de la conexión, estos valores son ingresados en una tabla y se mantienen almacenados durante toda la conexión.

La reunión de ambos protocolos les permite ofrecer una gama de servicios más amplia. Juntos, representan la totalidad del conjunto. TCP/IP es el protocolo de Capa 3 y Capa 4 en el que se basa Internet.

En la **Figura. 2.4** se muestra el flujo de información bajo el protocolo TCP/IP.

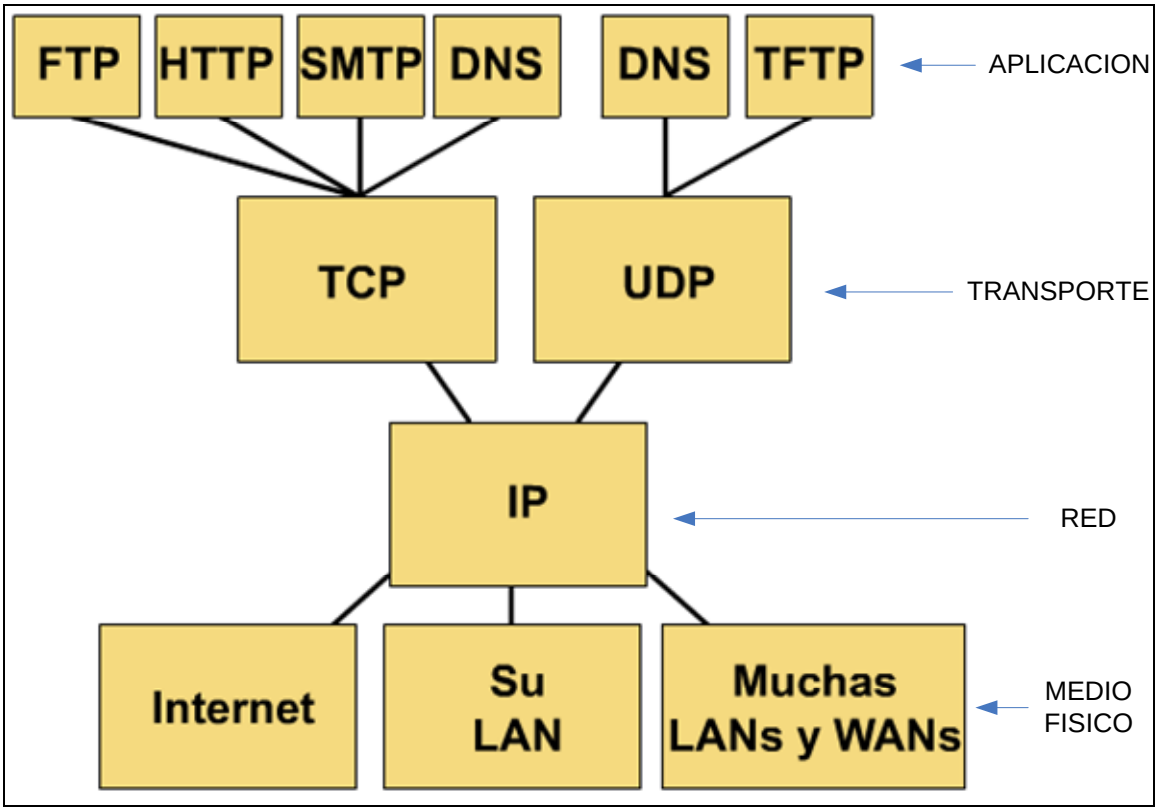


Figura. 2.4. Flujo del Protocolo TCP/IP

2.3.2.2 Pila Protocolar TCP/IP

TCP/IP es la colección de protocolos en que toda la comunicación de Internet esta basada. Diferentes distribuidores han desarrollado otros protocolos para gestión de redes, pero incluso la mayoría de sistemas operativos con sus propios protocolos desarrollados, soportan TCP/IP.

Los protocolos en algunas ocasiones se denominan pila protocolar. Una pila protocolar es un término apropiado que porque indica que el enfoque de capas usado para el diseño del software de gestión de redes como se presenta en la Figura. 2.5.

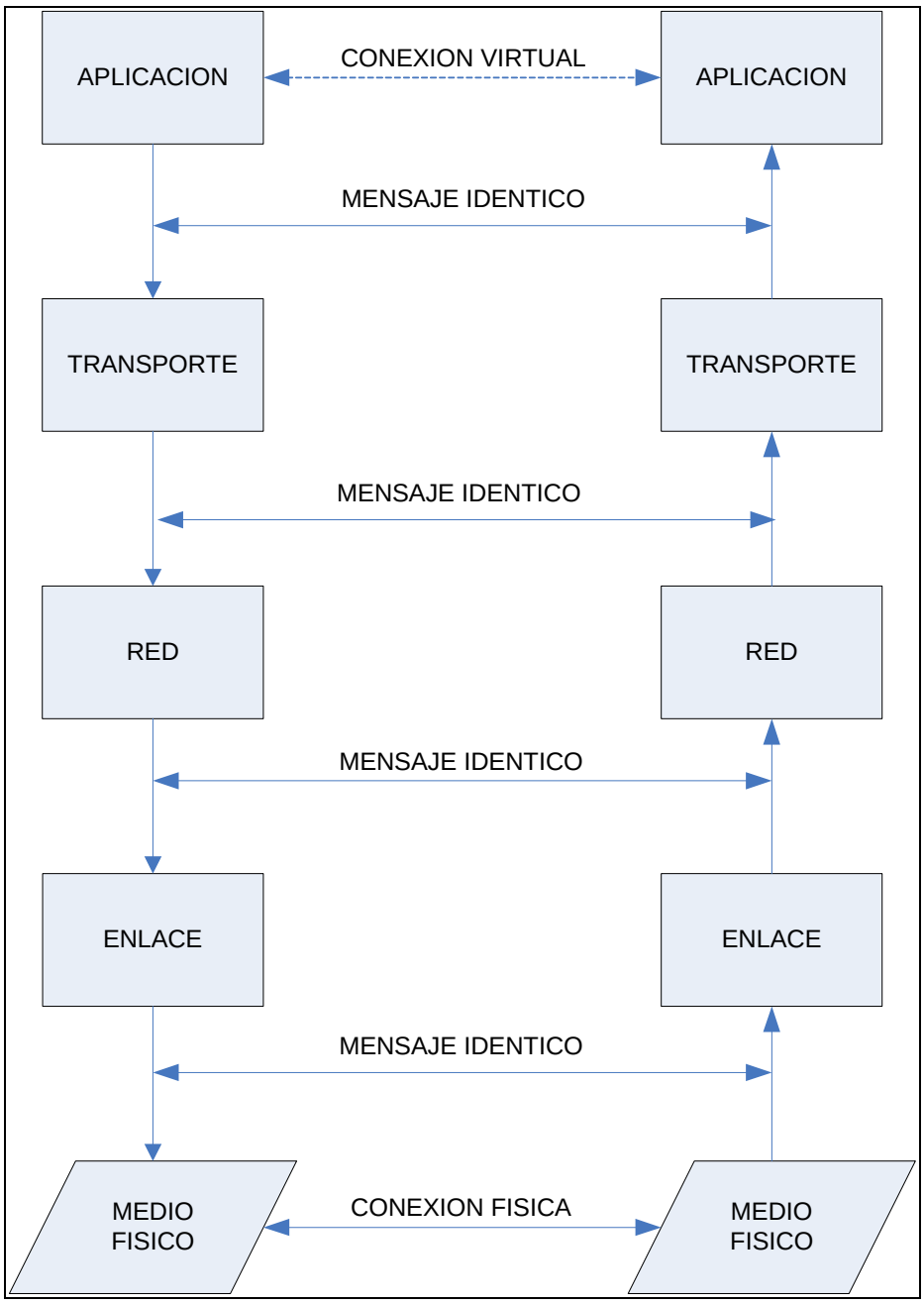


Figura. 2.5. Pila protocolar TCP/IP

Cada host o router en Internet debe ejecutar una pila protocolar. Los detalles de las conexiones físicas subyacentes están ocultos por el software. El software de transmisión en cada capa se comunica con la correspondiente capa al lado de la recepción mediante información almacenada en las cabeceras de los paquetes. Cada capa añade su cabecera en al inicio del mensaje para la capa superior

siguiente. Así mismo en el lado receptor las cabeceras son removidas del mensaje por las correspondientes capas.

2.4 ETHERNET

TCP/IP es un conjunto de protocolos independiente del medio físico que se utilice para transmitir la información, sin embargo el medio más utilizado para este fin es Ethernet.

Ethernet es a la vez el nombre de una técnica de comunicación y de una red comercial diseñada por Xerox Corporation en los años setenta. Las especificaciones de la red fueron adaptadas y completadas por la propia Xerox, Intel entre otras. Dichas especificaciones abarcan desde medio físico (nivel OSI 1) hasta ciertas técnicas a emplear en el nivel de enlace (OSI 2). Las principales características se presentan en la **Tabla. 2.3.**

Tabla. 2.3. Características estándar de una red Ethernet

Medio Físico Estándar	Cable Ethernet CAT 5/6
Topología de la Red	Estrella
Modo de Transmisión	Full-duplex, banda base
Codificación	Manchester
Velocidad Estándar	10Mbits/s
Método de Acceso	CSMA/CD (Carrier Sense Múltiple Access Colision Detection)

CSMA/CD es una técnica apta básicamente para topologías en bus y está descrita en la recomendación IEEE-.802.3, funciona de la siguiente forma:

Cuando una estación desea transmitir verifica si el medio físico está ocupado, a base de detectar si existe señal de datos. Si el medio físico está libre la estación toma el control del mismo y lo mantiene hasta concluir la transmisión. Si se detecta ocupación la estación espera y va haciendo varios intentos hasta que el medio está libre. Si dos o más estaciones intentan ocupar el medio simultáneamente, se detecta una colisión que obliga a cada estación a esperar un tiempo aleatorio antes de volver a reintentar transmitir. Este tiempo suele depender del número de veces sucesivas que se ha intentado el acceso y se suele doblar en cada intento con un límite máximo.

La técnica **CSMA/CD** permite que una estación sea retirada o colocada sin exigir el paro de la red pero, en cambio no permite garantizar la transmisión del mensaje en un tiempo determinado, ya que éste dependerá del nivel de ocupación de la red.

2.5 RCM3700

Figura. 2.6. RCM 3700

El sistema embebido que se muestra en la **Figura. 2.6**, se encuentra orientado a la conectividad, incorpora las capacidades del microprocesador Rabbit 3000, con una memoria flash para código, una memoria flash serial para datos y una memoria RAM estática, además de varios puertos de entrada salida y de comunicaciones.

Las características básicas de este dispositivo se muestran a continuación:

- Microprocesador: Rabbit 3000 operando a 22.1 MHz
- 33 E/S paralelas TTL: 31 configurables para E/S, 2 salidas fijas.
- Reset externo de E/S.
- Bus de entrada salida alterno que puede ser configurado como 8 líneas de datos y 5 líneas de direcciones.
- 10 timers de 8-bits (6 se pueden conectar en cascada) y uno de 10-bits.
- 512K de memoria y 512K de SRAM
- 1 Mbyte de memoria flash serial.
- Reloj de tiempo real.
- Watchdog.
- Permite la conexión de una batería de back-up (3V) provista por el usuario.
- Modulador PWM de 10 bits
- 2 canales decodificadores en cuadratura.
- 4 puertos seriales compatibles con CMOS (3.3 V).
- Puerto de conexión a Ethernet 10baseT.

En la **Figura. 2.7** se presenta un diagrama de bloques de los subsistemas que se encuentran dentro del módulo RCM3700.

El procesador Rabbit 3000 controla todas las operaciones del sistema, el código del programa se encuentra almacenado en la memoria flash de programa, los datos e información que se reutilizan dentro de un programa son almacenados en la memorias RAM estática cuya información es respaldada por una batería provista por

el usuario, además de esto existe una memoria flash serial que provee al sistema de una gran capacidad de almacenamiento adicional.

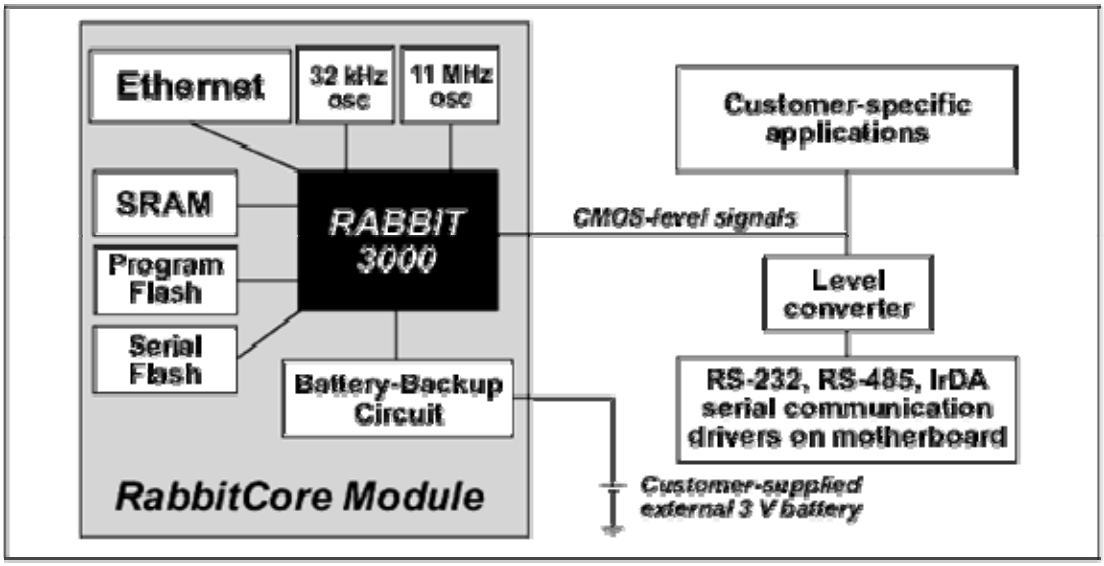


Figura. 2.7. Diagrama de bloques RCM 3700

El modulo RCM 3700 posee dos osciladores, el primero de 11 MHz permite operar al procesador Rabbit 3000, su señal puede ser dividida o incluso duplicado dependiendo de los requerimientos del usuario. El oscilador de 32KHz permite operar al reloj de tiempo real el cual es alimentado por la batería de respaldo, en el caso de pérdida de energía en el módulo.

En lo referente a las comunicaciones el sistema posee varios puertos seriales que le permiten manejar protocolos de comunicación RS232, RS232C, RS485, SPI, IrDA, dependiendo de los drivers a los que se encuentren conectados, finalmente posee un puerto ethernet 10 BaseT controlado por el chip RTL8019AS; sin embargo hay que acotar que por las limitaciones del sistema la mayor velocidad de conexión que se alcanza es de 64Kbps.

2.5.1 Rabbit 3000

Es un procesador de 8 bits, con un bus de datos externo de 8 bits y uno interno de la misma longitud, con un set compacto de instrucciones del tamaño de un byte.

Desciende de la familia de procesadores Z-180, por lo cual su set de instrucciones es muy similar al de dicha familia, sin embargo se han introducido mejoras que han hecho del procesador Rabbit 3000 un dispositivo más eficiente.

Opera en hasta velocidades de 40 MHz sin ningún tipo de emisión electromagnética, debido a su doblador de frecuencia interno y a su reloj con spectrum spreader, el cual modula la señal de reloj evitando que se produzcan emisiones electromagnéticas intensas que afecten a otros dispositivos que formen parte del sistema en donde se encuentra el procesador.

El procesador Rabbit 3000 ha sido diseñado para manejar chips de memoria estática de manera que no se necesite de una lógica externa para esto. Además, durante la operación lenta del reloj, el ciclo de trabajo del la memoria es debidamente reducido mediante hardware interno del procesador lo que provoca una disminución en el consumo de energía de las memorias.

El bus externo del procesador utiliza dos pulsos de reloj para los ciclos de lectura y 3 para los ciclos de escritura, esto tiene varias ventajas respecto al diseño de un solo pulso de reloj, estas ventajas son: un diseño fácil que evita los conflictos entre los buses, ciclos de escritura limpios con sólidos tiempos de suspensión para datos y direcciones, flexibilidad de obtener tiempos de habilitación de salida de la memoria mayores a la mitad del ciclo del bus, y la habilidad de utilizar una señal asimétrica de reloj generado por el doblador de reloj que posee el procesador.

El procesador opera a 3.6V sin embargo sus entradas pueden tolerar hasta 5V haciéndolo completamente compatible con dispositivos con lógica TTL.

2.5.2 Descripción de Pines del Módulo RCM 3700

En la **Figura. 2.8**, **Figura. 2.9** y en la **Tabla. 2.4**, se presentan un diagrama de los pines de controlador RCM 3700, un esquema de sus puertos de entrada salida y una descripción de las funciones que cada uno tiene.

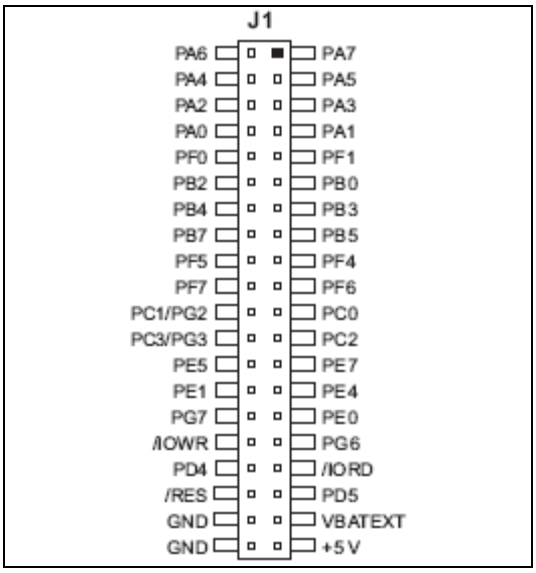


Figura. 2.8. Descripción de pines RCM 3700 (vista inferior)

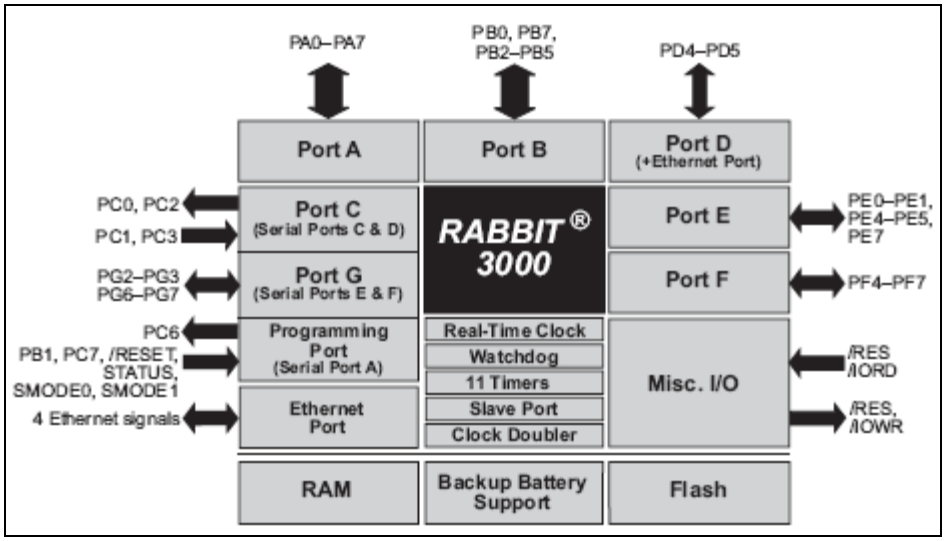


Figura. 2.9. Puertos de E/S RCM 3700

Tabla. 2.4. Descripción de pines RCM 3700

Pin	Nombre	Función Normal	Función Alterna	Notas
1-8	PA[7:0]	E/S paralelas	Bus externo de datos (ID0-ID7) Puerto esclavo del bus de datos (SD0-SD7)	Bus externo de datos
9	PF1	Entrada/Salida	QD1A CLKC	
10	PF0	Entrada/Salida	QD1B CLKD	
11	PB0	Entrada/Salida	CLKB	
12	PB2	Entrada/Salida	IA0 /SWR	Dirección externa v0 (Slave port write)
13	PB3	Entrada/Salida	IA1 /SRD	Dirección externa 1 (Slave port read)
14	PB4	Entrada/Salida	IA2 SA0	Dirección externa 2 Direction del Puerto Esclavo 0
15	PB5	Entrada/Salida	IA3 SA1	Dirección externa 3 Dirección del Puerto Esclavo 1
16	PB7	Entrada/Salida	IA5 /SLAVEATTN	External Address 5 (Slave Port Attention)

17	PF4	Entrada/Salida	AQD1B PWM0	
18	PF5	Entrada/Salida	AQD1A PWM1	
19	PF6	Entrada/Salida	AQD2B PWM2	
20	PF7	Entrada/Salida	AQD2A PWM3	
21	PC0	Salida	TXD	Puerto Serial D
22	PC1/PG2	Entrada/Salida	RXD/TXF	Puerto Serial D Puerto Serial F
23	PC2	Salida	TXC	Puerto Serial C
24	PC3/PG3	Entrada/Salida	RXC/RXF	Puerto Serial C Puerto Serial F
25	PE7	Entrada/Salida	I7 /SCS	Dirección externa 7 (Slave Port Chip Select)
26	PE5	Entrada/Salida	I5 INT1B	
27	PE4	Entrada/Salida	I4 INT0B	
28	PE1	Entrada/Salida	I1 INT1A	I/O Strobe 1 Interrupción 1A
29	PE0	Entrada/Salida	I0 INT0A	I/O Strobe 0 Interrupción 0A

30	PG7	Entrada/Salida	RXE	Puerto Serial E
31	PG6	Entrada/Salida	TXE	
32	/IOWR	Salida		Strobe para escritura interna
33	/IORD	Entrada		Strobe para lectura externa
34	PD4	Entrada/Salida	ATXB	Se alterna con el Puerto serial B
35	PD5	Entrada/Salida	ARXB	
36	/RES	Salida del reset	Entrada del reset	Salida del generador de reset
37	VBAT			Entrada de la batería de respaldo
38	GND			
39	+5 V			
40	GND			

2.5.3 Reloj de Tiempo Real (RTC, Real-Time Clock)

Un Reloj de Tiempo Real (RTC) es un reloj de computadora (la mayoría de veces en forma de un chip de circuito integrado) que almacena el valor del tiempo actual, incluso cuando el computador está apagado. Esto es utilizado en muchos tipos de computadores, y está presente en todos los computadores personales modernos. Los RTCs también están presentes en muchos sistemas embebidos (embedded systems), como el utilizado en el desarrollo de este proyecto: el RCM3700 RabbitCore.

Los relojes de tiempo real operan con una batería especial que no está conectada a la fuente de alimentación normal. En contraste, los relojes que no son de tiempo real no funcionan cuando el computador está apagado.

En el caso del RCM3700 RabbitCore, el reloj de fecha/hora (RTC) es un contador (tipo rizo: ripple) de 48[bits] que es manejado por un oscilador de 32.768[KHz]. El RTC es un contador tipo rizo modificado, compuesto por seis contadores separados de 8[bits]. Los acarreos son alimentados en todos los seis contadores de 8[bits] al mismo tiempo y entonces se tiene el rizo para 8[bits]. El tiempo para que tenga lugar este rizo es de unos pocos nanosegundos por bit, y ciertamente no debe exceder 200[ns] para todos los 8[bits], incluso al operar a bajo voltaje.

Los 48[bits] son suficientes para contar hasta 272[años] a la frecuencia de reloj de 32[KHz]. Por convención, se toma como tiempo cero las 12 AM del 1 de enero de 1980. El software de Z-World ignora el bit de mayor orden, dando al contador una capacidad de 136[años] desde el 1 de enero de 1980. Para leer el valor del contador, el valor es primero transferido a un registro de retención de 6[Bytes]. Entonces los bytes individuales pueden ser leídos de los registros de retención. Para realizar la transferencia, cualquier bit de datos se escribe en RTC0R, el primer registro de retención. El contador puede entonces ser leído como seis bytes de RTC0R hasta RTC5R. El contador y el oscilador de 32[KHz] son energizados por un pin de energía separado que puede estar suministrado con potencia mientras el resto del chip está apagado. Este diseño hace posible el apoyo de una batería. Puesto que el procesador opera en un reloj diferente que el RTC, existe la posibilidad de desarrollar una transferencia a los registro de retención mientras está tomando lugar un acarreo, resultando en una información incorrecta. Para prevenir esto, el procesador debe leer el reloj dos veces y asegurar que el valor es el mismo en ambas lecturas.

Si el procesador está operando a 32[KHz], el procedimiento de lectura del reloj debe ser modificado puesto que varias cuentas de reloj tendrían lugar en el tiempo

necesario por el cronometraje lento del procesador para leer el reloj. Una modificación apropiada sería ignorar los bytes más bajos y sólo leer los 5[bytes] más altos, que se cuenta una vez cada 256[pulsos de reloj] o cada 1/128[segundos]. Si la lectura no fue desarrollada en este periodo, pueden ignorarse los bits de menor orden.

Los registros del RTC no pueden fijarse por una operación de escritura, pero pueden ser borrados y contar individualmente, o por subconjuntos. En esta forma, cualquier registro o el contador entero de 48[bits] pueden fijarse en cualquier valor con no más de 256 pasos. Si el cristal de 32[KHz] no está instalado y el pin de entrada está aterrizado, no tendrá lugar el conteo y los seis registros pueden ser utilizados como una pequeña memoria respaldada por batería. Normalmente esto no sería muy productivo puesto que la circuitería necesaria para proporcionar el interruptor de energización podría también ser utilizado para respaldar con batería una RAM estática de baja potencia.

La **Tabla. 2.5** y **Tabla. 2.6** se detallan el Registro de Datos y el Registro de Control utilizados para la manipulación interna del Reloj de Tiempo Real empleado por el sistema embebido que comprende el controlador principal del sistema.

Tabla. 2.5. Registros de Datos (RTCxR) del Reloj de Tiempo Real

Reloj de Tiempo Real x Registro de Retención		(RTC0R) R/W	(Dirección = 0x02)
		(RTC1R)	(Dirección = 0x03)
		(RTC2R)	(Dirección = 0x04)
		(RTC3R)	(Dirección = 0x05)
		(RTC4R)	(Dirección = 0x06)
		(RTC5R)	(Dirección = 0x07)
Bit (s)	**Valor**	**Descripción**	
7:0	Lectura	Se retorna el valor actual del registro de retención del RTC de 48[bits]	

	Escritura	Escribiendo al RTC0R transfiere la cuenta actual del RTC a los seis registros de retención mientras el RTC continua contando

Tabla. 2.6. Registro de Control (RTCCR dir = 0x01) del Reloj de Tiempo Real

Bit (s)	Valor	Descripción
7:0	0x00	Escribiendo un 0x00 al RTCCR no tiene efecto en el contador RTC. Sin embargo, dependiendo de cual fue el comando previo, escribiendo un 0x00 puede 1. Deshabilitar la función de incremento de byte o 2. Cancelar el comando de reset del RTC. Si el comando 0xC0 es seguido por el comando 0x00, solo la función de incremento de byte será deshabilitada. El reset del RTC todavía tendrá lugar.
	0x40	Arma el RTC para un reset con código 0x80 o reset y función de incremento de byte con código 0x0C0
	0x80	Resetea todos los seis bytes del contador RTC a 0x00 si procedió el comando de armado 0x40
	0xC0	Resetea todos los seis bytes del contador RTC a 0x00 y entra en modo de incremento de byte. Preceda a este comando con el comando de armado 0x40
7:6	01	Esta combinación de bits debe ser usada con cada escritura del incremento de byte para incrementar los registros del reloj (es) correspondientes a los bits que contienen “1”. Ejemplo: 01001101 incrementa los registros 0, 2, 3. El modo de incremento de byte debe ser habilitado. Almacenando 0x00 cancela el modo de incremento de byte.
5:0	0	No causa efecto en el contador RTC
	1	Incrementa el byte correspondiente del contador RTC

2.6 MEMORIA FLASH SERIAL

La memoria flash es una forma evolucionada de la memoria EEPROM que permite que múltiples posiciones de memoria sean escritas o borradas en una misma operación de programación mediante impulsos eléctricos, frente a las anteriores que sólo permite escribir o borrar una única celda cada vez. Por ello, flash permite funcionar a velocidades muy superiores cuando los sistemas emplean lectura y escritura en diferentes puntos de esta memoria al mismo tiempo.

Las memorias flash son de tipo no volátil, esto es, la información que almacena no se pierde en cuanto se desconecta de la corriente, una característica muy valorada para ciertos dispositivos específicos en los que se necesita este tipo de memoria.

Tiene como característica ser realmente muy silenciosa, ya que no contiene ni actuadores mecánicos ni partes móviles. Sin embargo, todos los tipos de memoria flash sólo permiten un número limitado de escrituras y borrados, generalmente entre 50 000 y un millón, dependiendo de la celda, de la precisión del proceso de fabricación y del voltaje necesario para su borrado.

El chip de memoria flash utilizada en el controlador consiste de una serie de sectores. El tamaño de cada sector es el mismo dentro de un chip dado, pero puede variar de un chip a otro. Un sector también contiene bytes extra que no son disponibles para el usuario. Algunos de estos bytes son utilizados por el controlador de la siguiente manera:

- Un valor de la sincronización fijo, indicando que el sector ha sido escrito al menos una vez,
- Un número de versión, y
- Un valor de entero largo (long integer) para el número de veces que el sector ha sido escrito.

El usuario debe tener en mente que un chip de memoria flash tiene un número limitado de ciclos de escritura por sector, y que debe ser escrito un sector completo a la vez. Los chips utilizados en el Controlador son especificados típicamente en 50 000 ciclos de escritura. El uso más eficiente dicta que debe realizarse la escritura de todo el bloque de usuario a la memoria flash, para que solo los sectores llenos sean escritos.

Depende del usuario mantener la información apropiada dentro del programa, para poder guardar y recuperar los datos de la memoria flash.

La memoria flash serial utilizada por el controlador principal: RCM3700 RabbitCore es de 1MB de capacidad, la cual se encuentra distribuida bloques y páginas, como se muestra en la **Tabla. 2.7**:

Tabla. 2.7. Características de la memoria flash serial del Controlador

Tamaño de la memoria	1 Megabyte
Número de bloques	4096 bloques
Tamaño del bloque	264 Bytes
Tamaño del Prefijo	8 Bytes
Tamaño de la Página	256 Bytes

2.7 PANTALLA DE CRISTAL LÍQUIDO

La pantalla de cristal líquido o LCD (Liquid Crystal Display) es un dispositivo micro-controlado de visualización gráfica para la presentación de caracteres, símbolos o incluso dibujos (en algunos modelos), en este caso dispone de 2 filas de 16 caracteres cada una y cada carácter dispone de una matriz de 5x8 puntos (píxeles), aunque los hay de otro número de filas y caracteres. Este dispositivo esta gobernado internamente por un microcontrolador que regula todos los parámetros de presentación.

2.7.1 Características principales

Las características principales del LCD son las siguientes:

- Pantalla de caracteres ASCII, además de los caracteres Kanji y Griegos.
- Desplazamiento de los caracteres hacia la izquierda o la derecha.
- Proporciona la dirección de la posición absoluta o relativa del carácter.
- Memoria de 40 caracteres por línea de pantalla.
- Movimiento del cursor y cambio de su aspecto.
- Permite que el usuario pueda programar 8 caracteres.
- Conexión a un procesador usando un interfaz de 4 u 8 bits.
- Tensión nominal de alimentación de 5V, con un consumo menor a 5mA.

El LCD dispone de una matriz de 5x8 puntos para representar cada carácter. En total se pueden representar 256 caracteres diferentes. 240 caracteres están grabados dentro del LCD y representan las letras mayúsculas, minúsculas, signos de puntuación, números, etc. Adicionalmente, existen 8 caracteres que pueden ser definidos por el usuario.

2.7.2 La memoria del LCD

El LCD dispone de dos tipos de memorias independientes: la DD RAM y la CG RAM.

2.7.2.1 DD RAM (DISPLAY DATA RAM)

En esta memoria se almacenan los caracteres que están siendo visualizados o que se encuentran en posiciones no visibles. El display almacena en esta memoria dos líneas de 40 caracteres pero sólo se visualizan 2 líneas de 16 caracteres. Por ello la DD RAM tiene un tamaño de 2x40=80 bytes.

Debido a esta peculiar disposición de la DD RAM se puede pensar en el display como un display virtual constituido por dos líneas de 40 caracteres.

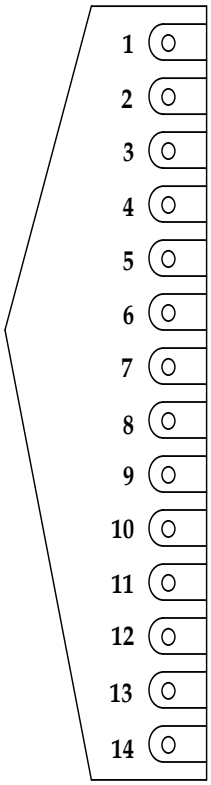
2.7.2.2 La CG RAM (CHARACTER GENERATOR RAM)

La CG RAM es la memoria que contiene los caracteres definibles por el usuario. Está formada por 64 posiciones. Cada posición es de 5 bits. La memoria está dividida en 8 bloques, correspondiendo cada bloque a un carácter definible por el usuario. Por ello el usuario puede definir como máximo 8 caracteres, cuyos códigos van del 0 al 7.

2.7.3 Asignación de pines

La Tabla. 2.8, muestra la asignación de pines para la pantalla de cristal líquido.

Tabla. 2.8. Asignación de pines del LCD.



PIN Nº	SIMB.	DESCRIPCIÓN	
1	V_{SS}	Tierra de alimentación GND	
2	V_{DD}	Alimentación de +5V	
3	V_O	Voltaje de ajuste de Contraste del LCD (0 a +5V)	
4	RS	Selección de: registro de control/registro de datos: RS=0 Selección registro de control RS=1 Selección registro de datos	
5	R/W	Señal de: lectura/escritura: R/W=0 Escritura (Read) R/W=1 Lectura (Write)	
6	E	Habilitación del modulo: E=0 Módulo desconectado E=1 Módulo conectado	
7	D0	Bit de datos LSB	BUS DE DATOS BIDIRECCIONAL
8	D1	Bit de datos	
9	D2	Bit de datos	

	10	D3	Bit de datos	
	11	D4	Bit de datos	
	12	D5	Bit de datos	
	13	D6	Bit de datos	
	14	D7	Bit de datos MSB	

2.7.4 Interfaz del LCD con el medio exterior

Los datos se transmiten por un bus de datos de 8 bits (El display ofrece la posibilidad de trabajar con este bus multiplexado en dos grupos de 4 bits). Para el control del display son necesarios 3 bits: una señal de Habilitación (E), una para indicar Lectura/Escritura (RW) y otra para seleccionar uno de los dos registros internos Seleccionar Registro (RS). Por ello, en el caso peor, el sistema de control del display necesitará utilizar 8+3=11 bits.

2.7.4.1 El bus de datos

El bus de datos del display se puede configurar para funcionar de dos formas diferentes, las cuales se presentan en la **Figura. 2.10**. Bien como un bus de 8 bits o como un bus multiplexado de 4 bits. El utilizar el bus multiplexado de 4 bits es una opción muy útil para ahorrar bits en el sistema de control. En vez de utilizar 11 bits en total, se utilizan sólo 7. De esta forma se ahorran bits pero se gana en complejidad del controlador, que tiene que multiplexar y demultiplexar los datos.

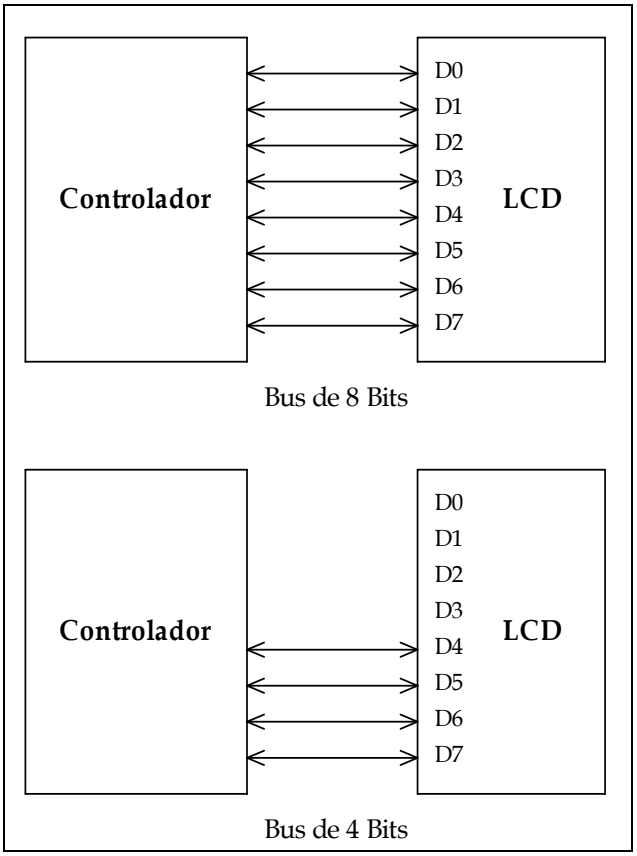


Figura. 2.10. Conexión del LCD utilizando un bus de 8 bits y de 4 bits

2.7.4.2 El bus de control

El bus de control está formado por 3 señales: RS, R/W y E. La señal E es la señal de validación de los datos. Cuando no se utiliza el display esta señal debe permanecer en 0. Sólo en las transferencias de información (lecturas o escrituras) es cuando se pone en nivel 1 para validar los datos, pasando después de un tiempo a nivel 0.

La señal R/W permite seleccionar si la operación que se va a realizar sobre el display es una lectura o una escritura. Cuando R/W=1 se realizan lecturas y cuando R/W=0 escrituras. Lo normal siempre es realizar escrituras, no obstante, el display ofrece la posibilidad de poder leer los contenidos de la memoria CG RAM y DD RAM, así como leer el estado interno del display (ocupado o disponible) y el contador de direcciones.

Con RS (Register Select) se selecciona el registro interno del display sobre el que se va a leer/escribir. El LCD dispone de dos registros internos: Registro de control y Registro de datos. Ambos registros son de lectura y escritura. RS=0 selecciona el registro de control y RS=1 el registro de datos. La Tabla. 2. 9Tabla. 2. 9 resume los tipos de registros que posee el controlador de la pantalla de cristal líquido.

Tabla. 2. 9. Tipos de registros del LCD

	REGISTRO DE CONTROL	REGISTRO DE DATOS
LECTURA	Lectura de la bandera de ocupado (D7) y del contador de direcciones (DO-D6)	Leer contenido de la memoria CG RAM o DD RAM
ESCRITURA	Ejecución de un comando interno: borrar display, desplazar el display, mover cursor, etc.	Escribir en la DD RAM o CG RAM

2.7.4.3 El control de contraste

Para controlar el contraste hay que introducir por el pin VC una tensión entre 5V y 0V. La tensión típica es de 0.6V. Normalmente se coloca un potenciómetro para poder ajustar en cada momento el contraste adecuado. En la **Figura. 2.11.** se muestra un esquema típico del control de contraste.

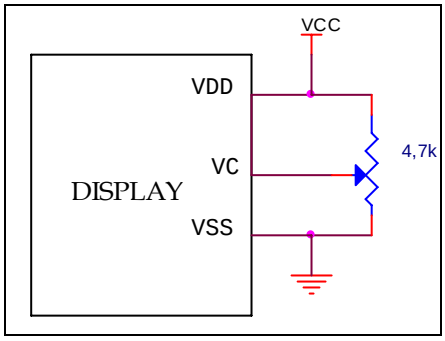


Figura. 2.11. Control de contraste en el LCD

2.7.5 Operaciones

Operaciones a realizar en el caso de 8 bits

- La señal E se encuentra siempre en 0 antes de realizar cualquier operación.
- Poner RS=1 y R/W=0
- Situar el dato a imprimir en el bus de daros del LCD (en este ejemplo se enviaría $41)
- E=1
- E=0
- El carácter ha sido impreso en el LCD

Operaciones a realizar en el caso de 4 bits

- Poner RS=1 y R/W=0
- Situar el valor $4 en el bus de daros del LCD (4 bits más significativos)
- E=1
- E=0
- Situar el valor $1 en el bus de daros del LCD (4 bits menos significativos)
- E=1
- E=0
- El carácter ha sido impreso en el LCD

2.7.5.1 Secuencia Típica de Inicialización del LCD

En la **Figura. 2.12** se ha representado en un diagrama la secuencia de inicialización del LCD para trabajar con un bus de datos de 8 o 4 bits.

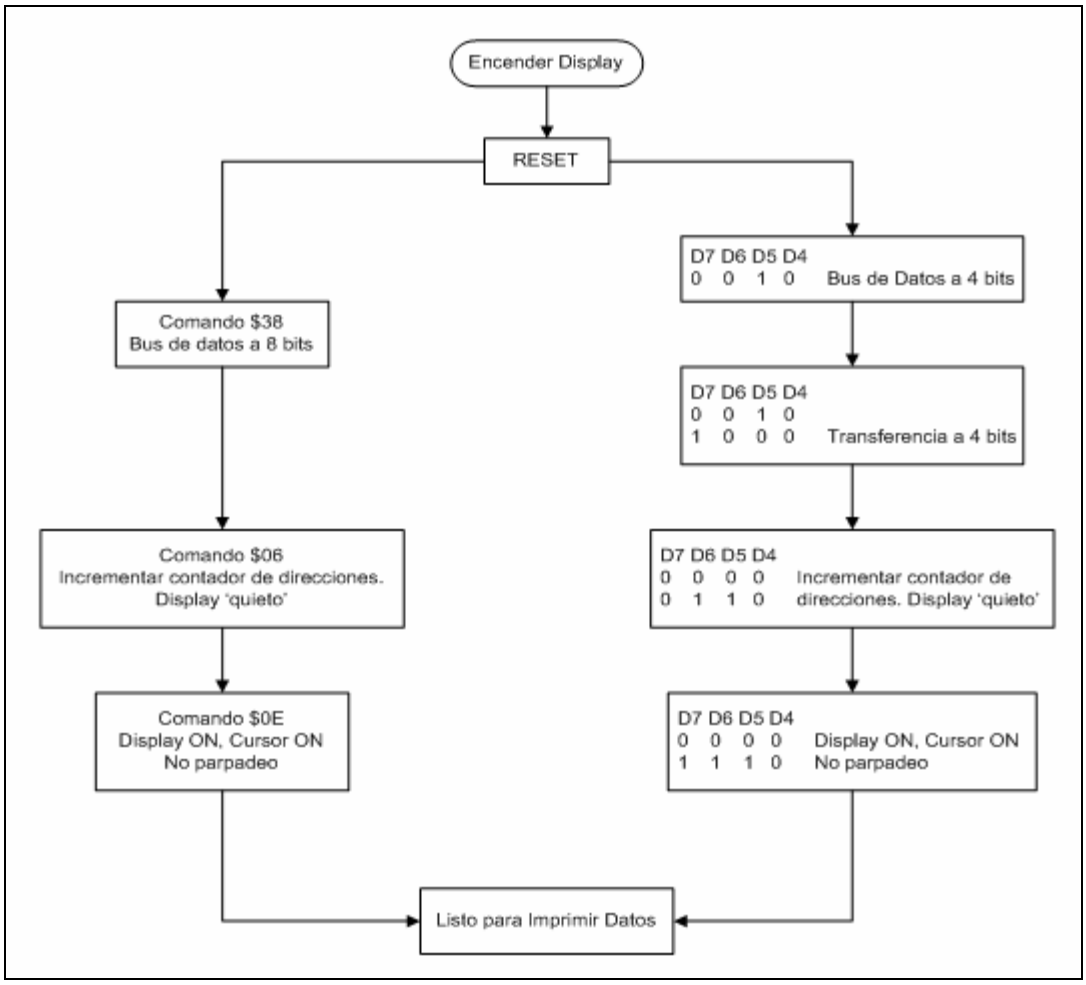


Figura. 2.12. Códigos a enviar para inicializar el LCD, tanto a 8 bits como a 4

Para el caso de 8 bits no hay ningún problema, sin embargo, el caso de 4 bits es un poco más complejo. Después de encender el LCD aparecerá la línea superior un poco más obscura que la inferior. Esto quiere decir que el display no ha sido inicializado todavía. En el caso de 4 bits solo se conectan los 4 bits más significativos del LCD, dejando los otros 4 al 'aire' al enviar el código 2 (bits 0 0 1 0) el display se

configura para trabajar a 4 bits. Se puede observar como la línea superior deja de estar más obscura que la inferior. A partir de este momento las transferencia hay que realizarlas en dos partes: primero se envían los 4 bits más significativos y después los 4 bits menos significativos. Para confirmar que la transferencia es a 4 bits hay que enviar el código $28; primero los bits 0 0 1 0 y después los bits 1 0 0 0. De aquí en adelante la inicialización el igual tanto para 8 bits como para 4, con la salvedad de que en el segundo caso hay que enviar los datos multiplexados.

2.7.6 Comandos del LCD

Los comandos que controlan a la pantalla de cristal líquido se encuentran en la **Tabla. 2.10.**

Tabla. 2.10. Resumen de Comandos del LCD

INSTRUCCIÓN	CÓDIGO										TIEMPO MÁX. DE EJECUCIÓN
	RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0	
Borrar el Display	0	0	0	0	0	0	0	0	0	1	82us – 1.64ms
Cursor a HOME	0	0	0	0	0	0	0	0	1	X	40us – 1.64ms
Establecer modo de funcionamiento	0	0	0	0	0	0	0	1	I/D	S	40us
Control ON/OFF	0	0	0	0	0	0	1	D	C	B	40us
Desplazamiento del cursor/display	0	0	0	0	0	1	S/C	R/L	X	X	40us
Modo de transferencia de la información	0	0	0	0	1	DL	1	0	X	X	40us
Acceso a la memoria CG RAM	0	0	0	1	Dirección de la CG RAM						40us
Acceso a la memoria DD RAM	0	0	1	Dirección de la DD RAM							40us

Lectura de dirección y de la bandera de ocupado	0	1	BF	Contador de Dirección	40us
Escritura de datos en CG RAM / DD RAM	1	0	Dato a Escribir		40us
Lectura de datos en CG RAM / DD RAM	1	1	Dato Leído		40us

I/D=1 : Incrementar contador de direcciones	I/D=0 : Decrementar contador de direcciones
S=1 : Desplazamiento del display (cada vez que se escribe un dato)	S=0 : Display quieto
D=1 : Display ON	D=0 : Display OFF
C=1 : Cursor ON	C=0 : Cursor OFF
B=1 : Parpadeo del carácter en la posición del cursor	B=0 : No hay parpadeo
S/C=1 : Desplazar el display	S/C=0 : Desplazar el cursor
R/L=1 : Desplazamiento a la derecha	R/L=0 : Desplazamiento a la izquierda
DL=1 : Configurar display a 8 bits	DL=0 : Configurar display a 4 bits
BF=1 : Display ocupado en una operación interna	BF=0 : Display listo para aceptar instrucciones
	X= Indeterminado

2.8 PROGRAMACIÓN ORIENTADA A OBJETOS

La Programación Orientada a Objetos (POO u OOP según siglas en inglés) es un paradigma1 de programación que define los programas en términos de "clases de objetos", objetos que son entidades que combinan estado (es decir, datos), comportamiento (esto es, procedimientos o métodos) e identidad (propiedad del objeto que lo diferencia del resto). La programación orientada a objetos expresa un programa como un conjunto de estos objetos, que colaboran entre ellos para realizar tareas. Esto permite hacer los programas y módulos más fáciles de escribir, mantener y reutilizar.

[1]Paradigma: conjunto de teorías generales, suposiciones, leyes o técnicas de que se vale una escuela de análisis o comunidad científica para evaluar todas las cosas.

De esta forma, un objeto contiene toda la información, (los denominados atributos) que permite definirlo e identificarlo frente a otros objetos pertenecientes a otras clases. A su vez, dispone de mecanismos de interacción (los llamados métodos) que favorecen la comunicación entre objetos (de una misma clase o de distintas), y en consecuencia, el cambio de estado en los propios objetos. Esta característica lleva a tratarlos como unidades indivisibles, en las que no se separan información (datos) y procesamiento (métodos).

Es por esta propiedad de conjunto entre datos y métodos, por las que el programador debe pensar indistintamente en ambos términos, ya que no debe dejar de lado nunca a los atributos en favor de los métodos, ni viceversa. Es cierto que una clase de objetos al contar con una serie de atributos definitorios, requiera de unos métodos para poder tratarlos, pero no hay que considerarlos a estos en un segundo plano ya que ambos conceptos están íntimamente entrelazados.

2.8.1 Diferencias entre la POO y la programación tradicional.

Las principales diferencias entre la programación tradicional (también llamada modular, procedural, imperativa) y la orientada a objetos son:

- La programación orientada a objetos es más moderna, es una evolución de la programación tradicional que plasma en el diseño de una familia de lenguajes conceptos que existían previamente con algunos nuevos.

- La programación orientada a objetos se basa en lenguajes que soportan sintáctica y semánticamente la unión entre los tipos abstractos de datos y sus operaciones (a esta unión se la suele llamar clase).

- La programación orientada a objetos incorpora en su entorno de ejecución mecanismos tales como el polimorfismo y el envío de mensajes entre objetos.

2.8.2 Ventajas y desventajas de los lenguajes orientados a objetos

2.8.2.1 Ventajas

- Un lenguaje orientado a objetos beneficia el desarrollo de programas extensos y sofisticados.
- Permite reutilizar y mantener el código, reduciendo el código de programación y ofreciendo la posibilidad para crear programas extensibles

- Permite el concepto de encapsulamiento, de esta manera se evita que modificaciones locales de código afecten a otros procedimientos.

- Un lenguaje orientado a objetos reduce las líneas de código, sentencias de bifurcación y módulos que son más comprensibles porque reflejan de forma clara la relación existente entre cada concepto a desarrollar y cada objeto que interviene en dicho desarrollo

2.8.2.2 Desventajas

- Requiere que el usuario tenga conocimiento de amplias librerías de clases, pertenecientes al lenguaje orientado a objetos antes, antes de empezar cualquier desarrollo.

- La ejecución de un programa orientado a objetos es más lenta porque se aprovecha todos los recursos de la plataforma utilizada para el desarrollo de dicho programa.

2.8.3 Conceptos de la Programación orientada a Objetos.

La programación orientada a objetos es una nueva forma de programar que trata de encontrar solución a problemas existentes en la programación tradicional.

Introduce nuevos conceptos, que superan y amplían conceptos antiguos ya conocidos. Entre ellos destacan los siguientes:

- **Objeto**: entidad provista de un conjunto de propiedades o atributos (datos) y de comportamiento o funcionalidad ("métodos").

- **Clase**: definiciones de las propiedades y comportamiento de un tipo de objeto concreto. La instanciación es la lectura de estas definiciones y la creación de un objeto a partir de ellas.

- **Método**: algoritmo asociado a un objeto (o a una clase de objetos), cuya ejecución se desencadena tras la recepción de un "mensaje". Desde el punto de vista del comportamiento, es lo que el objeto puede hacer. Un método puede producir un cambio en las propiedades del objeto, o la generación de un "evento" con un nuevo mensaje para otro objeto del sistema.

- **Evento**: un suceso en el sistema (tal como una interacción del usuario con la máquina, o un mensaje enviado por un objeto). El sistema maneja el evento enviando el mensaje adecuado al objeto pertinente.

- **Mensaje**: una comunicación dirigida a un objeto, que le ordena que ejecute uno de sus métodos con ciertos parámetros asociados al evento que lo generó.

- **Propiedad o atributo**: contenedor de un tipo de datos asociados a un objeto (o a una clase de objetos), que hace los datos visibles desde fuera del objeto, y cuyo valor puede ser alterado por la ejecución de algún método.

- **Herencia**: es la propiedad que permite a los objetos construirse a partir de otros objetos ya creados, es decir, la herencia permite definir nuevas clases, llamadas “clases derivadas”, a partir de otras clases llamadas “clases bases”.

2.9 TECNOLOGÍA .NET

.NET es un proyecto de Microsoft para crear una nueva plataforma de desarrollo de software con énfasis en transparencia de redes, con independencia de plataforma y que permita un rápido desarrollo de aplicaciones.

Entender exactamente que significa .NET puede ser relativamente complejo, esto se debe a que cada usuario entiende la tecnología desde sus propias perspectivas. Desde la perspectiva de un desarrollador, .NET es realmente 3 cosas: el .NET Framework, .NET Framework SDK, y el ambiente de desarrollo (primeramente Visual Studio .NET, luego Visual Studio .NET 2003, y ahora Visual Studio 2005).

El .NET Framework proporciona un conjunto central de funcionalidad para las aplicaciones, ofreciendo servicios y otras características para que el código sea ejecutado en un ambiente controlado.

2.9.1 POO, base de Microsoft .NET Framework 2.0

Todos los lenguajes de programación de la plataforma .NET, entre ellos Visual Basic 2005, son lenguajes de programación orientados a objetos.

El conjunto de librerías y el propio corazón de .NET que permite compilar, depurar y ejecutar aplicaciones .NET se denomina Microsoft .NET.

Desde que apareció Microsoft .NET, han aparecido tres versiones de Microsoft .NET Framework. La versión Microsoft .NET Framework 1.0 apareció en primer lugar y fue la que se utiliza dentro de Visual Studio .NET 2002. Poco más tarde apareció Microsoft .NET Framework 1.1 que fue integrada en Visual Studio .NET 2003. Actualmente, Microsoft ha desarrollado la versión Microsoft .NET Framework 2.0 que es la versión que se utiliza en Visual Studio 2005 y en las versiones Express Edition de la nueva familia de entornos de desarrollo rápido de Microsoft. Adicionalmente, SQL Server 2005 utiliza también esta versión de .NET.

Los diferentes lenguajes de programación de la plataforma, comparten el mismo entorno, normas, reglas, y librerías de Microsoft .NET Framework.

2.9.2 Elementos de Microsoft .NET Framework

2.9.2.1 SDK

El SDK es conocido también como Software Development Kit, y es el paquete que permite compilar, ejecutar, depurar y utilizar las bibliotecas de clases de .NET en nuestras aplicaciones, lo que nos facilita una enorme cantidad de trabajo.

2.9.2.2 BCL o Base Class Library

El BCL o bibliotecas de clases de .NET son un enorme conjunto de clases, más de 4000, que poseen una amplia funcionalidad y que sirven para desarrollar cualquier tipo de aplicación. Adicionalmente, el usuario puede desarrollar sus propias clases y con eso, puede contribuir con su experiencia al desarrollo de la aplicación.

Microsoft .NET Framework, no deja de ser por lo tanto en parte, un enorme repositorio de clases listas para usar. En .NET Framework, se hace referencia a las BCL mediante lo que se ha denominado Namespace -Espacios de Nombres- y que se engloban dentro del Namespace System.

2.9.2.3 CLR o Common Language Runtime

Una de las partes fundamentales de Microsoft .NET Framework, es el CLR o Common Language Runtime, que no es otra cosa que el entorno o motor de ejecución de lenguaje común.

Todo código escrito en .NET es ejecutado bajo el control del CLR como código administrado. Es aquí dónde encontramos una de las diferencias más notables entre las versiones de Visual Basic anteriores a .NET y las versiones de Visual Basic que tienen que ver con la plataforma .NET. Antes de .NET, las aplicaciones desarrolladas con Visual Basic se ejecutaban como código no administrado, mientras que las aplicaciones desarrolladas con Visual Basic bajo el entorno .NET, se ejecutan como código administrado, código administrado siempre por el CLR.

Dentro del CLR se encuentra el CTS o Common Type Specification, o Especificación de Tipos de Datos Común. El CTS lo forma los tipos y definiciones de cada tipo de dato utilizable en una aplicación .NET. Cada tipo de dato, hereda su tipo del objeto System.Object. El CTS está por otro lado, relacionado con el CLS o Common Language Specification, o lo que es lo mismo, la Especificación Común de Lenguajes, que son las reglas que hay que seguir a la hora de trabajar con los tipos de datos.

Un elemento adicional del CLR es el Garbage Collector o GC, que en su traducción más o menos exacta, se define como Recolector de Basura, y que tiene las funciones de gestor de limpieza de .NET eliminando de la memoria, todos aquellos objetos que no sean útiles en un momento dado, liberando al sistema de

recursos no utilizables. La ejecución del GC es una ejecución desatendida y trasparente por el programador y por el usuario.

2.9.2.4 MSIL

MSIL o IL es conocido como Microsoft Intermediate Language o simplemente Intermediate Language, o lo que es lo mismo, lenguaje intermedio. Todos los lenguajes administrados de la plataforma .NET, deben cumplir un conjunto de reglas y normas, y parte de este ajuste, es que una aplicación escrita en un lenguaje de programación determinado, debe ser compilada en un lenguaje intermedio, de manera tal, que una aplicación escrita por ejemplo en C# y otra igual en Visual Basic, se compilan al prácticamente el mismo lenguaje intermedio.

2.9.2.5 JIT

JIT son las siglas de Just In Time, o lo que es lo mismo, el procedimiento de .NET mediante el cuál, una aplicación compilada en código intermedio, es compilada cuando se lanza y ejecutada en última instancia de acuerdo al compilador que transformará el IL en instrucciones de ensamblador específicas para el sistema operativo en el cuál se está ejecutando.

2.10 MANEJO DE ARCHIVOS

Una de las principales funciones de un Sistema Operativo es la administración del almacenamiento de información, para lo cual es necesario contar con un “Sistema de Archivos”. Con este término se hace referencia, por un lado, a los mecanismos y estructuras que el sistema operativo utiliza para organizar la información en medios físicos tales como discos y diskettes (aspecto físico del sistema de archivos), y por otro a la visión que es ofrecida al usuario para permitir la manipulación de la información almacenada (una abstracción, o perspectiva lógica del sistema de archivos).

2.10.1 Conceptos de Entrada- Salida por archivos en .NET Framework

El espacio de nombre “System.IO” contiene las clases File y Directory, que proveen al .NET Framework la funcionalidad para manipular archivos y directorios. Asociadas con estas clases se encuentran las clases FileInfo y DirectoryInfo, las cuales permiten desarrollar aplicaciones de gran desempeño, gracias al uso de la característica “My”, siendo esta característica un atajo que proporciona Visual Basic 2005 para acceder a varios recursos del PC de manera rápida.

2.10.2 Definición de un Stream

El .NET Framenwork usa streams como soporte para la lectura y escritura de archivos. Se puede imaginar un stream como un conjunto unidimensional de datos contiguos, que tienen un inicio y un fin, en donde el cursor indica la posición actual en el stream como se indica en la **Figura. 2.13**.

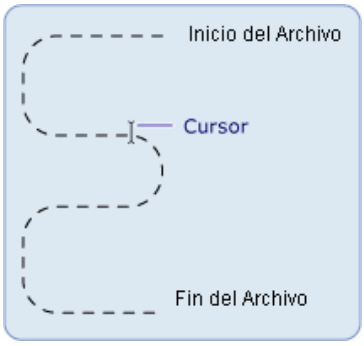


Figura. 2.13. Stream

2.10.2.1 Operaciones del Stream

Los datos contenidos en el stream pueden provenir de memoria, un archivo, o un socket TCP-IP. Los streams tienen operaciones fundamentales que pueden aplicarse a ellos:

- **Lectura.** Es posible leer de un stream, transferir datos desde un stream hacia una estructura de datos, como una cadena de caracteres o un arreglo de bytes.

- **Escritura.** Es posible escribir en un stream, transferir datos desde una fuente de datos hacia el stream.

- **Búsqueda.** Es posible consultar y modificar la posición en el stream.

Tipos de Stream

En el .NET Framework, un stream es representado por la clase Stream, que forma la clase abstracta para definirlos.

Existen varios tipos de streams. Para trabajo con entrada/salida de archivos (I/O), los tipos más importantes son las clases FileStream, la cual provee una manera para leer y escribir en archivos, y la clase IsolatedStorageFileStream, la cual proporciona una vía para crear archivos y directorios en un almacén aislado. Otros streams que pueden ser utilizados con la entrada/salida de archivos incluyen:

- BufferedStream
- CryptoStream
- MemoryStream
- NetworkStream

2.10.3 Acceso a Archivos y Atributos

Es posible controlar, la manera en que los archivos son creados, abiertos, y compartidos mediante las enumeraciones2 FileAccess, FileMode, y FileShare, las cuales contienen las banderas usadas por los constructores de la clase FileStream. Por ejemplo, cuando se abre o crea un objeto FileStream, la enumeración FileMode permite especificar cuando el archivo es abierto para agregar, cuando un nuevo

[2] Enumeraciones: Son listas de constantes numéricas, asociadas con un nombre. (MSDN 2005)

archivo es creado sino existe el archivo especificado, cuando un archivo es sobrescrito, etc.

La enumeración FileMode define constantes numéricas usadas por Visual Basic 2005 para establecer modos de acceso al archivo. Los miembros de esta enumeración son:

- **Append.** El archivo es abierto para agregar datos a este. Append es el valor por defecto.

- **Binary.** El archivo es abierto para lectura/escritura binaria, la posición donde la siguiente operación de lectura o escritura toma lugar dentro del archivo, debe ser indicada como el índice de un arreglo unidimensional de bytes.

- **Input.** El archivo es abierto para acceso de escritura.

- **Output.** El archivo es abierto para acceso de lectura.
- **Random.** El archivo es abierto para lectura/escritura aleatoria, la posición donde la siguiente operación de lectura o escritura toma lugar dentro del archivo es calculada automáticamente, solo se debe indicar el número de registro a ser operado.

La enumeración FileAttributes habilita el ensamblaje de la información de un archivo específico, la cual devuelve los atributos almacenados, tales como: compresión, encriptación, oculto, solo lectura, archivo de sistema, o archivo temporal.

2.11 BASE DE DATOS

Una base o banco de datos es un conjunto de datos que pertenecen al mismo contexto almacenados sistemáticamente para su posterior uso. En este sentido, una biblioteca puede considerarse una base de datos compuesta en su mayoría por documentos y textos impresos en papel e indexados para su consulta.

En la actualidad, y gracias al desarrollo tecnológico de campos como la informática y la electrónica, la mayoría de las bases de datos tienen formato electrónico, que ofrece un amplio rango de soluciones al problema de almacenar datos.

En informática existen los sistemas gestores de bases de datos (SGBD), que permiten almacenar y posteriormente acceder a los datos de forma rápida y estructurada. Las propiedades de los sistemas gestores de bases de datos se estudian en informática.

Las aplicaciones más usuales son para la gestión de empresas e instituciones públicas. También son ampliamente utilizadas en entornos científicos con el objeto de almacenar la información experimental.

2.11.1 Tipos de bases de datos

Las bases de datos pueden clasificarse de varias maneras, de acuerdo al criterio elegido para su clasificación:

2.11.1.1 Bases de datos estáticas

Éstas son bases de datos de sólo lectura, utilizadas primordialmente para almacenar datos históricos que posteriormente se pueden utilizar para estudiar el

comportamiento de un conjunto de datos a través del tiempo, realizar proyecciones y tomar decisiones.

2.11.1.2 Bases de datos dinámicas

Éstas son bases de datos donde la información almacenada se modifica con el tiempo, permitiendo operaciones como actualización y adición de datos, además de las operaciones fundamentales de consulta. Un ejemplo de esto puede ser la base de datos utilizada en un sistema de información de una tienda de abarrotes, una farmacia, un videoclub, etc.

2.11.1.3 Bases de datos bibliográficas

Solo contienen un representante de la fuente primaria, que permite localizarla. Un registro típico de una base de datos bibliográfica contiene información sobre el autor, fecha de publicación, editorial, título, edición, de una determinada publicación, etc. Puede contener un resumen o extracto de la publicación original, pero nunca el texto completo, porque sino estaríamos en presencia de una base de datos a texto completo (o de fuentes primarias). Como su nombre lo indica, el contenido son cifras o números. Por ejemplo, una colección de resultados de análisis de laboratorio, entre otras.

2.11.1.4 Bases de datos de texto completo

Almacenan las fuentes primarias, como por ejemplo, todo el contenido de las ediciones de una colección de revistas científicas.

2.11.2 Modelos de bases de datos

Además de la clasificación por la función de las bases de datos, éstas también se pueden clasificar de acuerdo a su modelo de administración de datos.

Un modelo de datos es básicamente una "descripción" de algo conocido como contenedor de datos (algo en donde se guarda la información), así como de los métodos para almacenar y recuperar información de esos contenedores. Los modelos de datos no son cosas físicas: son abstracciones que permiten la implementación de un sistema eficiente de base de datos; por lo general se refieren a algoritmos, y conceptos matemáticos.

2.11.2.1 Bases de datos jerárquicas

Éstas son bases de datos que, como su nombre indica, almacenan su información en una estructura jerárquica. En este modelo los datos se organizan en una forma similar a un árbol (visto al revés), en donde un nodo padre de información puede tener varios hijos. El nodo que no tiene padres es llamado raíz, y a los nodos que no tienen hijos se los conoce como hojas.

Las bases de datos jerárquicas son especialmente útiles en el caso de aplicaciones que manejan un gran volumen de información y datos muy compartidos permitiendo crear estructuras estables y de gran rendimiento.

Una de las principales limitaciones de este modelo es su incapacidad de representar eficientemente la redundancia de datos.

2.11.2.2 Bases de datos de red

Éste es un modelo ligeramente distinto del jerárquico; su diferencia fundamental es la modificación del concepto de nodo: se permite que un mismo nodo tenga varios padres (posibilidad no permitida en el modelo jerárquico).

Fue una gran mejora con respecto al modelo jerárquico, ya que ofrecía una solución eficiente al problema de redundancia de datos; pero, aun así, la dificultad que significa administrar la información en una base de datos de red ha significado

que sea un modelo utilizado en su mayoría por programadores más que por usuarios finales.

2.11.2.3 Base de datos relacional

Éste es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Pese a que ésta es la teoría de las bases de datos relacionales creadas por Edgar Frank Codd, la mayoría de las veces se conceptualiza de una manera más fácil de imaginar. Esto es pensando en cada relación como si fuese una tabla que está compuesta por registros (las filas de una tabla), que representarían las tuplas, y campos (las columnas de una tabla).

En este modelo, el lugar y la forma en que se almacenen los datos no tienen relevancia (a diferencia de otros modelos como el jerárquico y el de red). Esto tiene la considerable ventaja de que es más fácil de entender y de utilizar para un usuario esporádico de la base de datos. La información puede ser recuperada o almacenada mediante "consultas" que ofrecen una amplia flexibilidad y poder para administrar la información.

El lenguaje más habitual para construir las consultas a bases de datos relacionales es SQL, Structured Query Language o Lenguaje Estructurado de Consultas, un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales.

Durante su diseño, una base de datos relacional pasa por un proceso al que se le conoce como normalización de una base de datos.

CAPÍTULO 3

DISEÑO DE HARDWARE

3.1 DIAGRAMA DE BLOQUES DEL SISTEMA

La principal característica del sistema de control de acceso es permitir manejar dos interfases de manera completamente simultánea e independiente, como se muestra en la **Figura. 3.1** Cada interfase posee un lector relacionado con sus respectivas salidas, es decir si una tarjeta es registrada y posteriormente aceptada en una interfase, esta solo activa las interfases de la misma.

Desde otro punto de vista el sistema de control de acceso, tiene dos módulos principales estos son el Módulo de Control y los Módulos de Salida. Los cuales son los encargados de realizar las funciones que caracterizan al dispositivo.

El flujo de información se observa de manera clara mediante flechas en la **Figura. 3.1**, los lectores envían al módulo principal la información de una tarjeta presentada mediante dos líneas de datos correspondientes con el formato Wiegand de 26 bits. El módulo de control revisa si dicha tarjeta se encuentra registrada en el módulo correspondiente, en caso de ser así, este envía a continuación una trama de datos seriales que llegan a los módulos de salida e indican específicamente que salidas activar.

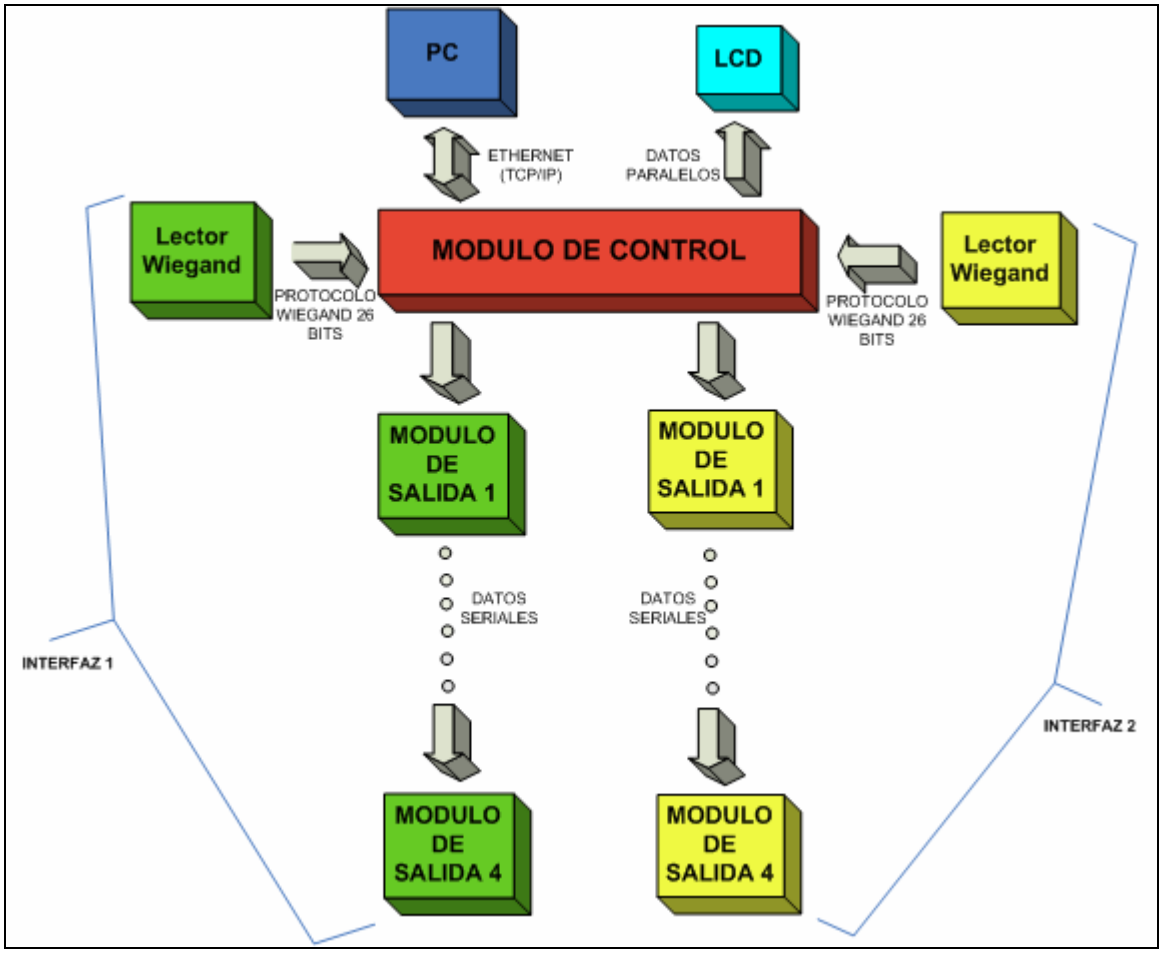


Figura. 3.1. Diagrama de bloques del sistema

Cada tarea que realice el módulo principal es informada a los usuarios mediante mensajes escritos presentados en un LCD, los datos transmitidos desde el módulo de control a este son paralelos.

Finalmente la comunicación con el PC es bidireccional, mediante Ehternet con el protocolo de comunicaciones TCP/IP.

3.2 DIAGRAMAS Y ESPECIFICACIONES DEL MÓDULO DE CONTROL

El módulo de control maneja de manera simultanea dos interfases por lo cual el controlador principal de éste posee una buena velocidad de procesamiento, así mismo necesita manejar una gran cantidad de memoria tanto volátil como no volátil, ya que la lógica del programa, el numero de usuarios que maneja y los registros que almacenan exigen una capacidad de almacenamiento elevada.

El sistema de control de acceso tiene la necesidad de conocer la fecha y hora en todo instante, con el fin de poder manejar los horarios de acceso que cada usuario del sistema posee, para lo cual es necesario el manejo de un reloj de tiempo real.

La característica de conectividad que posee el sistema de control de acceso hace necesario el uso de un puerto de conexión Ehthernet junto con las herramientas necesarias para el manejo del protocolo TCP/IP.

Para todo lo anteriormente expuesto la utilización del sistema de un sistema embebido con las características RabbitCore RCM3700, como controlador principal satisface completamente las necesidades del sistema de control de acceso.

En la **Figura. 3.2** se muestra el diagrama del módulo de control, donde se aprecia también otro diagrama de bloques correspondiente solo al controlador principal RCM 3700. Se observa que el núcleo del sistema es el procesador Rabbit 3000, el cual maneja todos los periféricos del sistema embebido y por consecuencia los periféricos del módulo de control, el procesador opera a una frecuencia de 22.1 MHz lo cual le da la rapidez necesaria para efectuar todas las tareas que exige el sistema de control de acceso, un diagrama esquemático de este se presenta en la **Figura. 3.3.**

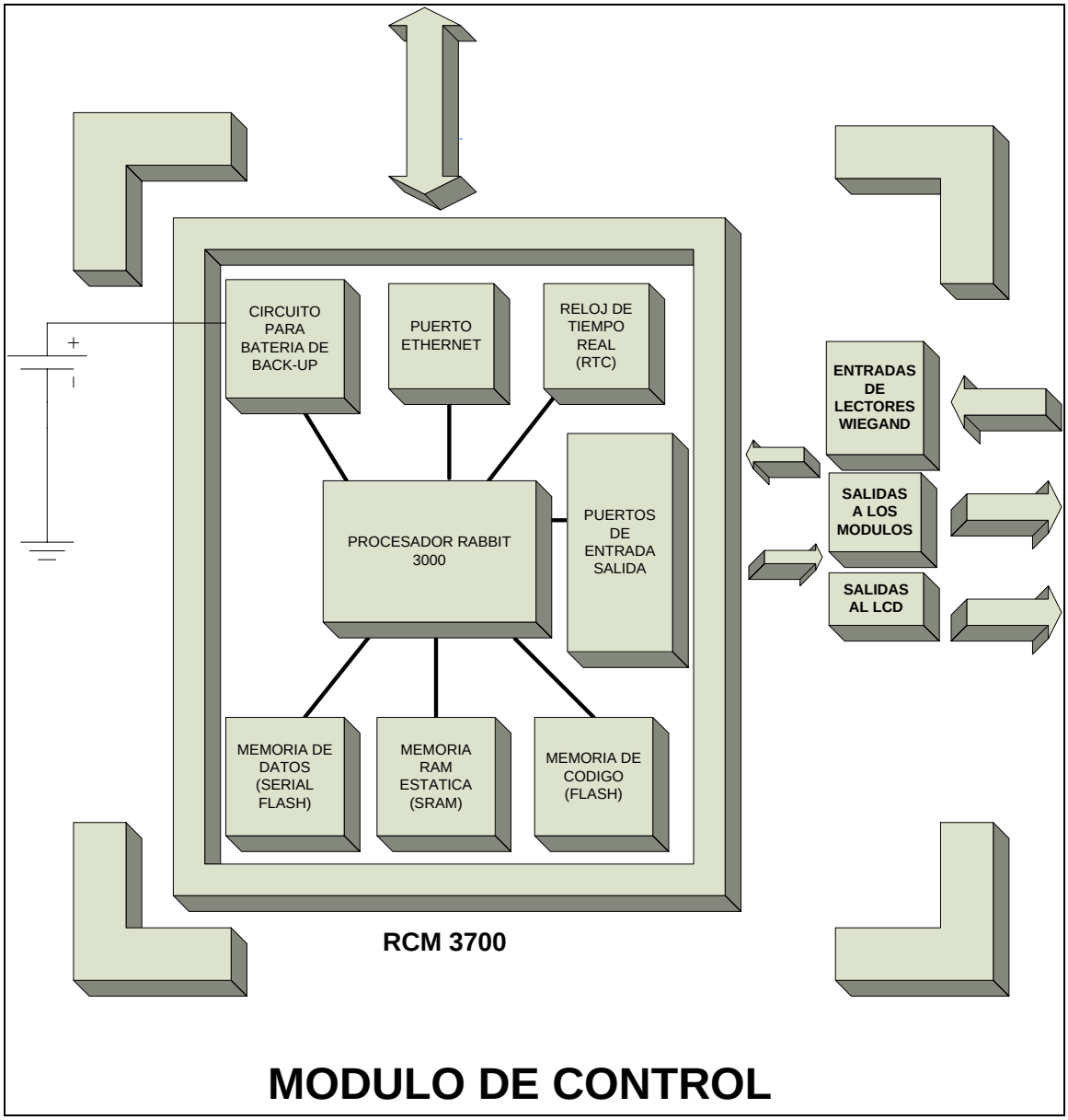


Figura. 3.2. Diagrama de bloques del módulo de control

Las memoria de código del controlador principal es de 512KB al igual que la memoria SRAM, así mismo posee una memoria Flash Serial con capacidad de 1MB para almacenar datos críticos del sistema, el reloj de tiempo real también esta integrado en el controlador RCM 3700 lo cual hace que el manejo de fecha y horas sea mas eficiente y rápido, cabe acotar que un circuito para una batería de back-up

permite mantener la operación del reloj de tiempo real en el caso de una perdida de energía en el sistema, además de esto dicho circuito alimenta la memoria SRAM para mantener los datos que en esta se encuentran, finalmente la comunicación del módulo principal con el PC también se encuentra integrada en el sistema embebido RCM 3700, ya que posee un puerto Ethernet controlado por el chip RTL8019AS, el cual provee de una conectividad de hasta 64Kbps utilizando el protocolo TCP/IP.

La información que fluye desde y hacia los puertos de entrada/salida del RCM 3700, viene de las entradas de los lectores, y va hacia las salidas a los módulos y al LCD.

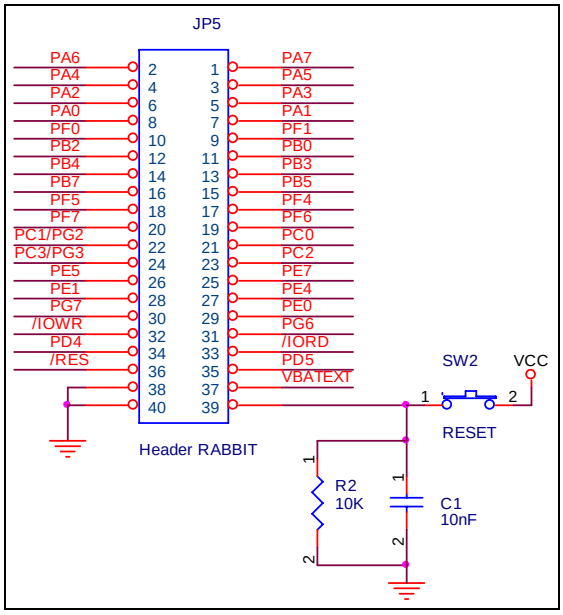


Figura. 3.3. Diagrama esquemático del controlador principal RCM 3700

La **Figura. 3.4** muestra el diagrama esquemático de las conexiones de los periféricos externos del módulo de control.

El puerto A del RCM 3700 solo puede ser configurado como un puerto de entrada o salida total, es decir todos los pines del puerto deben tener la misma dirección y no pueden ser configurados de manera individual, este es utilizado para enviar datos al LCD, a mas de esto los pines PB7, PB4 y PG7 se conectan a RS, R/W y E del LCD, para lo cual deben ser configurados como salidas TTL normales. Para finalizar con la conexión del LCD, se tiene un potenciómetro que permite controlar el contraste de este así como un interruptor que activa la luz de fondo para una mejor visualización, en caso de ser necesario.

Las interfases para los lectores se conectan al RCM 3700 mediante los pines PE0, PE4 que representan DATA0 y DATA 1 del lector 1, y PE1, PE5 que son DATA0 y DATA1 del lector 2. Estos pines tienen poseen la característica de activar interrupciones externas en el controlador principal, lo cual los hace muy útiles para captar los datos transmitidos por los lectores en cualquier instante, sin importar la tarea que este realizando el controlador.

Para cada una de las interfases de salida se utilizan 3 pines, ya que los datos son enviados de manera serial a los módulos de salida expandibles, una explicación detallada de este proceso se encuentra dada en la sección Diagramas y Especificaciones del Módulo de Salida, mientras tanto solo se debe acotar que los pines PB5, PB3, PB3 y PF4, PF6, PF5 (DATA, OE y CLK respectivamente para cada interfaz), son configurados como salidas TTL.

La señal DATA, transporta los datos de activación de pisos para las 32 salidas de los 4 módulos, el controlador transmite 32 bits (DATA=1, indica salida activada y DATA=0, indica salida inactiva) que luego son interpretados en los módulos de salida para activar o no las salidas físicas.

CLK, es una señal de reloj enviada por el módulo de control a los módulos de salida para sincronizar la señal DATA.

OE es activada luego que toda la trama de 32 bits haya sido enviada, y se mantiene así durante el tiempo de activación que haya sido configurado para las salidas de cada interfase (OE trabaja con lógica inversa).

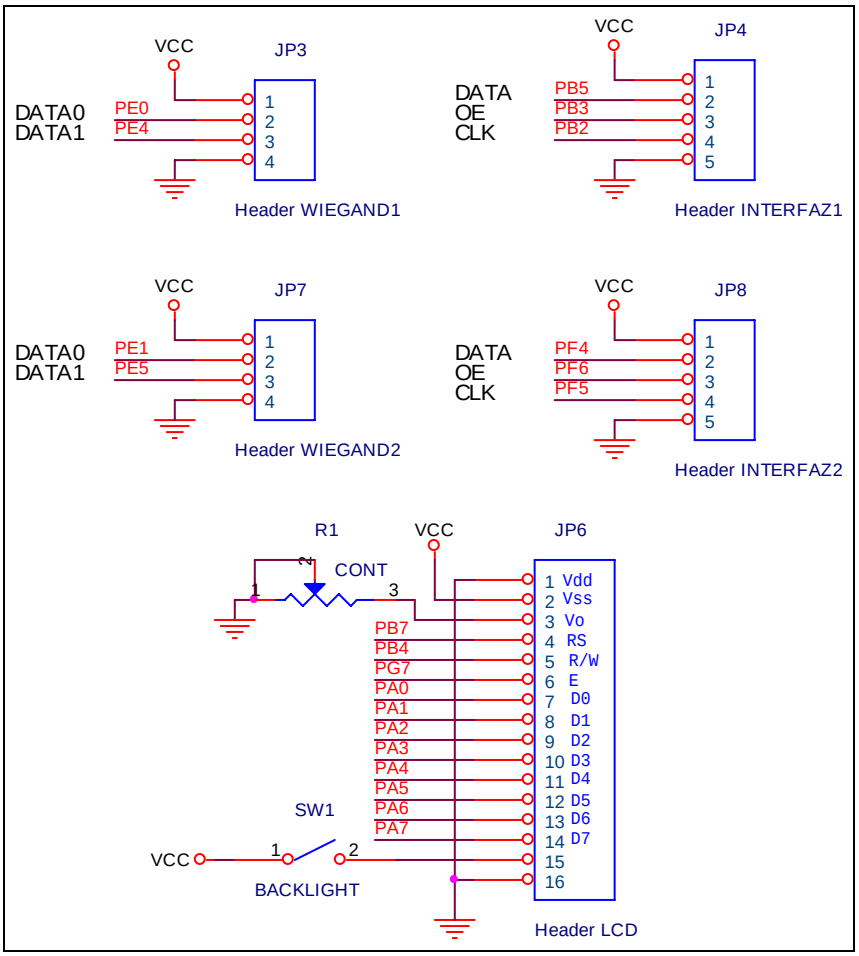


Figura. 3.4. Diagrama esquemático de los lectores, interfases de salida y LCD

3.3 DIAGRAMAS Y ESPECIFICACIONES DEL MÓDULO DE SALIDA

Los módulos de salida deben ser expandibles, y dado que cada interfase del sistema de control de acceso tiene la capacidad de manejar un máximo 32 salidas, se debe vivir a estas en 4 módulos cada uno con 8 salidas.

La **Figura. 3.5** muestra el diagrama de bloques de un módulo de salida, la información ingresa por el puerto de entrada ya sea directamente del módulo de control o de un módulo de salida anterior, dicha información son datos seriales que deben ser convertidos a datos paralelos en un bloque de conversión serial a paralelo, dichos datos posteriormente pasan a una interfase de activación de salidas físicas. La información que no corresponda a las salidas del módulo actual pasa al puerto de expansión para ser transmitida hacia el siguiente módulo de salida en el mismo formato serial que ingreso.

Este diseño asegura de manera práctica la modularidad y facilidad de expansión de sistema de control de acceso.

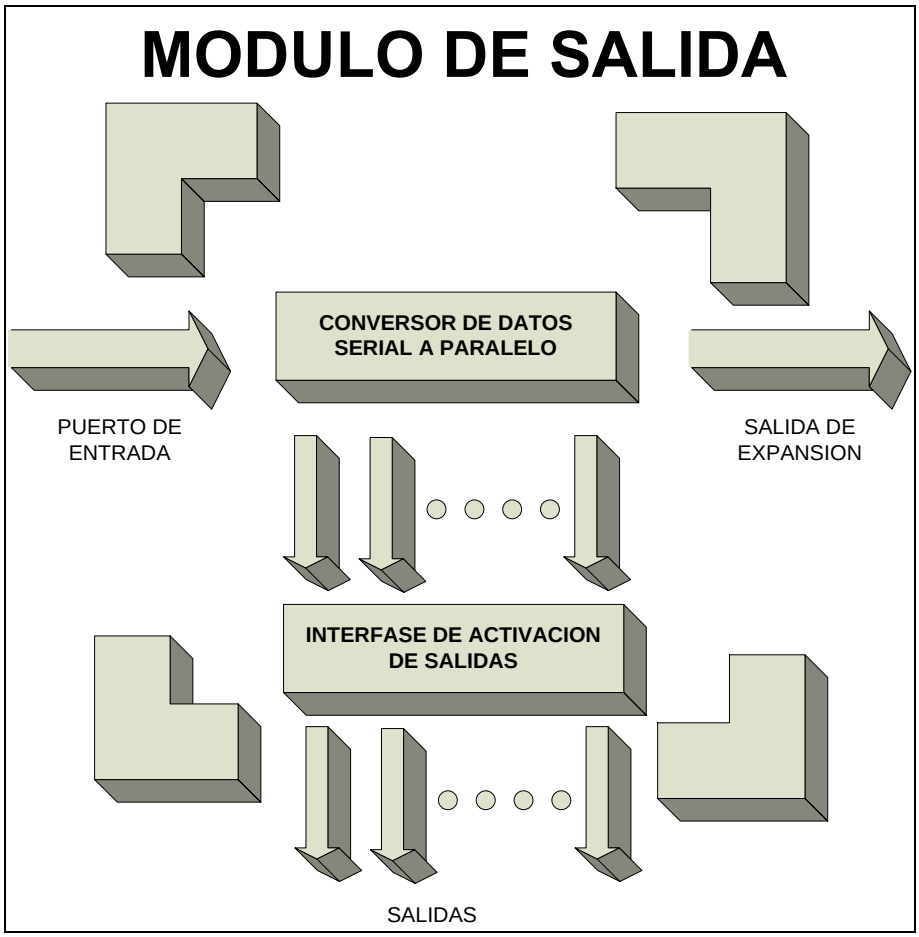


Figura. 3.5. Diagrama de bloques del módulo de salida

El bloque de conversión serial a paralelo representa un registro de desplazamiento serial 74LS299 como se muestra en el diagrama esquemático de la **Figura. 3.6.** El registro permite mantener y desplazar los datos recibidos seriamente, la señal DATA se conecta a su entrada serial esta señal provee los datos de los pisos como ya se menciono en la sección **DIAGRAMAS Y ESPECIFICACIONES DEL MÓDULO DE CONTROL,** la señal CLK ingresa al pin de reloj del registro, cada vez que esta señal tiene una transición de flanco positivo el dato que se encuentra en la entrada serial ingresa al registro y el ultimo dado de este sale hacia el pin de salida serial, el cual se conecta al puerto de expansión de salida permitiendo transmitir la señal DATA a los siguientes módulos, una vez transmitida toda la trama de 32 bits, la señal OE que se conecta con los pines OE de todos los registros registro es llevada a nivel bajo eliminando así el estado de alta impedancia de las salidas paralelas de los registros 74LS299 de todos los módulos de salida conectados, logrando así que todas las salidas que deben ser activadas lo hagan de manera simultanea. Cuado el tiempo de activación ha transcurrido la señal OE es llevada a nivel alto, volviendo así las salidas del registro a estar en alta impedancia.

Las salidas paralelas del registro 74LS299 están conectadas a las entradas del arreglo de transistores Darlington ULN2803 el cual se encarga de manejar los relés que activan las salidas físicas de los módulos.

En la **Figura. 3.7** se presenta un diagrama esquemático parcial de este arreglo Darlington, aquí se puede apreciar que está diseñado con las protecciones necesarias para manejar cargas inductivas, por lo cual su utilización en los módulos de salida es optimo, pues soporta hasta un voltaje máximo colector-emisor de 50 V con una corriente de colector de 500mA y sus entradas pueden ser activadas directamente por niveles de voltaje tipo TTL o CMOS sin problemas.

Los relés están conectados de tal manera que las cada salida del módulo ofrece un contacto físico puro y un punto de 5 V cada vez que es activada.

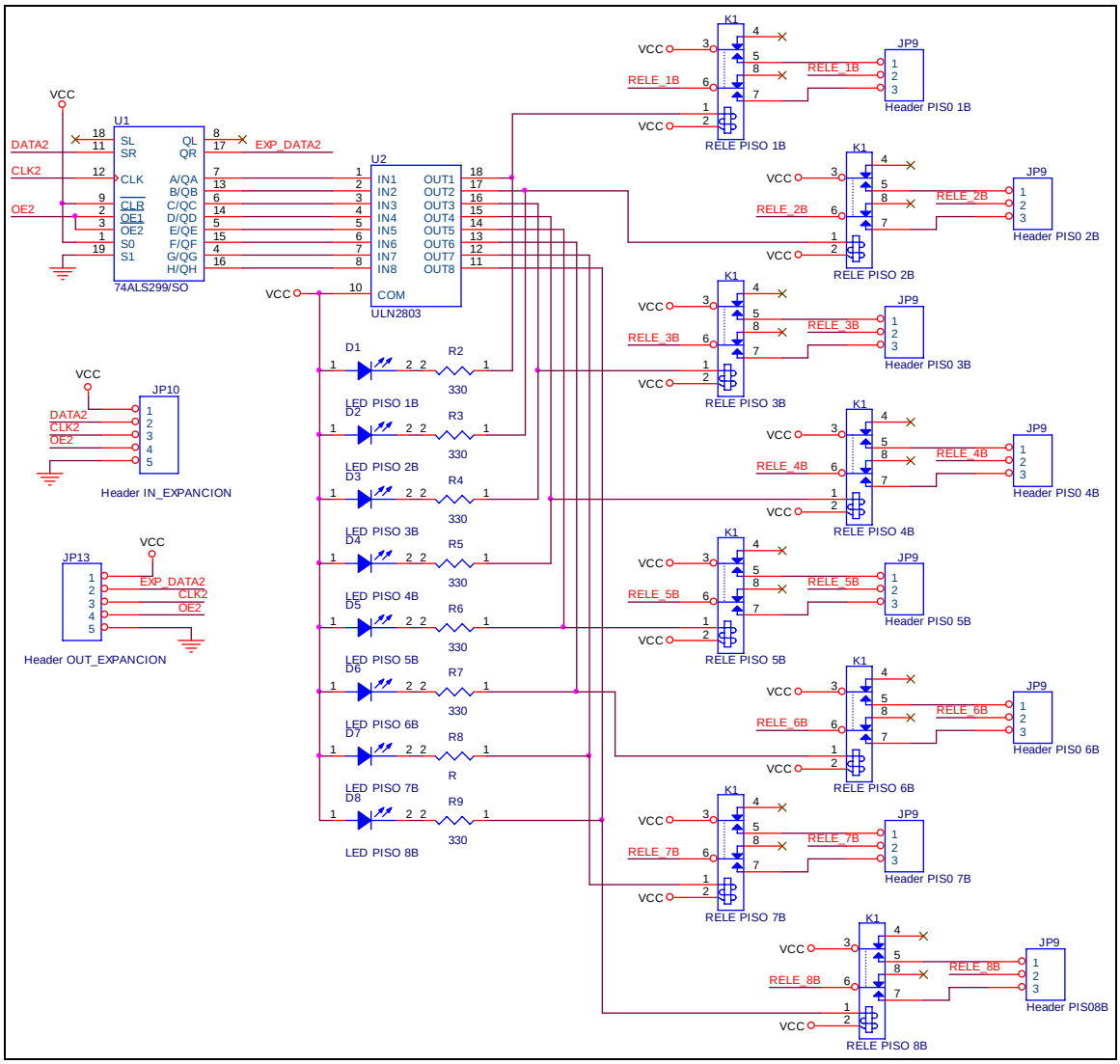


Figura. 3.6. Diagrama esquemático del módulo de salida

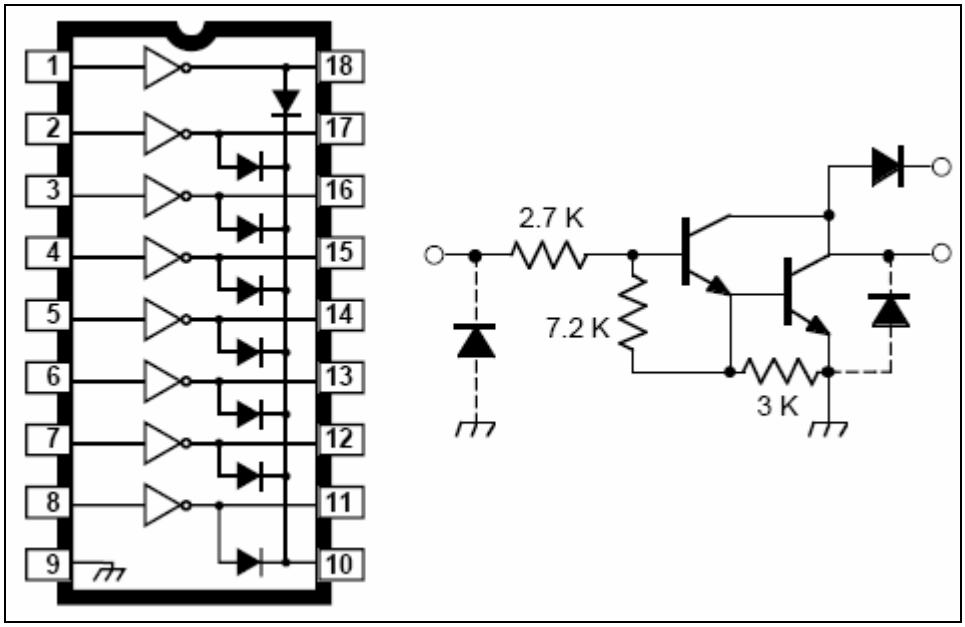


Figura. 3.7. Diagrama esquemático del arreglo Darlington ULN2803

CAPÍTULO 4

DESARROLLO DE SOFTWARE

4.1 DESARROLLO DEL SOFTWARE PARA EL CONTROLADOR

El controlador principal debe realizar varias tareas de manera simultánea con el fin de lograr manejar dos interfases de control en tiempo real. Por lo cual la lógica del software a implementar debe estar orientada a realizar multitareas.

Cada tarea comparte el control del programa y mientras se encuentra esperando algún resultado o ingresa a un bucle que consuma demasiado tiempo las tareas deben soltar dicho control voluntariamente para que otras se ejecuten de manera paralela. Esto implica que el control del programa es entregado a una tarea sin tomar en cuenta su importancia para el sistema y cualquiera puede tomar el control en cualquier momento, no existen prioridades. Así, cada una de dichas tareas debe ser estructurada de manera que no existan puntos críticos en ellas y tratando que todas tengan el mismo nivel de importancia con el fin de lograr una optima operación del sistema.

Cuando el control del programa retorna a una tarea que no ha finalizado, esta comienza e ejecutarse desde el punto siguiente donde antes se soltó el control, esto asegura una correcta secuencia de ejecución de la lógica del programa.

La mayoría de las tareas se ejecutan de manera continua mientras este operando el controlador, por lo cual es necesario ubicarlas dentro de un bucle infinito, asegurando su re ejecución una vez que hayan terminado.

La lógica de cada una de las tareas que realiza el controlador dentro del sistema de control de acceso se muestra en los diagramas de flujo de la **Figura. 4.1**, **Figura. 4.2** y **Figura. 4.3** y las mismas se encuentran detalladas en las secciones posteriores**.**

El controlador principal RCM 3700 permite su programación en leguaje C mediante el compilador provisto por Dynamic C.

Dynamic C es un sistema de desarrollo integrado para la escritura de software embebido, específicamente diseñado para usarse en controladores basados en microprocesadores Rabbit.

La estructura del software del controlador se encuentra esquematizada en los diagramas de flujo de un conjunto de tareas paralelas que se ejecutan dentro de un bucle infinito, proveyendo al sistema la capacidad de realizar varias funciones de manera simultanea, evitando los retardos o bloqueos en su operación.

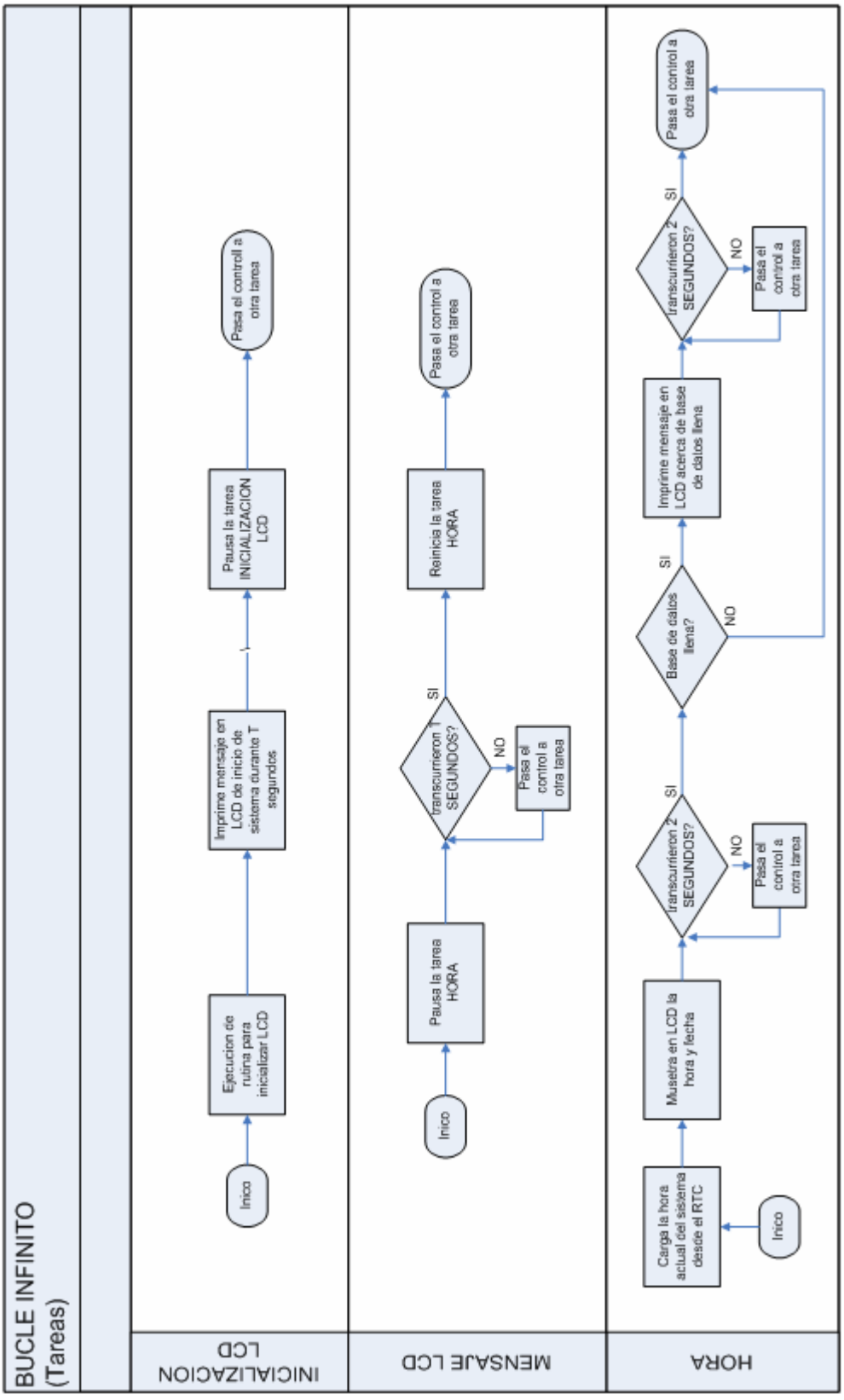


Figura. 4.1. Diagrama de bloques del software para el controlador

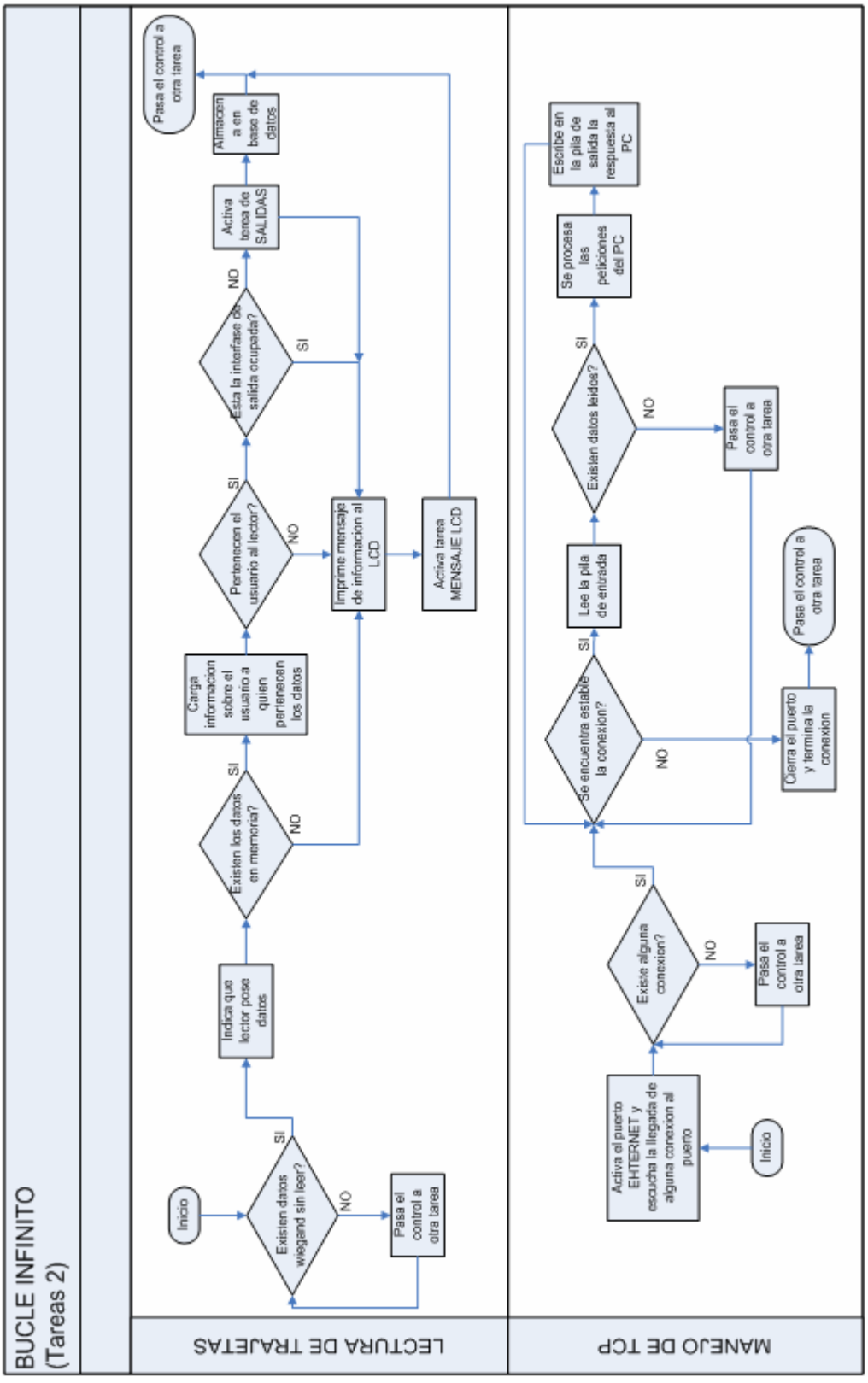
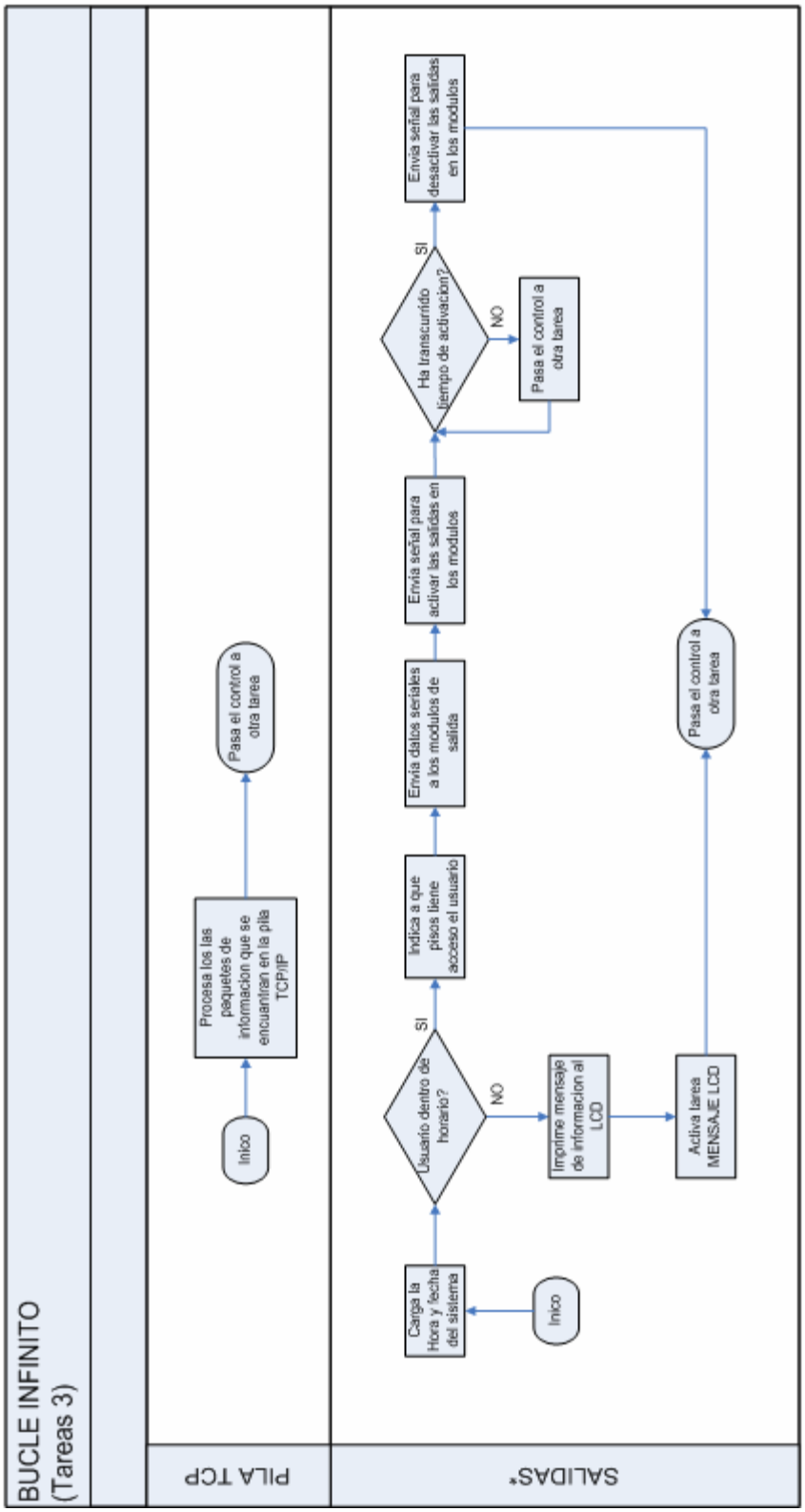


Figura. 4.2. Diagrama de bloques del software para el controlador



* Debe haber una tarea SALIDAS para cada interfase con el fin de lograr que las salidas de cada una se puedan activar simultáneamente.

Figura. 4.3. Diagrama de bloques del software para el controlador

A continuación se detalla cada uno de los procesos y tareas que realiza el controlador para dar funcionalidad al sistema de control de acceso.

4.1.1 Inicialización del sistema

En el sistema de control de acceso utiliza dispositivos periféricos tanto de entrada como de salida que le permiten interactuar con el mundo externo para desempeñar su operación.

Antes de de que las tareas comiencen a ejecutarse dentro del procesador se deben inicializar las interfases que se comunican con dichos dispositivos, con el fin de que la correcta operación de estos se asegure evitando posibles fallas criticas en el sistema.

Dichas interfases se presentan a continuación:

- *Interfase de los Módulos de Salida*. Los puertos de E/S del controlador deben configurarse para que este pueda comunicarse con los módulos de control. Estos periféricos solo reciben señales envidadas desde el módulo de control y se inicializan de la como se muestra en la **Tabla. 4.1**.

Tabla. 4.1. Inicialización de interfases para módulos de salida.

Señal	Puerto	Pin	E/S	Estado inicial
DATA1	B	5	Salida	Inactivo
OE1	B	3	Salida	Activo
CLK1	B	2	Salida	Inactivo
DATA2	F	4	Salida	Inactivo
OE2	F	6	Salida	Activo
CLK2	F	5	Salida	Inactivo

La funcionalidad de cada señal se encuentra detallada en el capitulo Desarrollo de Hardware, cabe indicar que el estado en que inician las señales sitúa a los módulos de salida en modo inactivo.

- *Interfase para LCD.* El sistema muestra mensajes de información acerca de su funcionamiento en un LCD, el cual se encuentra manejado directamente por el controlador principal.

Tabla. 4. 2. Inicialización de interfases para LCD.

Señal	Puerto	Pin	E/S	Estado inicial
D0-7	A	0-7	Salida	Inactivo
RS	B	7	Salida	Inactivo
R/W	B	4	Salida	Inactivo
E	G	7	Salida	Inactivo

El puerto A se encuentra destinado únicamente a la transmisión datos hacia el LCD mientras las otras salidas controlan las funciones del LCD. En este punto hay que notar que no se inicializa el dispositivo en si, simplemente lo hacen las interfases que lo controlan. La inicialización de las interfases para el LCD se muestra en la **Tabla. 4. 2.**

- *Interfases de Lectura Wiegand.* El módulo de control recibe información de los lectores de tarjetas, estos reciben la información en formato Wiegand de 26 bits.

Para esto el controlador utiliza 4 interrupciones externas, con el fin de que cualquiera de las líneas de datos tenga prioridad para ser leídos en cualquier

instante de la operación del sistema. La configuración de las interfases se presenta en la **Tabla. 4.3**.

Tabla. 4.3. Configuración de interfases para lectura Wiegand.

	Señal	Puerto	Pin	E/S	Formato
Interfase 1	DATA0	E	0	Salida	26 bits
	DATA1	E	4	Salida	
Interfase 2	DATA0	E	1	Salida	26 bits
	DATA1	E	5	Salida	

Cada vez que existe un dato en alguna de las líneas una interrupción es disparada en el software aquí los datos del lector que esta enviando los datos son recibidos, posteriormente se revisa si se recibió el numero de bits que corresponde al formato y se analiza los bits de paridad respectivos de la trama recibida, finalmente se forma un arreglo con los datos verificados para su posterior uso. Todo esto es manejado por la librería **wiegand.lib** la cual permite manejar hasta un máximo de 4 interfases, y provee funciones para la inicialización, verificación de datos y lectura de los mismos. La lectura de las interfases se encuentra esquematizada en la Figura. 4.1, ¡Error! No se encuentra el origen de la referencia. en la tarea **Lectura de Tarjetas** la cual se detalla en secciones posteriores.

- *Flash Serial.* El sistema embebido RCM 3700 posee una memoria flash serial de 1 MB incorporada, sin embargo este dispositivo necesita ser inicializado antes de usarlo. Se debe definir el pin que controla la selección de la memoria serial en el controlador, dicho pin es el 6 del puerto E.

El periférico es controlado por la librería sflash.lib, la cual posee funciones que inicializan al periférico y sus drivers, permiten la lectura y escritura de la memoria RAM del dispositivo y el respectivo almacenamiento y lectura de los sectores de memoria flash. Antes de inicializar el chip de memoria es indispensable que se inicialice el driver que lo manejará.

- *TCP/IP*. El control del chip Ethernet y del manejo básico del protocolo TCP/IP se realiza mediante la librería dcrtcp.lib, para inicializar el sistema se debe configurar la dirección IP del mismo y la manera en que se va a trabajar, la comunicación en el sistema de control de acceso se realiza mediante sockets, donde el controlador principal realiza la función de un servidor y escucha por un puerto predefinido la llegada de una petición de comunicación para posteriormente establecerla, procesarla y responderla. Una explicación mas detallada de la comunicación se da posteriormente en la tarea **Manejo de TCP**.

- *Memoria de Usuarios.* Con el fin de minimizar los tiempos de búsqueda de usuarios registrados dentro de la base de datos del controlador, son cargados en memoria RAM todos los datos de usuarios existentes en la memoria Flash Serial. De esta manera al manipular los datos para lectura solo se maneja a nivel de RAM y para escritura se lo hace tanto en RAM como en Flash.

4.1.2 Inicialización LCD

Esta tarea es ejecutada una sola vez a pesar de que se encuentra dentro de bucle infinito, su objetivo es ejecutarlas rutinas de inicialización del controlador LCD. Luego de esto se imprime un mensaje de inicio en el mismo, durante 4 segundos.

Antes de finalizar esta tarea, se la pone en estado de pausa de manera que esta no vuelve ha ser ejecutada durante toda la operación restante del sistema. Un diagrama de flujo de esta tarea se muestra en la **Figura. 4.2.**

4.1.3 Mensaje LCD

La tarea Mensaje LCD se encarga de mantener un mensaje en el LCD durante un tiempo determinado, impidiendo que otras tareas que manejan el dispositivo (específicamente la tarea Hora) se ejecuten durante ese lapso.

Inicialmente la tarea inicia en estado de pausa y luego de terminar su ejecución siempre vuelve a este estado, solo se activa cada vez que se desee escribir un mensaje en el LCD.

Como se indica en el diagrama de flujo de la **Figura. 4.2** al iniciarse la ejecución se pausa la tarea Hora para esperar el tiempo deseado para la presentación del mensaje antes de proceder, durante este tiempo suelta el control del programa a otras tareas. Una vez transcurrido el periodo de tiempo se reinicia la tarea Hora y finaliza la ejecución de Mensaje LCD.

4.1.4 Hora

Esta tarea esta siempre activa, sin embargo puede ser pausada y reiniciada por otras tareas, las funciones que realiza son la de presentar en el LCD la fecha y hora del día, y de ser el caso, advertir que la base de datos se encuentra llena y necesita ser descargada. La **Figura. 4.2** muestra el diagrama de flujo correspondiente a esta tarea, al inicio de su ejecución se carga e interpreta la hora y fecha desde el reloj de tiempo real (RTC) , para que esta pueda ser impresa en el LCD, luego suelta el control del programa a otras tareas en espera de que trascurran 2 segundos para la visualización del mensaje, una vez terminado el periodo de tiempo revisa si la base de datos se encuentra llena, en caso de estarlo imprime un mensaje de alerta de

esto en el LCD y al igual que antes suelta el control durante 2 segundos para finalmente terminar con su ejecución. En caso de no estar llena la base de datos simplemente la tarea termina en ese memento y espera la siguiente oportunidad para ser ejecutada.

4.1.5 Lectura de tarjetas

La tarea siempre esta activa y nuca es pausada por otra tarea, su objetivo principal es el de leer y procesar los datos de un lector. Esta tarea esta directamente relacionada con las funciones de la librería wiegand.lib como ya se menciono en secciones anteriores. La **Figura. 4.3** esquematiza la forma lógica que sigue esta tarea.

Cuando se inicia su ejecución la tarea espera a que alguno de los lectores envíe datos al módulo de control, delegando el control del programa otras hasta que se de la condición que espera. Una vez que se conforme que existan datos provenientes de los lectores, se verifica de qué lector vine el dato y posteriormente si dichos datos existen en memoria, de no ser así se imprime un mensaje de información en el LCD y se activa la tarea Mensaje LCD soltando el control y terminando la ejecución de la tarea; caso contrario se carga la información de quien pertenecen los datos leídos y se verifica si los datos pertenecen al lector que los leyó, si los datos corresponden a otro lector se presenta un mensaje de información en el LCD, se activa la tarea Mensaje LCD y finalmente se termina con la ejecución; en caso de que los datos correspondan al lector se verifica si las interfases de salida a los que pertenecen están ocupadas, si se encuentran ocupadas se al igual que en lo anteriores casos se imprime el respectivo mensaje de información, se activa la tarea Mensaje LCD finalizando la ejecución; caso contrario se activa la tarea Salidas y se almacena en la base de datos el registro la fecha y hora del momento junto con el numero de tarjeta que se leyó, con esto se finaliza la ejecución de la tarea Lectura de Tarjetas.

4.1.6 Manejo de TCP

En lo que se refiere a la comunicación con la PC el controlador principal realiza la función de un servidor que espera de la petición de un cliente, esto se realiza mediante el uso de sockets. Como se menciono en la sección de Inicialización, se debe establecer una dirección IP y abrir un puerto para la comunicación, en la **Figura. 4.3** se presenta un diagrama de flujo de esta tarea donde se observa que el controlador inicialmente escucha la llegada de alguna comunicación por el puerto y se mantiene así hasta que se de dicha condición, liberando el control del programa a otras tareas. Una vez que se tenga una comunicación en el puerto se verifica si esta se encuentra estable, de ser esto cierto se lee la pila de entrada, si no existen datos leídos se pasa el control a otras tareas y posteriormente se vuelve a verificar si la conexión es estable. Cuando existen datos leídos en la pila, se procesan las peticiones del PC y se escribiben en la pila de salida las respuestas a dichas peticiones, a continuación se vuelve a observar si la comunicación sigue estable; de no darse esta condición, el controlador cierra el puerto y finaliza la ejecución de la tarea liberando el control del programa a otras tarea.

La comunicación entre el controlador y la PC se llevan a cabo mediante un protocolo de petición y respuesta previamente definido, cuya estructura se adapta fácilmente al modelo de cliente servidor que se maneja, las tramas y la función que realizan se presentan en la **Tabla. 4.4**.

Tabla. 4.4. Protocolo de comunicación PC – Módulo de Control.

Función	Cod. Fun.	Parámetros de Petición			Parámetros de Respuesta		
		Posición	Valor	Descripción	Posición	Valor	Descripción
Igualar Reloj	10	1	0 - 59	Segundo	1	0	Función realizada sin problemas.
		2	0 – 59	Minuto			
		3	0 – 23	Hora			
		4	1 – 31	Día del mes			
		5	1- 12	Mes			

		6	0 - 147	Año (0 → 1980)			
		7	0 – 6	Día de la semana (0 → Domingo)			
Agregar Usuario en el Controlador	11	1-8	*	Código Wiegand	1	0	Función realizada sin problemas.
		9-12	0 - 2^{32}	Código de Pisos (cada bit indica un piso)		1	No hay memoria.
		13-36	0 – 23	Horarios		2	Usuario repetido.
		37	1 – 2	Lector			
Borrar un Usuario	12	1 - 8	*	Código Wiegand	1	0	Función realizada sin problemas.
						1	No existe usuario.
Borrar Bloque de Memoria	13	1	1	Bloque de Usuarios	1	0	Función realizada sin problemas.
			2	Base de Datos			
Numero de Usuarios en Memoria	14				1 - 2	0 - 4011	Numero de usuarios.
Enviar Usuarios al	15				1-8	*	Código Wiegand

PC					9-12	$0 - 2^{32}$	Código de Pisos (cada bit indica un piso)
					13-36	0 – 23	Horarios
					37	1 – 2	Lector
Tiempos de Activación de las Interfases	16	1	1 -2	Interfase	1	0	Función realizada sin problemas.
		2	120	Tiempo de activación			
Numero de Registros	17				1 - 4	0 - 70440	Numero de registros en la memoria
Enviar base de datos	18				1 - 8	*	Código Wiegand
					9	0 – 59	Segundos
					10	0 – 59	Minutos
					11	0 – 23	Horas
					12	0 – 12	Mes
					13	0 – 31	Día del mes
					14	0 - 147	Año
					15	0 – 6	Día de la semana
					16	1 – 2	Número del lector
Aviso para Almacenar Usuarios	19	1 - 2	0 – 4011	Número de usuarios a almacenar	1	0	Función realizada sin problemas.

Almacenar Usuarios (Esta función se repite dependiendo del numero de usuarios a almacenar)	20	1 – 8	-	Código Wiegand	1	0	Función realizada sin problemas.
		9 – 12	0 - 2^{32}	Código de Pisos (cada bit indica un piso)			
		13 - 36	0 – 23	Horarios			
		37	1 – 2	Lector			

* *Los 24 bits de datos del código Wiegand son enviados en grupos de 12 bits, el primero en los 4 bytes bajos y el segundo en los 4 bytes altos, con el fin de conservar compatibilidad para el uso futuro de tarjetas con mayor numero de bits.*

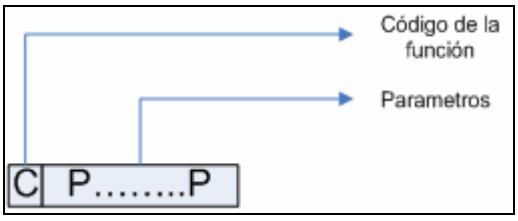


Figura. 4.4. Tramas de Petición y Respuesta

La **Figura. 4.4** muestra la composición de las tramas en la comunicación entre el módulo de control y el PC, siempre inician con el código de la función en el primer byte de la trama, el resto de bytes corresponden a los parámetros según como se indica en la Tabla

Tabla. 4.4. Protocolo de comunicación PC – Módulo de Control.. Hay que tomar en cuenta que en estas tramas de datos no se indica para quien corresponde el mensaje, esto ya lo hace el protocolo TCP/IP, por lo que lo único que resta es recibir la información de manera directa y procesarla.

4.1.7 Pila TCP

En esta tarea simplemente se procesan todos los paquetes de información de la pila protocolar correspondiente a los sockets abiertos o a las peticiones de comunicación, de manera que la información contenida en ellos al controlador se encuentre lista para ser usada en cualquier momento que sea necesario.

4.1.8 Salidas

La tarea Salidas permite activar y desactivar las interfases de salida durante un tiempo determinado, esta tarea inicia en un estado pausado y cada ocasión que es activada se ejecuta una sola vez para luego retornar a su estado inicial.

En primer lugar carga la hora y fecha del reloj de tiempo real, y determina si el usuario que ha activado los lectores tiene acceso en ese momentos, de no ser esto verdadero imprime un mensaje de información al respecto, activa la tarea Mensaje LCD, y finaliza la ejecución pasando el control del programa a otras tareas. Si el horario se encuentra dentro del horario, se indica a que pisos tiene acceso, dichos datos son transmitidos serialmente a los módulos de salida y se envía una señal para activar simultáneamente todas las salidas, realizado esto se suelta el control a otras tareas durante un tiempo de activación previamente determinado; cuando se termina el tiempo de activación, las salidas vuelven a estar desactivadas y se termina la ejecución del programa pasando el control al resto de tareas.

4.2 DISTRIBUCIÓN DE LA MEMORIA DEL SISTEMA

La memoria flash serial se subdivide en tres secciones, las cuales se muestran en la **Figura. 4.5.**

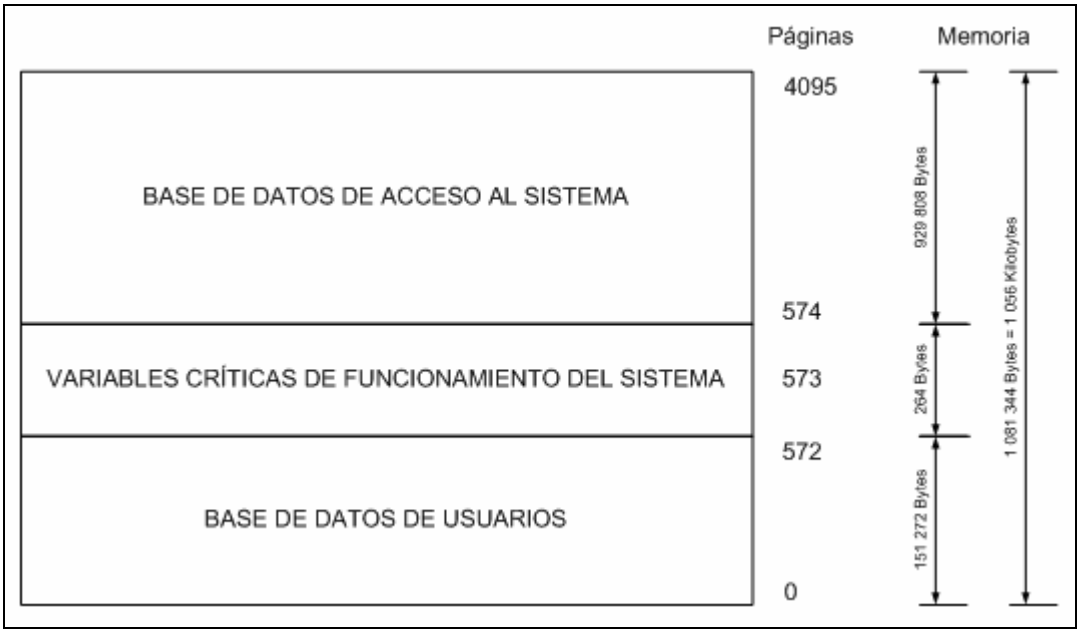


Figura. 4.5. Distribución de la memoria flash serial

4.2.1 Base de datos de usuarios

En esta sección se almacena los usuarios que se encuentran registrados en el sistema, los cuales tendrán un acceso personalizado, individualmente, al edificio. Los datos que se almacenarán por usuario son:

- Código de la tarjeta magnética (Código de factibilidad y número de tarjeta), ocupa 4 bytes.
- Número de lector al que pertenece (Lector 1, lector 2 o ambos), ocupa 1 byte.
- Código de acceso a los pisos del edificio (Cualquier combinación de hasta 32 pisos), ocupa 4 bytes.
- Horario de acceso al edificio (Cualquier combinación de los 7 días de la semana y sus 24 horas), ocupa 28 bytes.

La distribución de memoria correspondiente a esta sección se muestra en la **Tabla. 4.5**.

Tabla. 4.5. Distribución de memoria para la Base de Datos de Usuarios

Memoria reservada	151 272 Bytes
Tamaño por usuario	37 Bytes
Usuarios por bloque	7 usuarios
Memoria utilizada por bloque	259 Bytes
Páginas a utilizar	573 páginas (de la 0 a la 572)
Total usuarios	4 011 usuarios
Memoria utilizada	148 407 Bytes

4.2.2 Variables críticas de funcionamiento del sistema

En esta sección de memoria flash se almacena las variables críticas para el correcto desempeño del sistema, las cuales son:

- Número de usuarios almacenados.
- Página en la cual se escribirá la base de datos.
- Bandera para indicar que la base de datos está llena.
- Tiempo de activación de las dos interfases Wiegand.
- Número de espacios vacíos, entre los usuarios almacenados.

La distribución de memoria de esta sección está en la **Tabla. 4.6**.

Tabla. 4.6. Distribución de memoria para las Variables Críticas de Funcionamiento del Sistema

Memoria Reservada	264 Bytes
Páginas a utilizar	1 página (573)
Número de variables almacenadas	5 variables
Memoria utilizada	9 Bytes

4.2.3 Base de datos de acceso al sistema

En esta sección se almacena la Base de datos correspondiente al acceso al sistema, es decir, se almacena los datos del usuario que ha ingresado al edificio, dichos datos son:

- Código de la tarjeta magnética (Código de factibilidad y número de tarjeta), ocupa 4 bytes.
- Número de lector al que pertenece (Lector 1, lector 2 o ambos), ocupa 1 byte.
- Fecha y hora a la que el usuario (registrado) ingresó, ocupa 4 bytes.

La distribución de memoria para esta sección de memoria flash se detalla en la **Tabla. 4.7**.

Tabla. 4.7. Distribución de memoria para la Base de Datos de Acceso al Sistema

Memoria reservada	929 808 Bytes
Tamaño por dato	13 Bytes
Datos por bloque	20 datos
Memoria utilizada por bloque	260 Bytes
Páginas a utilizar	3522 páginas (de la 574 a la 4095)
Total datos	70 440 datos
Memoria utilizada	915 720 Bytes

4.3 DESARROLLO DEL SOFTWARE DE LA INTERFAZ DE USUARIO

La interfaz de usuario ha sido desarrollada en Microsoft Visual Basic 2005 Express, una evolución del lenguaje Visual Basic, que ha sido diseñado para la construcción productiva de aplicaciones seguras y orientadas a objetos. Visual Basic permite a los desarrolladores orientarse sobre Windows, Web, y dispositivos móviles. Similar a todos los lenguajes orientados al uso de Microsoft .NET Framework, los

programas escritos en Visual Basic, se benefician de la seguridad e interoperabilidad de lenguajes.

Esta generación de Visual Basic mantiene como característica la facilidad para crear aplicaciones de manera fácil y rápida, brinda compatibilidad con versiones anteriores e incorpora nuevas características que integran totalmente las aplicaciones desarrollados con el .NET framework.

Los elementos principales que brindan la funcionalidad a la interfaz de usuario incluyen conexiones TCP, manejo de archivos, almacenamiento de nombres de usuarios y password o clave.

4.3.1 Conexión tcp.

Uno de los elementos principales de la interfaz de usuario es la función TcpConnect, encargada de establecer la conexión entre el sistema de control y la PC mediante un socket TCP. En esta conexión el sistema de control actúa como servidor y la PC como cliente, de esta manera el cliente envía una petición y el servidor responde de acuerdo a las funciones establecidas en el desarrollo del programa del controlador antes descrito.

El funcionamiento general de la función TcpConnect se resume en la **Figura. 4.6** donde, una vez realizada la conexión, se procede a escribir los datos del BUFFER[3] en el STREAM y se espera por la respuesta del controlador la cual, se lee del STREAM y almacena en el BUFFER, finalizando con el cierre de la conexión.

[3] BUFFER: Área de almacenamiento temporal en un computador, en esencia se compone de un arreglo de tipos de datos primitivos como bytes, caracteres, enteros, etc.

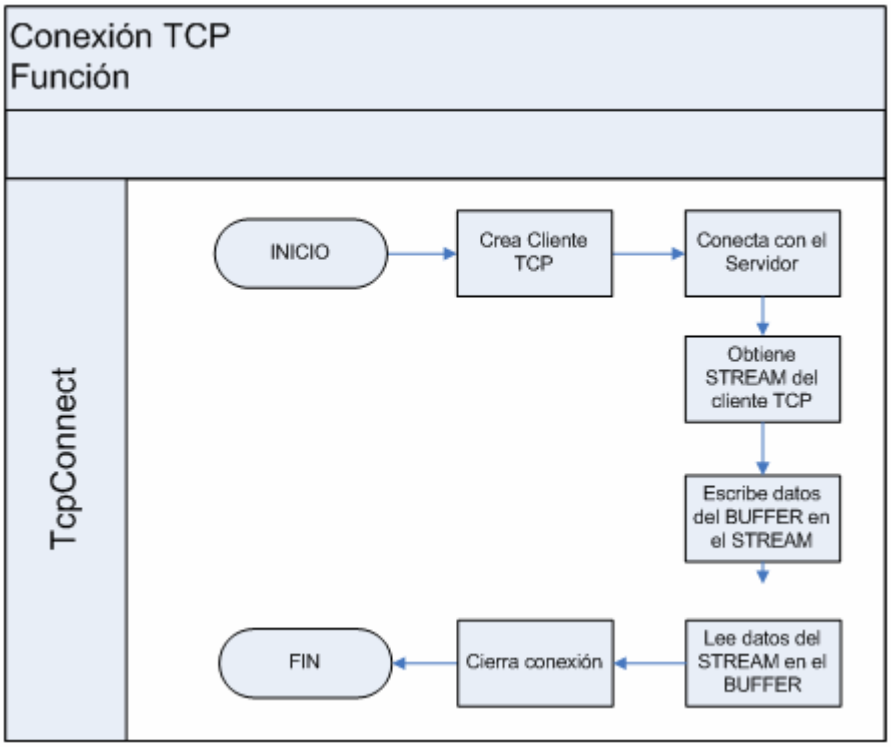


Figura. 4.6. Diagrama de flujo, Conexión TCP/IP

4.3.1.1 Prototipo de la función TcpConnect.

La función TcpConnect se compone de 3 parámetros como se detalla a continuación:

Public Function tcpconnect(func As Byte, datos() As Byte) As Byte()

Parámetros:

- Func, corresponde al primer byte del BUFFER y representa el número de función a ser usada en el controlador.

- Datos, corresponde al arreglo de bytes con que operará el controlador

La respuesta de esta función es otro arreglo de bytes que tendrá la información proporcionada por el controlador.

4.3.1.2 Elementos de TcpConnect.

La función TcpConnect es una implementación que fue desarrollada para cumplir los requerimientos de conexión entre el PC y el sistema de control, los elementos que dan soporte a la función son la clase TcpClient y la clase NetworkStream que son parte del .NET framework.

La clase TcpClient proporciona conexiones de cliente para servicios TCP de red. Las propiedades más importantes de esta clase son:

- ReceiveBufferSize: obtiene o establece el tamaño del BUFFER de recepción.
- ReceiveTimeout: obtiene o establece la cantidad de tiempo que un TcpClient esperará para recibir datos una vez iniciada la operación de lectura.
- SendBufferSize: obtiene o establece el tamaño del BUFFER de envío.
- SendTimeout: obtiene o establece la cantidad tiempo que un TcpClient esperará por la culminación exitosa de una operación de escritura.

Los métodos de la clase TcpClient utilizados son los siguientes:

- Connect: Conecta al cliente con un host remoto TCP, usando el nombre específico del host y el número de puerto.
- Close: Libera el TcpClient sin cerrar las conexiones que esta haya establecido.
- GetStream: Retorna el objeto NetworkStream usado para enviar y recibir datos.

La clase “NetworkStream” proporciona métodos para enviar y recibir datos sobre un Stream en modo de bloques. Se debe tener claro que un Stream es el flujo de datos desde una fuente a un simple receptor a través de un canal, por lo tanto un objeto NertworkStream solo puede ser creado sobre un TcpClient (socket TCP) conectado.

Los métodos de la clase NetworkStream usados para la escritura y lectura son Write y Read respectivamente. Los prototipos de estos métodos se detallan a continuación:

write(data, offset, Length)
read(data, offset, Length)

En donde data es el BUFFER que contiene la información a ser escrita o leida, offset es el desplazamiento dentro del BUFFER y length es el número de bytes a ser escritos o leidos.

4.3.2 Archivos.

Una de las funcionalidades del programa desarrollado para este proyecto requiere almacenar y manipular datos referentes tanto a información sobre usuarios así como registros de ingreso al sistema. La recopilación de esta información se encuentra almacenada en una base de datos que permite un fácil acceso y modificación por parte del usuario.

Existen varias maneras de administrar datos que pertenecen a un mismo contexto, para lo cual se usan Sistemas Gestores de Bases de Datos (SGBD), que permiten almacenar y posteriormente acceder a los datos de forma rápida y estructurada, algunos de estos sistemas son SQL Server, Microsoft Access, etc.

4.3.2.1 Ventajas y desventajas del Sistema de Archivos.

En el desarrollo de este proyecto y de manera particular se analizó las ventajas y desventajas de usar archivos en lugar de un SGBD para la administración de la base de datos, las cuales se detallan a continuación:

Ventajas:

- Un sistema de archivos no requiere de paquetes de software adicional para administrar la información, lo cual reduce los requerimientos de instalación y costo.

- El desarrollo de este proyecto requiere definir tipos de datos personalizados, los cuales pueden ser definidos en las tablas de un SGBD, pero sería ineficiente y limitado por las características de compatibilidad, mientras en un sistema de archivos la definición de un tipo personalizado de dato solo depende de la flexibilidad del lenguaje de programación.

- Un SGBD requiere una cadena de conexión para acceder a la base de datos la cual por seguridad se genera solo en tiempo de diseño, mientras el uso de archivos nos permite acceder a la información en tiempo de ejecución sin necesidad de generar una cadena de conexión.

- Este proyecto requiere de características de fácil movilidad y transporte de información por parte del servicio de mantenimiento, lo que se facilita al manejar varios archivos pequeños y no uno solo que concentre toda la información.

- El manejo de archivos en Visual Basic 2005 se beneficia de las características de seguridad e interoperabilidad del .NET framework.

Desventajas:

- El sistema SGBD cuenta con un amplio fundamento informático que permite acceder de manera rápida y estructurada a los datos almacenados, un sistema de archivos común no cuenta con esta tecnología.

- Las opciones de seguridad y acceso a la base de datos son elementos constitutivos de un SGBD, la seguridad ofrecida por un archivo es relativa a la seguridad ofrecida por el lenguaje de programación y al sistema operativo en el cual se ejecute la aplicación.

- Los SGBD cuentan con herramientas para el desarrollo de reportes de manera sencilla, mientras un sistema de archivos requiere más desarrollo para este propósito.

4.3.2.2 Creación del Archivo de Usuarios.

Para cubrir las necesidades de este proyecto se ha definido dos tipos de archivos, en este caso se detalla el diseño del archivo usado para almacenar la información de los usuarios. Semejante a un SGBD, en el cual se diseña una tabla con sus respectivos campos, en el caso de archivos se define una estructura que representa el tipo de dato personalizado que será almacenado. En la Tabla. 4.8 se detalla la estructura definida para el archivo de usuarios:

Tabla. 4.8. Estructura de archivo de usuarios

TIPO DE DATO	NOMBRE	NIVEL DE ACCESO	DETALLE
Int64	Wiegand	Público	Almacena el código wiegand de la tarjeta magnética

Int16	WieH	Público	Almacena 12 bits MSB del código wiegand. Brinda compatibilidad con el sistema de control
Int16	WieL	Público	Almacena 12 bits LSB del código wiegand. Brinda compatibilidad con el sistema de control
Byte	Facti	Público	Almacena el código de factibilidad o número de lote de la tarjeta magnética.
Int64	Pisos	Público	Almacena el código generado para el acceso a pisos
Byte	Módulo	Público	Almacena el módulo al que pertenece el usuario.

El archivo de usuarios mantiene compatibilidad tanto con la información almacenada en el sistema de control como con la información mostrada en la interfaz de usuario, de esta manera, el archivo puede ser descargado desde el PC al controlador y viceversa. La extensión escogida para representar un archivo de usuarios es COH.

4.3.2.3 Creación de Archivo de Registros.

La creación del archivo de registros de ingreso al sistema es similar al archivo de usuarios, por lo tanto la definición de la estructura que almacena los registros de ingreso al sistema es la que se detalla en la Tabla. 4.9.

Tabla. 4.9. Estructura de registros de ingreso

TIPO DE DATO	NOMBRE	NIVEL DE ACCESO	DETALLE
Byte	Segundos	Público	Almacena los segundos del ingreso
Byte	Minutos	Público	Almacena los minutos del ingreso

Byte	Horas	Público	Almacena la hora del ingreso
Byte	Dia	Público	Amacena día del mes
Byte	Mes	Público	Almacena mes del año
Integer	Anio	Público	Almacena año del ingreso
Byte	Dayweek	Público	Almacena el día de la semana
Byte	Módulo	Público	Almacena el módulo al que pertenece el usuario.
Byte	Facti	Público	Almacena el código de factibilidad o número de lote de la tarjeta magnética.
Int64	Wiegand	Público	Almacena el código wiegand de la tarjeta magnética

El archivo de registros no mantiene compatibilidad en formato con los registros del sistema de control por lo tanto solo se permite la descarga de registros de acceso desde el controlador hacia la PC para su posterior interpretación y explotación en esta. La extensión seleccionada para este archivo es COD.

4.3.2.4 Función FileOpen.

Antes de poder realizar cualquier operación de lectura/escritura sobre un archivo, este debe ser abierto mediante la función FileOpen, la cual localiza un buffer para las operaciones sobre el archivo y determina el modo de acceso a ser usado con el buffer.El prototipo de la función FileOpen es el siguiente:

FileOpen(FileNumber As Integer,
FileName As String, Mode As OpenMode,)

Parámetros:

- FileNumber: Es cualquier número de archivo válido. Se recomienda el uso de la función FreeFile() para obtener el próximo número valido de archivo.

- FileName: Es la expresión que especifica el nombre del archivo. Puede incluir el directorio y unidad lógica.

- Mode: Es una enumeración o elemento de un arreglo de constantes numéricas que especifica el modo en que el archivo es abierto. Los modos Input, Output, y Append son usados cuando se accede a los archivos de manera secuencial, como es el caso de los archivos de texto, mientras Binary se usa para acceso binario al archivo y Random se usa en acceso aleatorio.

4.3.2.5 Función FreeFile.

Retorna un valor entero que representa el próximo número de archivo para ser usado con la función FileOpen. El número de archivo representa la cantidad de archivos que existen en el directorio de trabajo, o el número de archivos abiertos simultáneamente, el cual no puede ser mayor a 255.

4.3.2.6 Función FilePut

Escribe datos de una variable a un archivo en disco. Existen varios proptotipos de está función ya que permite varias sobrecargas, el usado en este proyecto es el siguiente:

FilePut(FileNumber as integer, Value as structure)

Parámetros:

- FileNumber, es cualquier número de archivo válido, que se encuentre abierto.

- Value, es una estructura que contiene la información que será escrita en el archivo.

4.3.2.7 Función FileGet.

Lee datos de un archivo abierto en disco en una variable. Existen varios proptotipos de está función ya que permite varias sobrecargas, el usado en este proyecto es el siguiente:

FilePut(FileNumber as integer, Value as structure, RecordNumber as integer)

Parámetros:

- FileNumber, es cualquier número de archivo válido, que se encuentre abierto.
- Value, es el nombre válido de una estructura en la cual se leerá la información del archivo.

4.3.2.8 Función FileClose.

Concluye las operaciones de lectura/escritura en un archivo abierto por la función FileOpen. El prototipo de está función es:

FileClose(FileNumber as integer)

Parámetros:

- FileNumber, es cualquier número de archivo válido, que fue abierto con OpenFile. Si se omite este parámetro se cerrará todos los archivos activos abiertos con FileOpen.

4.3.3 Usuarios y Claves.

Para almacenar el nombre de usuario y password es necesario adicionar parámetros de configuración a la aplicación, esto se realiza mediante la modificación de las propiedades del proyecto dentro del ambiente integrado de desarrollo Visual Basic, estas opciones están dentro del espacio Settings como se muestra en la **Figura. 4.7.**

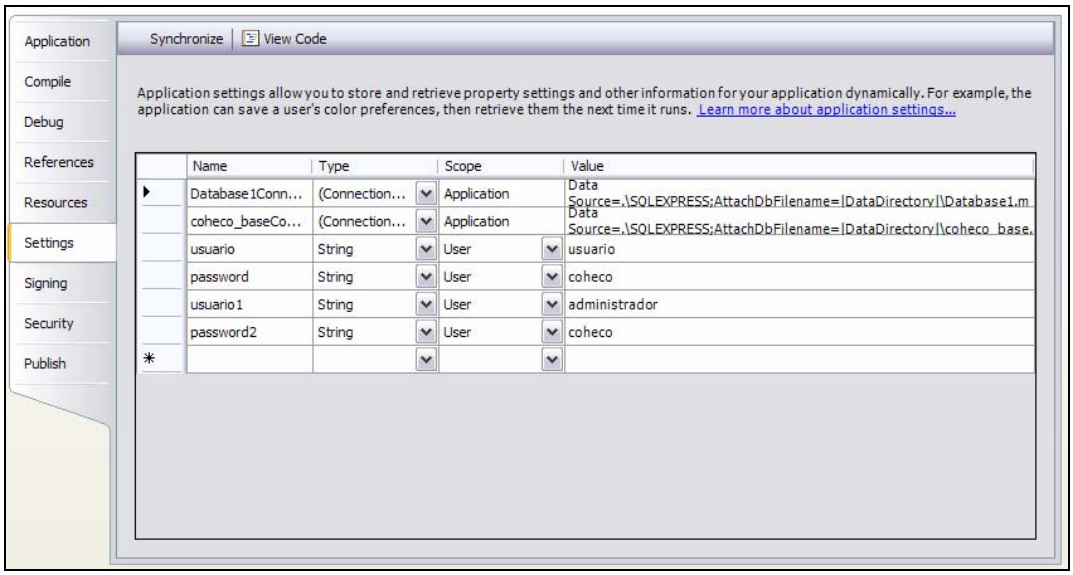


Figura. 4.7. Propiedades del proyecto, espacio settings

Los parámetros de configuración de la aplicación permiten almacenar y recuperar información dinámicamente. En el caso de este proyecto los usuarios y claves son almacenados como parámetros de configuración que pueden ser guardados y recuperados la próxima vez que la aplicación sea ejecutada.

Para leer o modificar los parámetros de configuración se usa el espacio “My”, que es una nueva característica de Visual Basic 2005 que nos permite acceder de manera rápida a varios recursos de red, del sistema y de la aplicación.

4.3.4 Ambiente Gráfico

Las opciones del programa han sido distribuidas en 3 formularios principales, con el objetivo de agrupar las funciones que tienen características comunes, estos formularios son:

- Formulario OnLine
- Formulario OffLine

- Formulario Consulta Base

4.3.4.1 Formulario OnLine

Este es el formulario principal de la aplicación que agrupa todas las funciones que operan sobre el controlador conectado a la PC mediante TCP y además proporciona el acceso a los otros formularios. Las funciones que incluye este formulario usan las tramas descritas en las secciones 4.1.6 y 4.3.1.

Las funciones que se agrupan en el formulario OnLine son:

- Conectar: Esta función comprueba que la PC este conectada a la red y que el controlador se encuentre dentro de la misma red, el funcionamiento se resume en la **Figura. 4.8.**

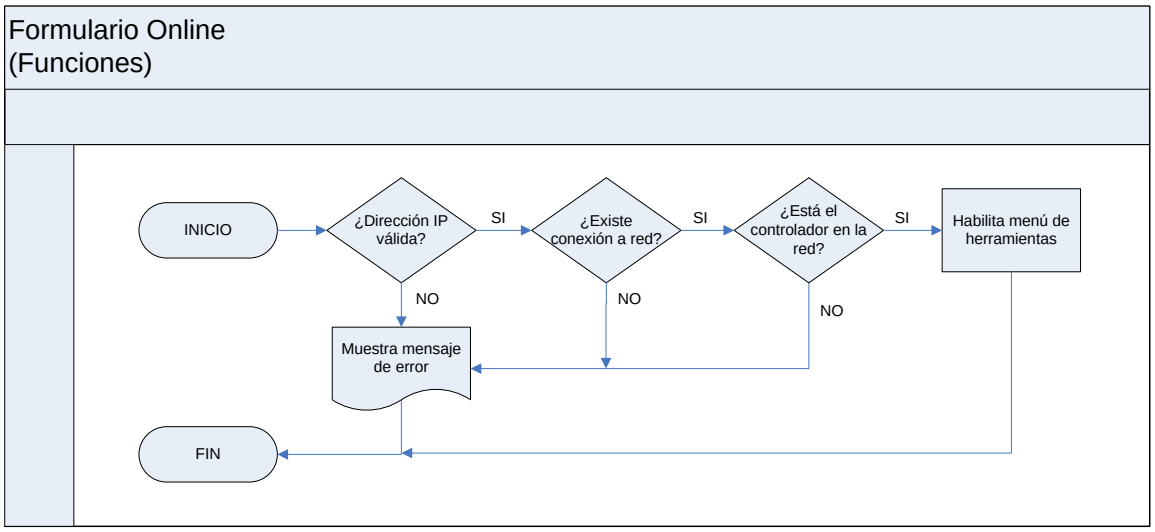


Figura. 4.8. Diagrama Función Conectar

- Actualizar Hora y fecha: Actualiza la hora y fecha en el controlador.
- Tiempo de Activación: Permite modificar la cantidad de tiempo que las interfases de salida permanecen encendidas luego de que un usuario registrado accede al sistema.
- Agregar Usuario: Permite agregar un usuario en el controlador para lo cual valida y obtiene la información ingresada en los controles del formulario OnLine, que incluyen el número de tarjeta magnética, código de factibilidad, código de pisos, código de horas y módulo wiegand. Esta función cumple con el diagrama de la Figura. 4.9.

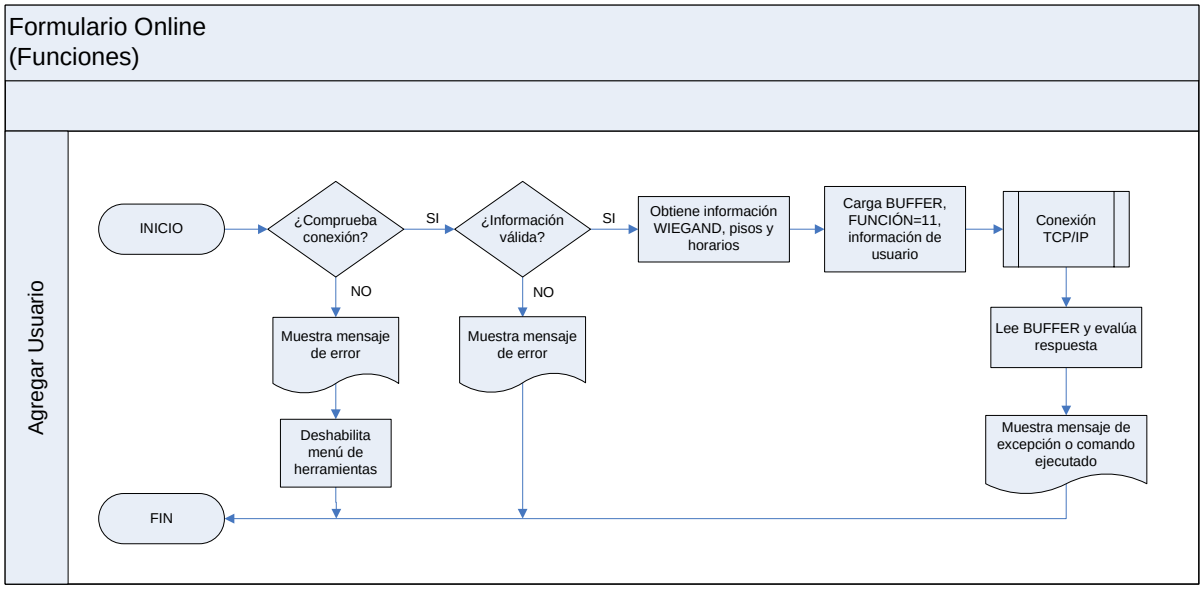


Figura. 4.9. Diagrama de flujo agregar usuario

- Eliminar usuario: Permite eliminar un usuario en el controlador a partir de la información ingresada en el formulario OnLine.
- Editar usuario: Permite editar un usuario en el controlador a partir de la información ingresada en el formulario OnLine. No define una nueva

función en el controlador, ya que es una combinación de la función Eliminar y Agregar Usuario.

- Descargar Base de datos: Permite descargar desde el controlador y almacenar en un archivo de extensión COD, todos los registros de ingreso al sistema.

- Leer usuarios del controlador: Permite descargar desde el controlador y almacenar en un archivo de extensión COH, la información de los usuarios registrados en el sistema. Esta función se encuentra protegida mediante un control de ingreso con un usuario y clave diferente al usado para el ingreso general a la aplicación.

- Grabar usuarios en el Controlador: Permite grabar la información de un archivo de extensión COH (archivo de usuarios) en el controlador. El archivo puede haber sido previamente descargado con la función “Leer usuarios del controlador” o puede ser creado con el formulario OffLine.

- Modificar Usuario: Esta función permite modificar el nombre de usuario y clave tanto para el acceso general a la aplicación como para el acceso a la función “Leer usuarios del controlador”. Para poder modificar los usuarios y claves se debe conocer los valores que actualmente están establecidos.

4.3.4.2 Formulario OffLine.

Este formulario está orientado a la escritura y edición de archivos de extensión COH, que contienen la información de usuarios, cuenta con un control ListView como

elemento principal, que permite observar los detalles del archivo COD. Las funciones agrupadas en este formulario son:

- Abrir: Abre un archivo de usuarios descargado desde el controlador, o creado previamente, para su edición y visualización.

- Agregar Usuario: Muestra un formulario auxiliar con controles similares al formulario OnLine, que nos permite ingresar la información de un usuario para posteriormente ser visualizado en el control ListView.

- Eliminar Usuario: Permite eliminar el usuario que se ha seleccionado en el control ListView.

- Editar Usuario: Permite editar un usuario seleccionado en el control ListView, mostrando el formulario auxiliar de la función “Agregar usuario” pero con la información del usuario ha ser modificado cargado en sus controles.

- Guardar: Permite guardar los cambios realizados en el archivo abierto o en su defecto permite guardar uno nuevo para su posterior escritura en el controlador.

4.3.4.3 Formulario Consulta Base

Este formulario es el que nos permite realizar consultas en los archivos de registros de ingreso descargados del controlador. El elemento principal de este formulario es un control RichTextBox, en donde se muestran los resultados de las

consultas, agregando el formato adecuado al texto para una mejor visualización. Las funciones que agrupa este formulario son:

- Abrir: Abre un archivo de registros de ingreso al sistema COD, y habilita los campos de consulta una vez que se ha leído toda la información del archivo.

- Consultar: Permite realizar una búsqueda personalizada en donde se puede especificar la identificación magnética del usuario, fecha particular o rango de fechas, módulo wiegand de ingreso, hora particular o rango de horas. Las búsquedas pueden combinar los diferentes campos de consulta.

- Exportar: Permite exportar el resultado de la consulta a un archivo en formato RTF (Rich Text Format), el cual es compatible con la mayoría de procesadores de texto como Microsoft Word.

CAPÍTULO 5

IMPLEMENTACIÓN

El sistema de control de acceso esta conformado por dos tipos de módulos, uno de control y cuatro de salidas para cada interfase.

Cada módulo se compone por una sola tarjeta electrónica, lo cual provee una característica modular al sistema, con la capacidad de expansión específicamente para los módulos de salida. Todo esto hace que se pueda detallar completamente al sistema al conocer a fondo la implementación de cada uno de los módulos.

A continuación se detallará la implementación de hardware que incluye diagrama de conexiones, diseño de placas de circuito impreso, requerimientos de instalación y distribución de software, junto con los elementos adicionales que se necesitan para implementar el sistema en un edificio.

5.1 HARDWARE

La implementación de hardware se detalla en dos partes principales que constituyen el módulo principal y los módulos de expansión de salidas.

5.1.1 Implementación del Módulo de Control.

La **Figura. 5.1** muestra un diagrama esquemático total del módulo de control, donde se puede apreciar las secciones de entrada/salida de cada interfase, junto con los conectores pertenecientes al controlador principal RCM 3700 y al LCD, lectores wiegand 1 y 2, puertos de alimentación, batería de backup y puerto para expansión de módulos de salida 1 y 2.

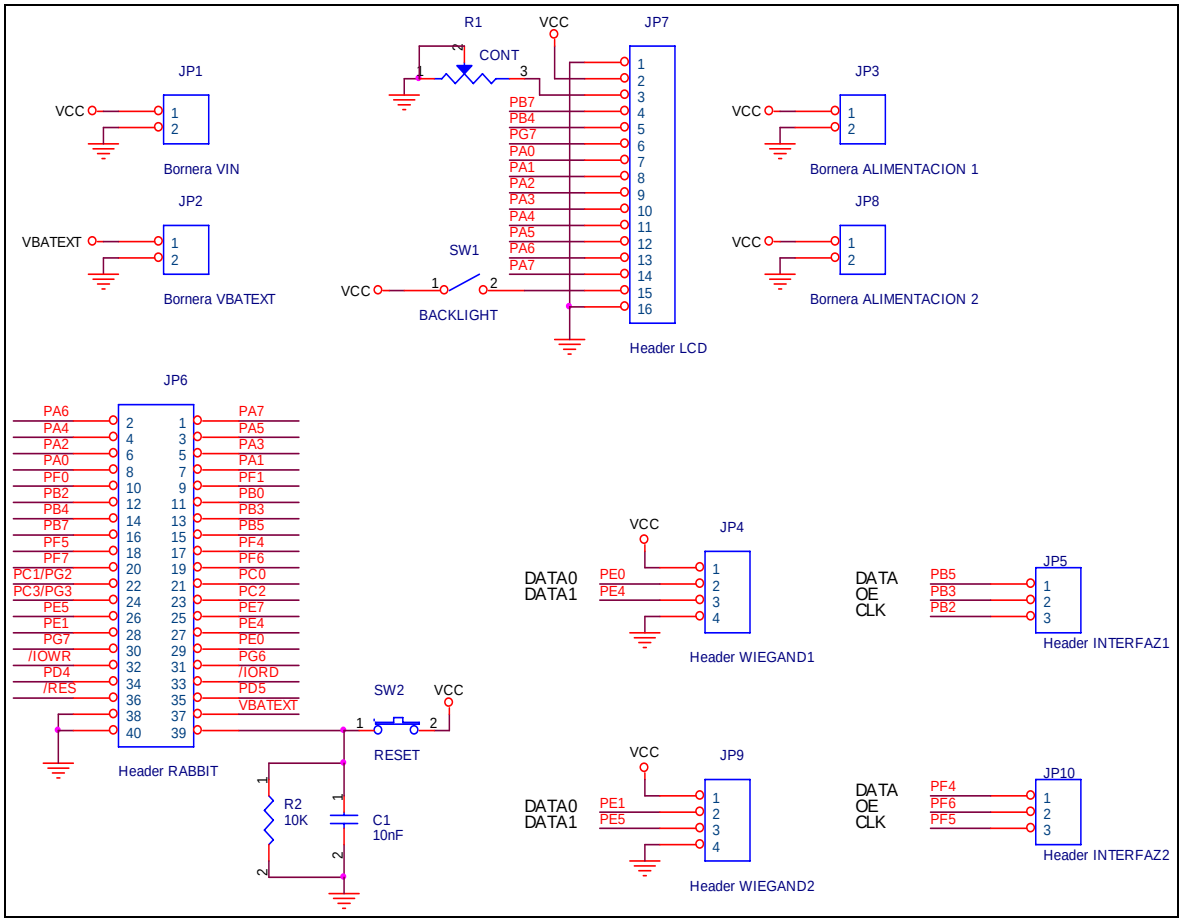


Figura. 5.1. Diagrama de conexión de la placa del módulo principal

La lista de materiales utilizados en esta placa se encuentra detallada en la **Tabla. 5.1.** Lista de materiales del módulo de control, sin embargo es necesario acotar que a dicha lista se debe agregar el controlador principal, dos lectores

wiegand de proximidad y el LCD, pues estos son conectados a la tarjeta mediante cable plano que va directo a los headers de conexión.

La **Figura. 5.2** y **Figura. 5.3** muestran las capas superior e inferior del PCB del módulo de control respectivamente. Mientras en la **Figura. 5.4** se indica en detalle la posición de cada uno de los elementos que conforman parte de ella.

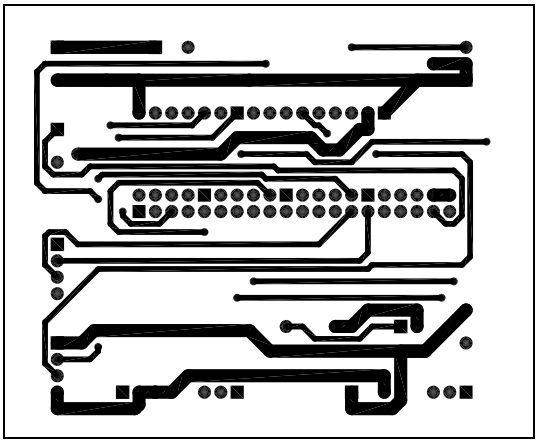


Figura. 5.2. Capa superior de la placa del módulo de control

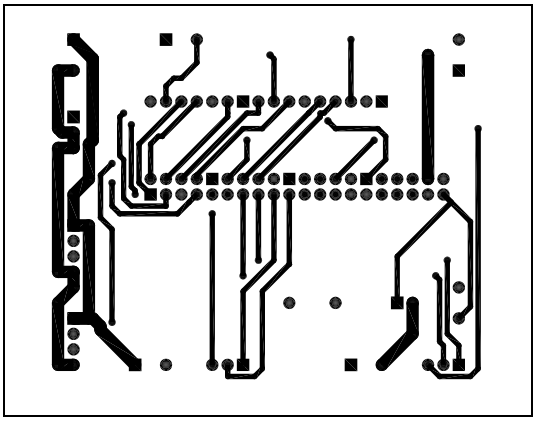


Figura. 5.3. Capa inferior de la placa del módulo de control

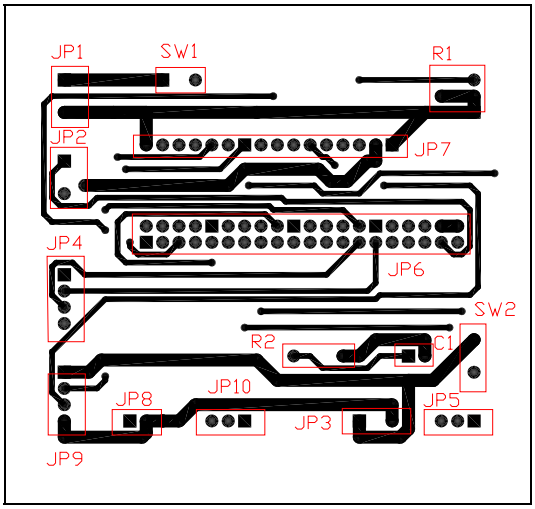


Figura. 5.4. Posición de los elementos en la placa del módulo de control

Tabla. 5.1. Lista de materiales del módulo de control

No.	Cantidad	Descripción/ Valor	Referencia de la Parte
1	1	Capacitor 10nF	C1
2	1	Bornera	JP1
3	1	Bornera	JP2
4	1	Bornera	JP3
5	1	Header 1x4	JP4
6	1	Header 1x3	JP5
7	1	Header 2x20	JP6
8	1	Header 1x15	JP7
9	1	Bornera	JP8
10	1	Header 1x4	JP9
11	1	Header 1x3	JP10
12	1	Potenciometo 5K	R1
13	1	10K	R2
14	1	Switch 2 posiciones	SW1
15	1	Pulsador NC	SW2

5.1.2 Implementación de Módulo de Salida.

El módulo de salida cuenta con un puerto de entrada y otro de salida con las señales DATA, OE y CLK, las cuales controlan los datos seriales mediante el registro de desplazamiento 74299 que permite conectar en cascada hasta un máximo de 4 módulos de expansión. Los módulos de expansión además incluyen drivers para los

relés, leds indicadores, puertos de alimentación y elementos adicionales que se detallan en la

Figura. 5.5.

La lista de materiales utilizados en esta placa se encuentra detallada en la **Tabla. 5.2**, Mientras que en la **Figura. 5.6** y **Figura. 5.7** muestran las capas superior e inferior del PCB del modulo respectivamente. Finalmente en la **Figura. 5.4** se indica en detalle la posición de cada uno de los elementos que la conforman.

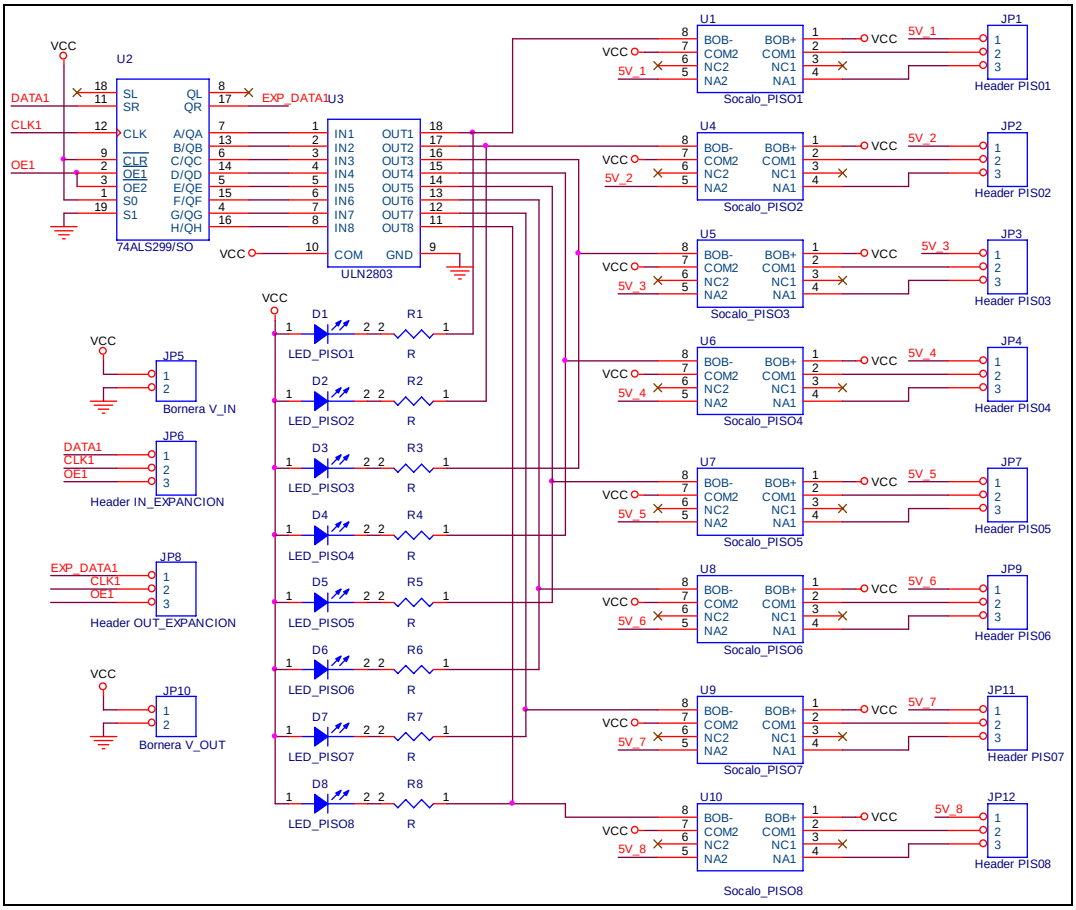


Figura. 5.5. Diagrama de conexión la la placa del módulo de expansión

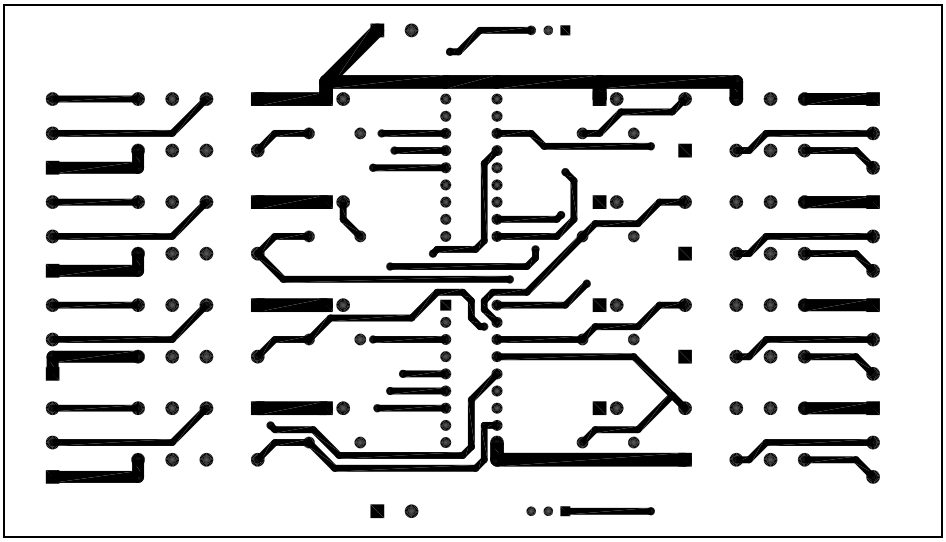


Figura. 5.6. Capa superior de la placa del módulo de salida

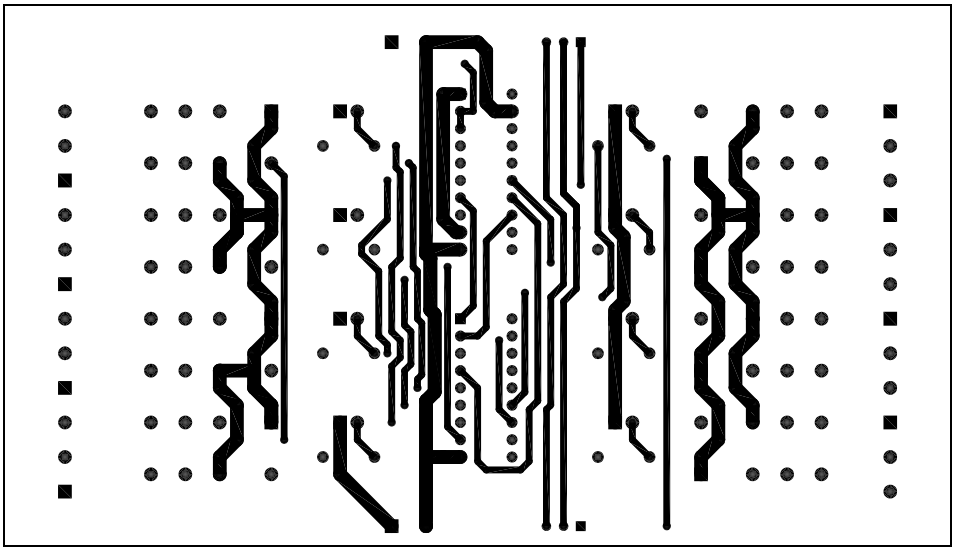


Figura. 5.7. Capa inferior de la placa del módulo de salida

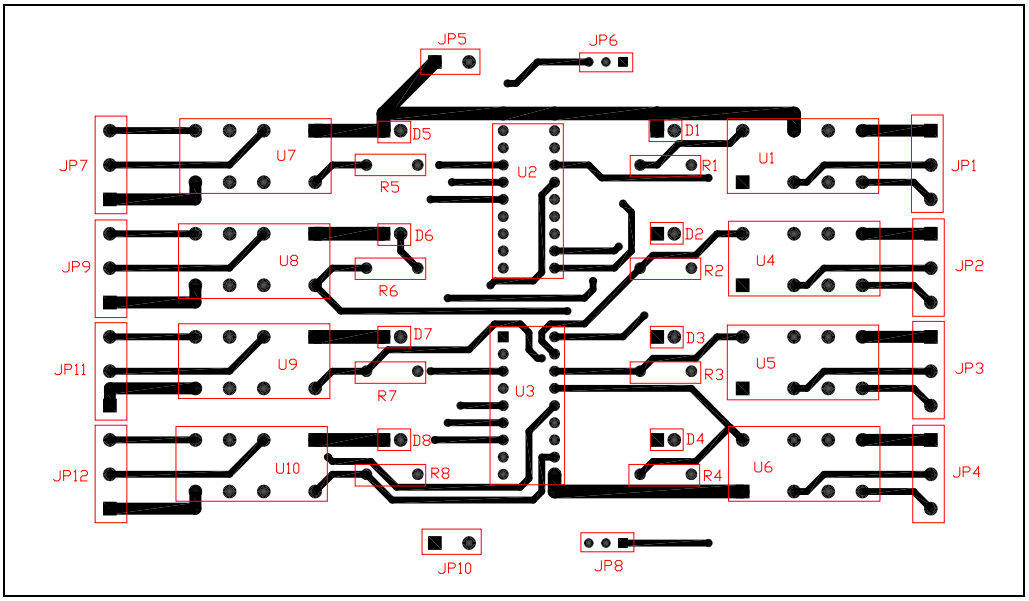


Figura. 5.8. Posición de los elementos en la placa del módulo de salida.

Tabla. 5.2. Lista de materiales del módulo de salida

No.	Cantidad	Descripción/Valor	Referencia de la Parte
1	1	LED	D1
2	1	LED	D2
3	1	LED	D3
4	1	LED	D4
5	1	LED	D5
6	1	LED	D6
7	1	LED	D7
8	1	LED	D8
9	1	Bornera	JP1
10	1	Bornera	JP2
11	1	Bornera	JP3
12	1	Bornera	JP4
13	1	Bornera	JP5
14	1	Header 1x3	JP6
15	1	Bornera	JP7
16	1	Bornera	JP8
17	1	Header 1x3	JP9
18	1	Bornera	JP10
19	1	Bornera	JP11
20	1	Bornera	JP12
21	1	Resistencia 1K	R1

22	1	Resistencia 1K	R2
23	1	Resistencia 1K	R3
24	1	Resistencia 1K	R4
25	1	Resistencia 1K	R5
26	1	Resistencia 1K	R6
27	1	Resistencia 1K	R7
28	1	Resistencia 1K	R8
29	1	Relé 5Vdc	U1
30	1	74ALS299	U2
31	1	ULN2803	U3
32	1	Relé 5Vdc	U4
33	1	Relé 5Vdc	U5
34	1	Relé 5Vdc	U6
35	1	Relé 5Vdc	U7
36	1	Relé 5Vdc	U8
37	1	Relé 5Vdc	U9
38	1	Relé 5Vdc	U10

5.2 SOFTWARE

La implementación de software en el ambiente integrado de desarrollo Visual Basic comprende: los requerimientos de software, requerimientos de hardware de computador necesarios para ejecutar la aplicación y la publicación de la misma.

5.2.1 Requerimientos de hardware de computador

Los requerimientos de hardware necesarios para la conexión con el módulo controlador y la ejecución correcta de la aplicación son los siguientes:

- Procesador INTEL Pentium III, AMD Athlon Xp, o superior
- 256 Mb de memoria RAM
- Al menos 100 Mb de espacio libre en el disco duro
- Tarjeta de red 10/100 base T, con conector RJ45
- Cable Ethernet cruzado categoría 5, con conectores RJ45

5.2.2 Requerimientos de software de computador.

Los requerimientos de software son los siguientes:

- Sistema operativo Windows 98 SE o superior
- Microsoft .NET Framework 2.0
- Windows Installer 3.1

5.2.3 Publicación.

La aplicación desarrollada en Visual Basic 2005 Express ha sido publicada mediante la herramienta incluida en el IDE (ambiente integrado de desarrollo) para este propósito. Los programas de instalación desarrollados en esta nueva versión de Visual Basic se diferencian de sus predecesores, en que ya no incluyen todos los prerrequisitos de software necesarios para su ejecución, en su lugar el programa de instalación ejecuta descargas automáticas desde el sitio web del vendedor del componente. La ventaja de este nuevo instalador basado en el .NET Framework 2.0 es que ya no registra componentes, permitiendo la convivencia de librerías o DLL con un mismo nombre pero de diferente versión sin ningún conflicto, esto permite actualizar las aplicaciones sin perder la funcionalidad de las versiones anteriores.

Las claves de acceso establecidas por defecto luego de la instalación se detallan en la **Tabla. 5.3**.

Tabla. 5.3. Usuarios y claves de acceso

	Acceso general	Herramienta de descarga de usuarios
Usuario	usuario	administrador
Password	coheco	coheco

CAPÍTULO 6

PRUEBAS Y RESULTADOS

6.1 PRUEBAS Y RESULTADOS ELÉCTRICOS

Las pruebas eléctricas fueron realizadas con diferentes configuraciones del sistema, esto se debe a las características modulares con las que cuenta el diseño, por lo tanto cada implementación tendrá un comportamiento propio que puede ser determinado mediante la información que se detalla a continuación.

6.1.1 Módulo de Control

Esta prueba consiste en energizar y medir el consumo de energía del módulo de control, los resultados se detallan en la

Tabla. 6.1**.** Los datos de voltaje y corriente obtenidos fueron tomados directamente de las borneras de alimentación del módulo de control.

En la prueba se utilizo el módulo de control con todos sus componentes básicos conectados (LCD, controlador principal, lectores de tarjetas magnéticos), pero sin ningún módulo de salida.

Tabla. 6.1. Pruebas eléctricas y resultados del módulo de control

Pruebas	Resultados	
	Voltaje [V]	Corriente [mA]
Módulo energizado	5	138
Módulo energizado con luz de backlight	5	385
Módulo energizado, chip de ethernet activo.	5	203
Módulo energizado, chip de ethernet activo y recibiendo datos.	5	220
Módulo energizado, chip de ethernet activo y enviando datos.	5	206

Para activar el chip de ethernet es necesario conectar el módulo de control al PC mediante un cable UTP cruzado y la transmisión de datos se considera cualquier actividad de conexión en la interfaz de usuario como la descarga de registros de ingreso.

Los resultados demuestran que el módulo de control tiene una carga máxima el momento en que se utiliza la luz de backlight del LCD, además se aprecia que la diferencia de consumo de corriente cuando el módulo se encuentra conectado al PC (chip de ethernet activo) es aproximadamente el doble de la necesaria simplemente para energizar el módulo.

6.1.2 Módulo de Salida.

Las características eléctricas del módulo de salida se encuentran en la **Tabla. 6.** 2**.** En las pruebas realizadas se utilizó el módulo de control, la interfaz de usuario y un módulo de salidas. Para la activación de salidas es necesario grabar un usuario en el controlador que tenga acceso a todos los pisos representados por las salidas y en el horario adecuado. Las lecturas de corriente se toman de manera aislada en el módulo de salida luego del acceso al sistema y la activación de las salidas mediante el uso de la tarjeta magnética grabada.

Las lecturas de voltaje y corriente fueron tomadas en las borneras de alimentación de un solo módulo de salida conectado al módulo de control, el mismo que contaba con todos sus elementos completamente funcionales.

Tabla. 6. 2 Pruebas y resultados eléctricos del modulo de salida

Pruebas	Resultados	
	Voltaje [V]	Corriente [mA]
Módulo energizado, sin salidas activadas.	5	36
Módulo energizado, 8 salidas activadas.	5	190

6.1.3 Sistema Completo

Los resultados eléctricos que se muestran en la Tabla. 6.3 corresponden al sistema completo que se compone del módulo de control y ocho módulos de salidas. Esta es la condición de carga máxima que puede alcanzar el sistema, considerando incluso el uso de una tarjeta magnética maestra con acceso a todos los pisos (todas las salidas activas), backlight y actividad del chip de ethernet.

Tabla. 6.3. Pruebas eléctricas y resultados del sistema

Pruebas	Resultados	
	Voltaje [V]	Corriente [mA]
Módulos energizados, sin salidas activadas.	5	174
Módulo energizado, 64 salidas activadas chip de ethernet activo y luz de backlight.	5	2284

Los datos obtenidos en todas las pruebas permiten dimensionar el tipo de fuente que se necesitaría para alimentar a las diferentes configuraciones que puede llegar a tener el sistema.

La siguiente ecuación muestra la corriente máxima que puede llegar a necesitar el sistema, dependiendo de la cantidad de módulos de salida que este posea.

I_{max} = n*190+780 [mA] ; n → número de módulos de salida

6.2 PRUEBAS Y RESULTADOS DE SOFTWARE

Las pruebas y resultados de software se refrieren sobre todo a los tiempos de respuesta y funcionamiento del sistema en las condiciones máximas para las que fue diseñado.

La manipulación de las variables del sistema permite simular que el controlador se encuentra con la base de datos llena. Para este propósito, dentro del código del controlador luego de la inicialización de todas las variables se debe adicionar las siguientes líneas de código:

```
pg_libre.pag_base=4095;   //Apunta a la pagina final de la base de datos
actualizar_pg_libre();    //Actualiza en flash variables criticas
j_ibase=20;               //Numero de registros listos para grabarse en la
                          ultima pagina
```

Esto hace que el controlador inicie su funcionamiento con la memoria de la base de datos completamente llena, con lo cual si se lo conecta al PC, puede descargarse toda la información y medir el tiempo que demora en realizarse dicha operación, los resultados obtenidos se encuentran en la **Tabla. 6.4.**

Para las pruebas con los usuarios es necesario generar un archivo con 4011 registros para ser utilizado con la interfaz de usuario, con lo cual es posible simular las operaciones de escritura o lectura en el controlador trabajando a capacidad máxima. La generación del archivo se realizó mediante el siguiente código en visual Basic:

```
Private sub generar()

    Dim numar as int16  = Freefile()

    FileOpen(numar, ”C:\generado.COH” , OpenMode.Random)

    For n As Integer = 1 To 4011
      FilePut(numar, estructura)

    Next

    FileClose(num_ar)
End sub
```

Tabla. 6.4. Pruebas y resultados de software

Pruebas	Resultados
Capacidad máxima de usuarios en memoria	4011
Capacidad máxima de registros en memoria.	70440
Tiempo de inicialización del sistema	20 [s]
Grabación de 4011 usuarios	40 [s]
Lectura de 4011 usuarios	10[s]
Lectura de 70440 registros	155 [s]
Agregar el usuario 4011	1[s]
Modificar el usuario 4011	3[s]
Eliminar el usuario 4011	1[s]
Activar salidas del usuario 4011	< 1[s]

La inicialización del sistema representa una etapa muy importante en el funcionamiento del sistema, ya que en este tiempo el controlador reserva bloques de memoria RAM en donde se almacenará información crítica, que requiere tiempos de lectura mínimo.

Los tiempos relacionados con el usuario 4011 representan los retrasos máximos en la respuesta que tendrá el sistema trabajando a máxima capacidad. Por lo tanto si el usuario 4011 accede al sistema, las salidas que dan acceso al piso correspondiente se activarán en un tiempo inferior a 1 segundo, lo que representa un gran desempeño del control de acceso.

Los resultados obtenidos demuestran que la operación que demanda más tiempo para su ejecución es la lectura de los 70440 registros de ingreso (155 [s]), considerando que esta operación es parte del mantenimiento de rutina, no representa ninguna incomodidad para el usuario el cual podrá acceder al sistema luego de la lectura de registros e incluso durante la ejecución de esta, gracias al funcionamiento de tareas en paralelo en el controlador.

6.2.1 Condiciones de prueba

Las comunicación entre el controlador principal y el PC se realizó mediante un enlace punto a punto con cable cruzado, UTP categoría 5. El PC utilizado posee un procesador Mobile AMD Athlon XP-M 2200, con Windows XP SP2, memoria RAM de 512 MB y una tarjeta de red 10/100 base T.

La alimentación para todas las pruebas del sistema se realizó mediante una fuente tipo switching de 5 [V] y 10 [A].

CAPÍTULO 7

ANÁLISIS TÉCNICO COMPARATIVO

En el presente capítulo se describirá las pruebas realizadas al sistema y los resultados obtenidos, así como un análisis comparativo con la primera versión del sistema.

7.1 ANÁLISIS TÉCNICO COMPARATIVO CON LA PRIMERA VERSIÓN DEL SISTEMA

En base a las pruebas realizadas en este proyecto y a la información recolectada sobre la primera versión del sistema, se ha desarrollado un análisis comparativo que pretende demostrar las diferencias y ventajas funcionales que caracterizan a esta nueva versión.

La información recolectada se ha dividido en categorías con tablas comparativas, en las cuales se incluyen las aclaraciones necesarias.

7.1.1 Control de acceso.

La **Tabla. 7.1** resume las características de funcionamiento generales relacionadas con el control de acceso y usuarios.

Tabla. 7.1. Comparación de control de acceso

Característica	PSSM	PSSM duo
Número de usuarios	1024	4011
Número de lectores	1	2
Base de datos de Usuarios	Microsoft Acces	Sistema de archivos individuales
Control de acceso por horario	No soportado	Reloj de tiempo real, con acceso controlado las 24 horas y 7 días de la semana.
Número de salidas	Hasta 31 salidas	32 x 2 (32 salidas por lector)
Combinación de Salidas	Una sola salida activa a la vez	Hasta 2^{32} (4294967296) combinaciones posibles
Asignación de lector	No soportado	Individual y combinación de lectores 1 y 2

En la segunda versión del sistema, la cantidad de usuarios es compartida por los 2 lectores, es decir que los 4011 registros disponibles serán asignados a cada lector de acuerdo a las características del lugar donde sea implementado y al número de lectores instalados.

Los identificadores pueden ser tarjetas de deslizamiento o elementos de proximidad llamados "**tags**" cuya presentación varía en forma de tarjeta, llavero, moneda, etc. El lector mantiene relación con la tecnología utilizada en el identificador, cumpliendo con el formato Wiegand de 26 bits.

7.1.2 Registros de acceso al sistema.

Los registros de acceso al sistema constituyen una característica totalmente nueva que funciona en conjunto con el control de acceso, el reloj de tiempo real y la

interfaz de usuario, que permite consulta con presentación de reportes, éstas características se detallan en la **Tabla. 7.2**.

Tabla. 7.2. Registros de acceso al sistema

Característica	PSSM	PSSM duo
Número de registros	No soportado	Hasta 70440 registros
Consulta de registros		Soportado mediante archivos y exportación en formato RTF.

7.1.3 Comunicación y mantenimiento.

En la **Tabla. 7.3** se encuentran las características que dan soporte a la comunicación entre el módulo controlador y el usuario. Además se incluyen los detalles útiles para el mantenimiento del sistema y administración por parte del servicio técnico.

Tabla. 7.3 Comunicación y mantenimiento

Comunicación	RS232	TCP
Capacidad de red	No soportada	Ethernet 10/100 base T
Interfaz de administración	Mediante PC y teclado matricial con LCD	Mediante PC
Mensajes de soporte para mantenimiento	No soportado	Mediante LCD
Grabación de código Wiegand y factibilidad	No soportado, graba lectura directa	Grabación real por código sin necesidad de lectura

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

- El sistema de control de acceso cumple de manera satisfactoria con los objetivos y alcance propuesto para este proyecto, mostrando un mejor desempeño en varios sentidos respecto a la versión anterior del mismo.

- La implementación de un sistema modular y expandible, permite que el mantenimiento del sistema sea mas económico y eficiente pues no se necesita cambiar por completo el sistema si este sufre algún daño, basta con remplazar el modulo que tenga problemas.

- El manejo de interfases independientes con una memoria compartida provee al sistema de control de acceso de flexibilidad, pues puede ser aplicado a controlar dos ascensores de manera independiente.

- El uso de técnicas de programación en multitarea hace que el sistema operativo del controlador principal pueda realizar varias tareas de manera paralela.

- La utilización del RCM 3700 como controlador principal del sistema permitió que la mayoría de los dispositivos básicos que lo caracterizan se encuentren integrados en un solo dispositivo, de esta manera se obtuvo un diseño más robusto y eficiente.

- El reloj de tiempo real incluido en el módulo RCM 3700 permitió implementar los registros de acceso al sistema.

- La batería de backup es indispensable para respaldar la información contenida en el reloj de tiempo real y la memoria RAM del módulo RCM 3700.

- El uso de la memoria flash para almacenar datos y variables críticas del sistema hace que el sistema sea confiable en lo referente al manejo de información.

- El tiempo acceso a una memoria flash serial limita el desempeño del sistema, provocando retrasos cuando se accede a los últimos usuarios registrados.

- La reservación de un espacio de memoria RAM para almacenar la información de los usuarios al arrancar el controlador, permitió alcanzar velocidades de acceso inferiores a 1 segundo para el último usuario registrado.

- Una adecuada organización y optimización de los datos a almacenar, junto con algoritmos de búsqueda adecuados hacen que una base de datos se vuelva más eficiente.

- La conectividad mediante ethernet y TCP/IP permite al sistema acoplarse a las tecnologías actuales de redes, logrando incluso a comunicarse de manera remota.

- La implementación de un protocolo de comunicaciones propio entre el controlador principal y el PC, provee de seguridad al sistema pues evita que se pueda acceder a la información del controlador con métodos convencionales.

- Una ventaja de desarrollar aplicaciones en Visual Basic 2005 Express, es que estas son ejecutadas bajo la administración del CLR (Common

Language Runtime) del .NET framework, de esta manera una aplicación no puede salirse de control provocando fallos generales en el sistema.

RECOMENDACIONES

- Se debe planificar revisiones de rutina de la base de datos y del sistema en general de acuerdo al tráfico en el ascensor.

- Se debe tomar en cuenta las características eléctricas del sistema, para dimensionar la fuente de alimentación.

- Verificar los mensajes desplegados en el LCD, ya que estos proporcionan información útil sobre el estado de la base datos, hora, fecha, inicialización, etc.

- Se recomienda remplazar la batería de backup cada 3 años aún cuando no exista pérdidas de información.

- Con el fin de mejorar los tiempos de respuesta en el sistema es recomendable trabajar con una copia de la información que posee, en RAM. Pues la memoria flash posee como desventaja, tiempos de acceso demasiado grandes en comparación con los de la memoria RAM.

- El manejo de estructuras de datos que constantemente deban ser almacenados en memoria no volátil como la memoria flash, debe ser adecuadamente organizado, pues esta memoria posee ciclos de grabación limitados; por lo cual se debe procurar no escribir varias veces un mismo sector.

- Con el fin de dotar de la capacidad de realizar varias funciones al sistema, es recomendable utilizar técnicas de programación en multitarea.

- En el diseño de los circuitos impresos, se debe tomar en cuenta la cantidad de corriente que cada pista puede llegar a manejar, con el fin de dimensionar adecuadamente el grosor de estas, evitando así problemas de sobrecalentamiento y caída de voltaje en los circuitos.

REFERENCIAS BIBLIOGRÁFICAS

- Módulo RabbitCore RCM3700, User's Manual, Z-WOLRD.
- Microprocesador Rabbit 3000, User's Manual, Z-WORLD.
- Microprocesador Rabbit 3000, Designer's Handbook, Z-WORLD.
- TCP/IP, An Introduction to TCP/IP, Z-WORLD.
- Teoría de Redes, Semestre 1 de CCNA, Academia de Networking CISCO
- Formato de tarjetas magnéticas, Understanding Card Data Formats - Technology Basics White Paper HID, HID Corp.
- RODRÍGUEZ Patricia, *Diseño y construcción de un sistema de control de acceso para ascensores Mitsubishi utilizando sensores de proximidad*, ESPE, Proyecto de grado, Sangolquí - Ecuador, 2002.
- GRANIZO Evelio, *Programación Orientada a Objetos en C++ Teoría y Ejercicios*, Editorial ESPE, primera edición, Quito-Ecuador, 1997.
- CASTAÑEDA Juan José, SABANA Maribel, *Visual Basic 2005 Express Como debe de ser*, Editorial Megabyte, primera edición, Lima-Perú, 2005.

- HORROCKS Kris, CAMPBELL Sean, HATCHARD Derek, BERNHARDT Peter, SWIGART Scott, *Introducing Visual Basic 2005 for Developers*, primera edición, Microsoft, 2005.

- Bases de datos, http://es.wikipedia.org/wiki/Base_de_datos.html

- Programación orientada a objetos, http://www.investigacion.frc.utn.edu.ar/labsis/Publicaciones/apunte_linux/ma.html

- Microsoft MSDN 2005 Express Edition, http://www.msdn.com

ANEXO 1
Manual de Usuario

MANUAL DE OPERACIÓN DEL PROGRAMA

INTRODUCCIÓN

El presente manual pretende ser una guía completa para el uso del programa que acompaña al CONTROL DE ACCESO PARA ASCENSORES MITSUBISHI. Los temas incluidos en este manual son:

- Requerimientos del Sistema
- Instalación
- Uso de la aplicación

1. REQUERIMIENTOS DEL SISTEMA

Los requerimientos del sistema se refieren tanto al software como al hardware que será necesario para ejecutar la aplicación de manera adecuada aprovechando todas las características que incluye en su diseño.

1.1 REQUERIMIENTOS DE SOFTWARE

Los requerimientos de software para la instalación y ejecución de la aplicación son los siguientes

- Sistema operativo Windows 98 SE o superior
- Microsoft .NET Framework 2.0
- Windows Installer 3.1

1.2 REQUERIMIENTOS DE HARDWARE

Los requerimientos de hardware que incluyen los necesarios para la conexión con el módulo controlador son los siguientes:

- Procesador INTEL Pentium III, AMD Athlon Xp, o superior
- 256 Mb de memoria RAM
- Al menos 100 Mb de espacio libre en el disco duro
- Tarjeta de red 10/100 base T, con conector RJ45
- Cable Ethernet cruzado categoría 5, con conectores RJ45

2. INSTALACIÓN

Para instalar la aplicación se debe ejecutar el programa **SETUP**, que es el método de publicación desarrollado, este programa instalador verificará los requerimientos de software, en caso de no cumplir con estos, se procede a la descarga automática desde el sitio web del vendedor del paquete. Si ya se posee los paquetes de software, es aconsejable realizar la instalación de estos antes de ejecutar el **SETUP.**

El programa de instalación viene acompañado con el manifiesto de publicación, en donde se encuentra información sobre el autor y la versión del paquete. En la **Figura 1** se detalla el programa instalador y los archivos que acompañan a este.

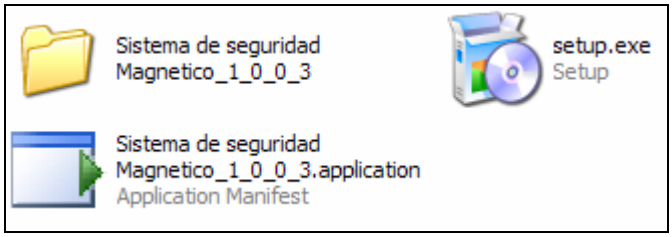


Figura 1. Setup y manifiesto de publicación

3. USO DE LA APLICACIÓN

3.1 AUTENTIFICACIÓN

El programa cuenta con un control de acceso protegido mediante nombre de usuario y contraseña, esta información debe ser ingresada en el diálogo que se muestra en la **Figura 2.**

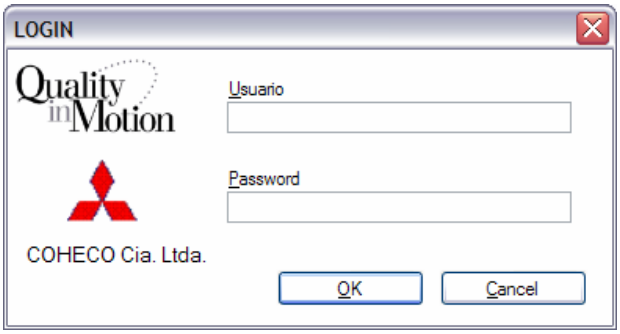


Figura 2. Diálogo de usuario y password

El nombre de usuario no distingue entre letras mayúsculas y minúsculas pero el password si, por lo tanto se debe tener cuidado en la manera en que se ingresa la información en este campo.

3.2 FORMULARIO ONLINE

El Formulario **OnLine** incluye todas las herramientas para trabajar con el módulo de control conectado al PC, si el módulo no se encuentra conectado solo se puede acceder a las herramientas de los formularios OffLine y Consulta Base que se detallaran posteriormente. Las partes principales de este formulario se indican en la Figura 3.

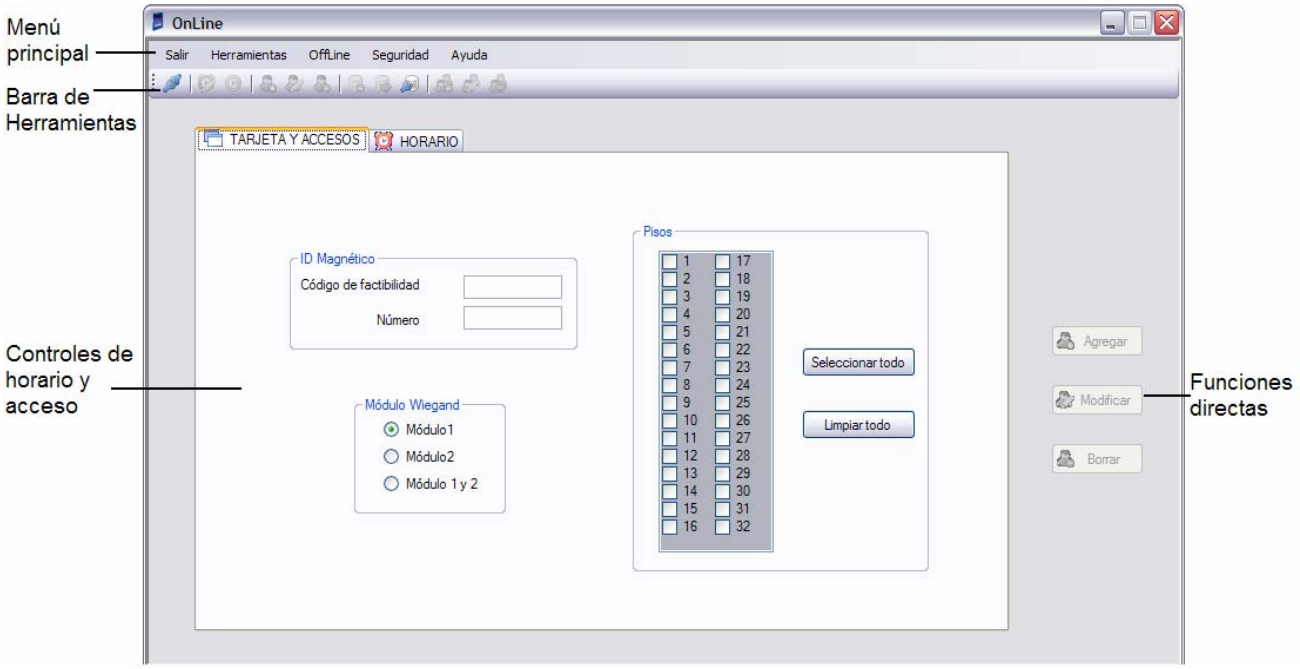


Figura 3. Formulario OnLine

Todas las funciones que incluye este formulario y que se describen a continuación pueden ser ejecutadas desde el menú principal, la barra de herramientas y en algunos casos desde botones de función directa.

3.2.1 Conectar

Esta función despliega el diálogo de la **Figura 4** en donde se nos permite definir la dirección IP y puerto con los que nos conectaremos con el controlador principal. Los valores por defecto se mostrarán siempre al iniciar el diálogo, y pueden ser modificados de acuerdo a la configuración de red programada en el controlador principal.

Figura 4. Diálogo de conexión

En caso de no tener una red disponible o no existir en la red un controlador con la dirección IP indicada, el programa mostrará el mensaje de error correspondiente y no habilitará ninguna de las otras funciones del formulario OnLine.

3.2.2 Actualizar Hora y Fecha

Iguala la hora y fecha del controlador principal con la hora y fecha del PC, mostrando un mensaje que indica el éxito de la operación.

3.2.3 Modificar tiempo de activación

Esta función permite modificar el tiempo que las interfases de salida del controlador principal permanecen activas luego de que el ingreso de un usuario sea verificado. Este tiempo puede tener un valor entre 1 y 99 segundos, y es independiente para cada módulo wiegand, por lo tanto es posible modificar el tiempo de activación de manera independiente o en conjunto haciendo uso de las casillas de verificación que se muestran en la **Figura 5.**

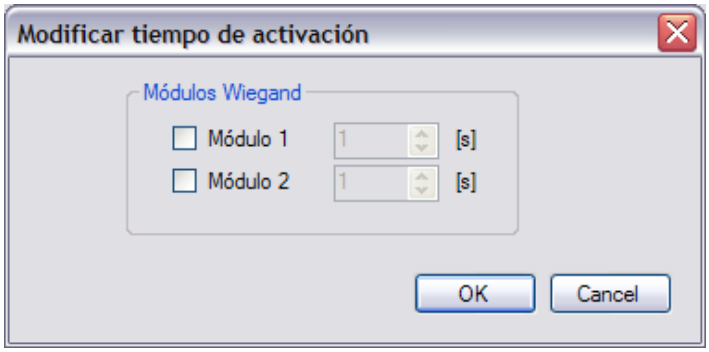


Figura 5. Modificar tiempo de activación

3.2.4 Grabar usuarios en el controlador

Esta función despliega el diálogo de la figura 6, que permite abrir un archivo de extensión **COH** para ser grabado en el controlador principal. Los archivos **COH**[4] contienen información de horario y acceso de los usuarios, estos archivos pueden ser descargados desde el controlador o pueden ser creados mediante las funciones del formulario **OffLine.**

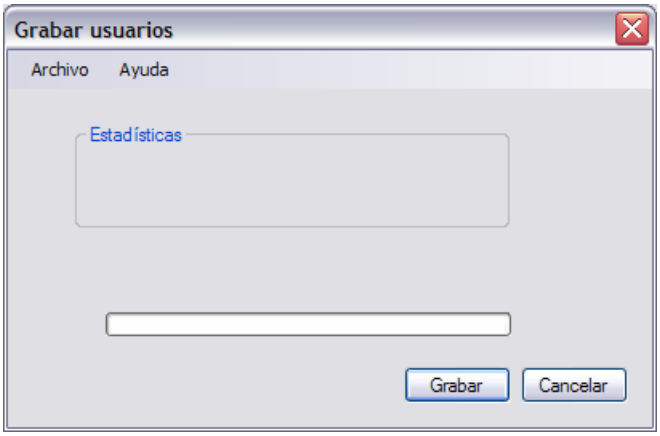


Figura 6. Grabar usuarios en el controlador

Para grabar un archivo en el controlador ingrese al menú archivo, elija abrir, escoja el archivo de extensión **COH** a ser grabado y acepte. En estadística se mostrará el número de usuarios que existen en el archivo, presione grabar y verifique

[4] COD y COH: son extensiones usadas para identificar archivos de base datos y usuarios respectivamente. Estas extensiones al no ser de uso restringido pueden ser usadas por otros programas sin tener ninguna relación con el PSSM duo.

el progreso de la operación. El tiempo de grabación depende del número de usuarios en el archivo y del tráfico de la red donde se encuentra instalado el sistema.

3.2.5 Leer usuarios del controlador

Esta función se encuentra protegida por un segundo sistema de autentificación como el indicado en la sección 0.

La **Figura 7** muestra el diálogo de descarga, el cual informa el número de usuarios que existen en el controlador. Para leer los usuarios presione aceptar, se observará el progreso de la descarga, al finalizar se abrirá un diálogo para guardar el archivo. La extensión por defecto es COH y no podrá ser modificada.

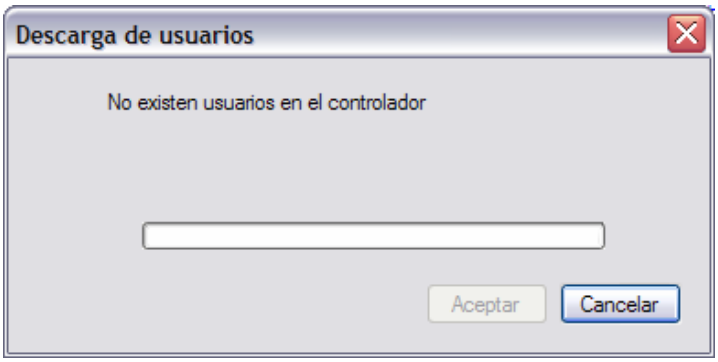


Figura 7. Leer usuarios del controlador

3.2.6 Borrar usuarios del controlador

Esta función elimina todos los usuarios del controlador. No existe manera de recuperar la información borrada, por lo tanto se debe tener cuidado al usar esta función.

3.2.7 Descargar base de datos del controlador

La descarga de base datos se refiere a la lectura de los registros de ingreso al sistema por parte de los usuarios. Esta función utiliza el diálogo que se muestra en la

Figura 8, en donde se tiene información del número de registros existentes y del progreso de lectura.

Para descargar damos clic en OK y se verifica el progreso de la operación, al terminar la lectura se abrirá un diálogo para guardar el archivo de extensión **COD**, se debe elegir un nombre adecuado y se acepta.

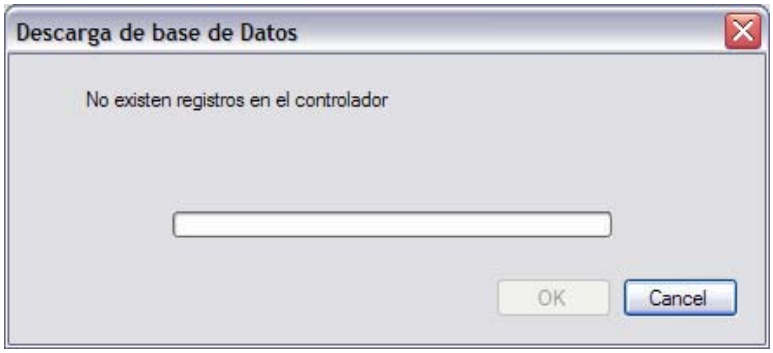


Figura 8. Descarga de base de datos del controlador

3.2.8 Borrar base de datos del controlador

Esta función elimina todos los registros de acceso al sistema, puede ser ejecutada en cualquier momento por el administrador del sistema y además se ejecuta automáticamente al finalizar la descarga de la base de datos del controlador.

3.2.9 Consultar base de datos

Esta es la única función del formulario OnLine que no requiere tener conectado el controlador principal a la PC.

El formulario de consulta incluye herramientas para determinar los criterios de búsqueda, campo para desplegar el resultado de la consulta y herramienta de exportación. Todos estos elementos solo funcionarán con un archivo de base de datos de extensión **COD** que sea previamente descargado del controlador y almacenado en el PC.

Los pasos para realizar una consulta sin criterios de búsqueda o simplemente mostrar todos los registros, son los siguientes:

1. Ingresar al formulario de consulta de base a través de la barra de herramientas o el menú principal del formulario OnLine.
2. Abrir un archivo de extensión COD mediante las herramientas del menú principal del formulario de Consulta de base de Datos. (La carga del archivo tardará entre 15 y 30 segundos dependiendo del número de registros)
3. Desactivar todas las casillas de los campos de consulta.
4. Dar un clic en el botón consultar.

Los pasos para realizar una consulta utilizando criterios de búsqueda o personalizada son los siguientes:

1. Repetir los pasos 1 y 2 descritos en la consulta sin criterios de búsqueda.
2. Activar las casillas de los campos de consulta de acuerdo a los requerimientos de búsqueda.
3. Dar un clic en el botón consultar.

La exportación de una consulta solo será posible si esta tiene como resultado al menos un elemento, para lo cual luego de la consulta se debe dar un clic en exportar. Se presentará un diálogo para guardar el archivo en formato RTF (Rich Text Format) en donde debemos especificar el nombre y ubicación donde deseamos almacenarlo. Los archivos RTF pueden ser visualizados y editados con cualquier procesador de texto como Microsoft Word.

Un ejemplo de cómo realizar una consulta personalizada se muestra en la Figura 9, en donde los resultados corresponderán a los registros de ingreso del usuario 12-

12343, comprendidos entre el 17 de Noviembre de 2006 y el 28 de Noviembre de 2006, y que hayan ocurrido a las 5 AM.

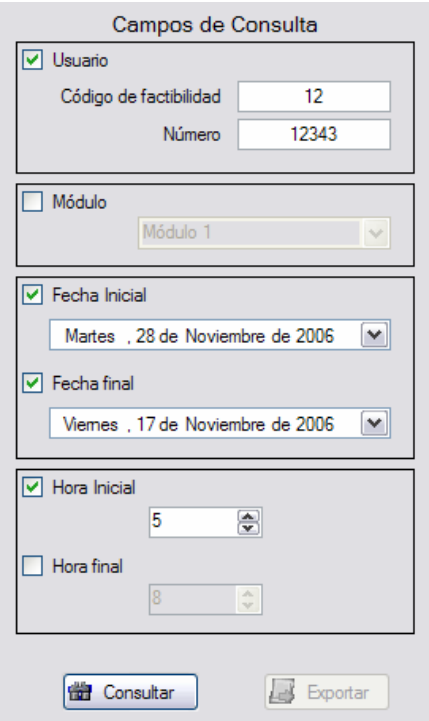


Figura 9. Consulta personalizada

La aplicación soporta cualquier combinación entre los diferentes campos de consulta, además valida el ingreso de fechas y horas cuando se usa un rango inicial y final para estos campos.

3.2.10 Funciones directas

Las funciones directas del formulario OnLine están orientadas a la grabación, modificación y eliminación de un solo usuario en el controlador. El uso de estas funciones esta directamente relacionado con las dos secciones de los controles de horarios y acceso cuyo uso se detalla a continuación.

La primera sección que se muestra en la Figura 10, se refiere al acceso a pisos, el ID magnético con su respectivo código de factibilidad y el módulo wiegand.

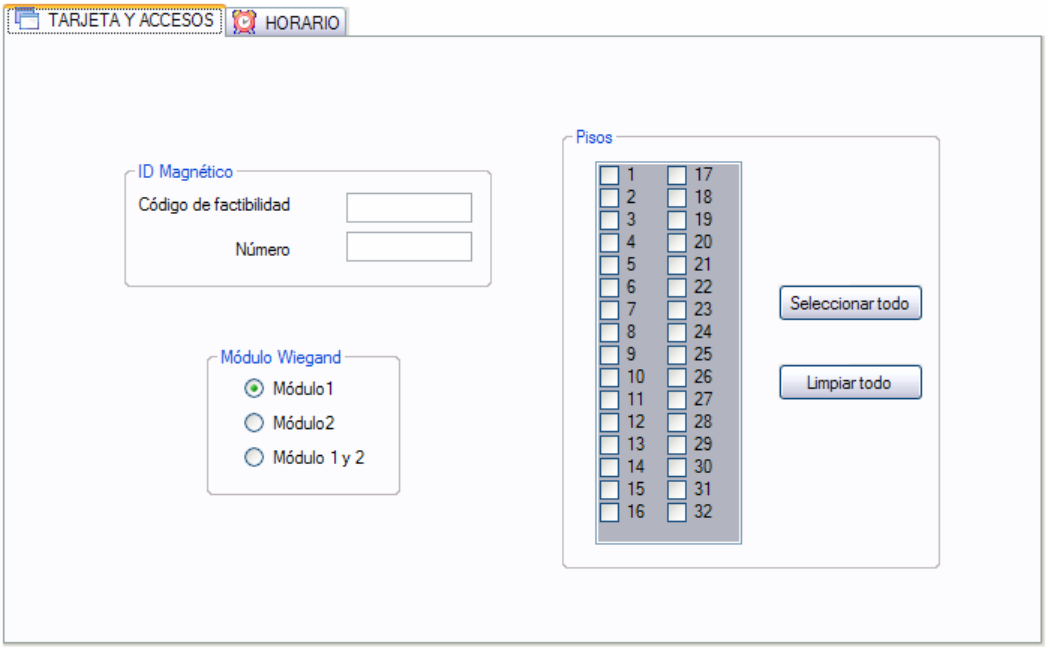


Figura 10. Controles de acceso

Para permitir al usuario, determinado por el ID magnético, acceder a un piso específico es necesario habilitar la correspondiente casilla de verificación en la sección Pisos. La selección puede ser múltiple o individual dando un total de 4294967296 combinaciones posibles por usuario. El módulo wiegand se refiere al lector específico en el cual el usuario tendrá acceso, siendo útil cuando el sistema se instala en edificios con zonas separadas como parqueaderos, sótanos, azoteas, etc.

La segunda sección que corresponde a los controles de horario se detalla en la Figura 11, en donde se adicionan controles de selección múltiple, personalizada, individual y por días de la semana, de acuerdo a las necesidades del administrador del sistema.

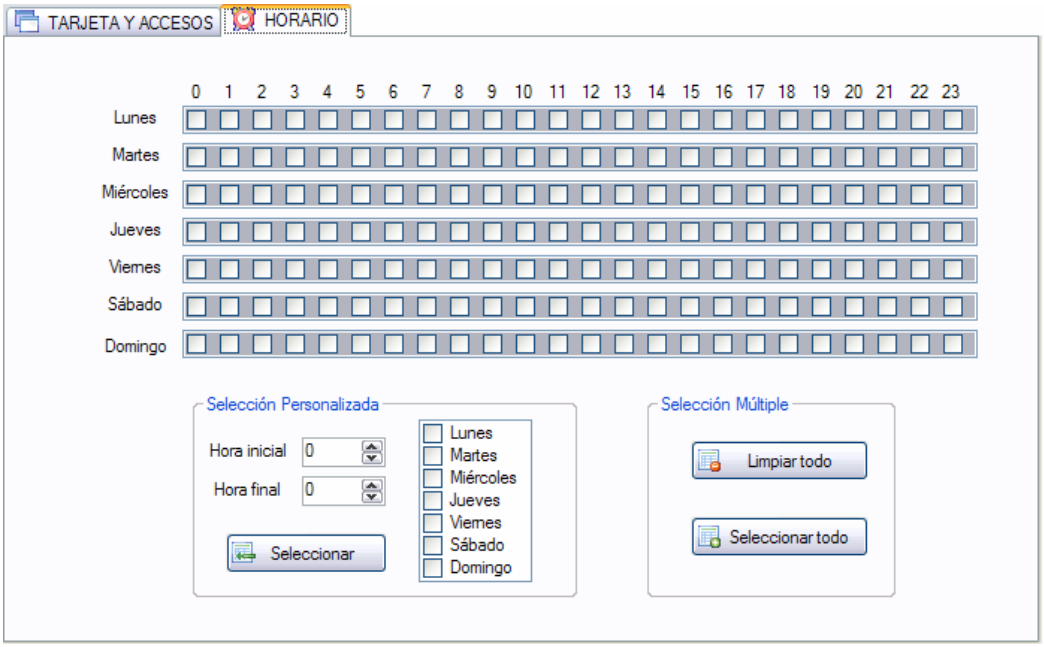


Figura 11. Controles de horario

Para realizar una selección individual se debe habilitar cualquiera de las casillas correspondientes a los siete días de la semana y al horario de 0 a 23 horas. La selección personalizada permite establecer días específicos y un horario similar útil cunado el usuario en cuestión tiene por ejemplo un horario de oficina de 8 AM a 16 PM de lunes a viernes. La selección múltiple permite generar tarjetas maestras con acceso a todos los pisos y en todos los horarios.

Una vez ingresada la información necesaria en los controles de acceso y horario, se procede a modificar, agregar o eliminar un nuevo usuario en el controlador mediante los botones de función directa de la Figura 12.

Figura 12. Botones de función directa

3.2.11 Funciones de seguridad

Se refiere a las opciones de cambio de usuario y password tanto para el acceso general como para el acceso a la función usada para leer usuarios del controlador. Para poder realizar cualquier cambio de información, es necesario conocer los usuarios y claves actuales, además de verificar que la nueva información ingresada para la clave sea correcta, tal como se detalla en el formulario de la Figura 13.

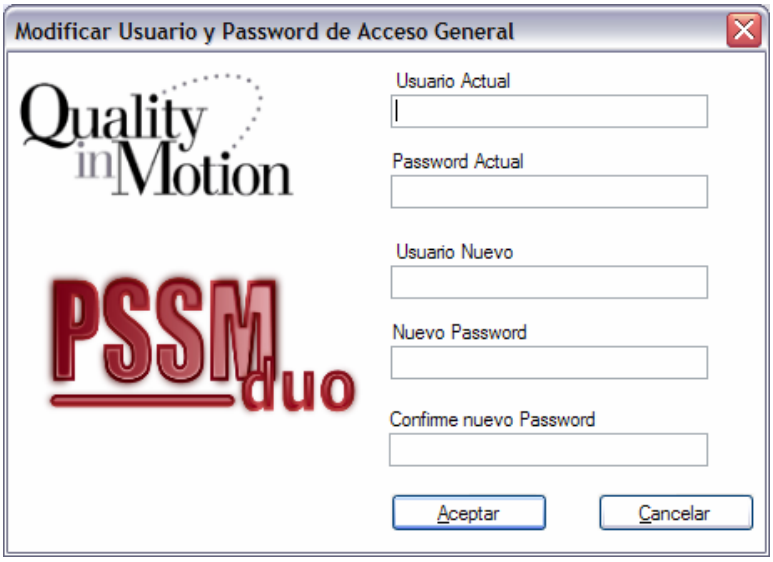


Figura 13. Formulario de cambio de usuario y password

3.3 FORMULARIO OFFLINE

El Formulario **OffLine** incluye todas las herramientas para trabajar en la ausencia del módulo de control principal. Su uso está especialmente orientado a la creación y edición de archivos de extensión **COH**, que contienen información de usuarios. La estructura y uso del formulario OffLine cumple las mismas reglas de las funciones directas del formulario **OnLine** descritas en la sección 0 con la única diferencia de que toda la información ingresada no se escribe directamente en el controlador principal sino que es almacenada en archivos de extensión **COH**.

El formulario **OffLine** es el que se muestra en la Figura 14, la lista de usuarios representa cada elemento que será agregado al archivo. El uso de la lista adiciona funciones de ordenamiento por ID, módulo, código Wiegand y código de pisos.

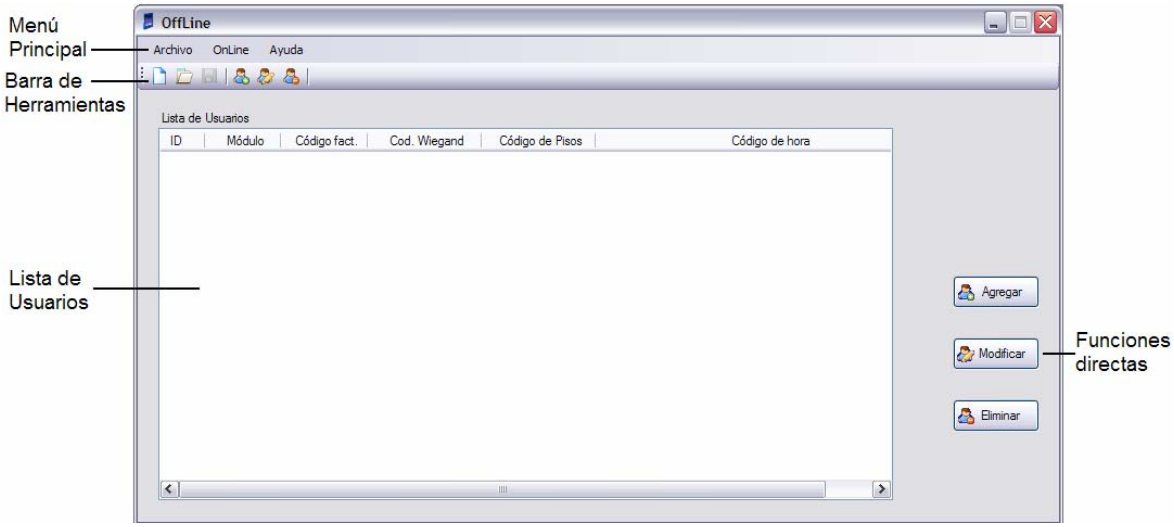


Figura 14. Formulario OffLine

Para editar un archivo es necesario abrirlo mediante las funciones del menú principal del formulario **OffLine** y la lista nos desplegará todos los usuarios que contenía el archivo**.** A continuación se debe seleccionar el elemento a modificar, damos clic en modificar de las funciones directas y se despliega el formulario con los controles de acceso y horarios cargados con la información del usuario, al cerrar se nos preguntará si deseamos guardar los cambios realizados.

Para agregar un nuevo usuario damos clic en agregar y se despliega el mismo formulario con los controles de acceso y horario vacíos listos para ingresar la nueva información.

Para crear un nuevo archivo se debe acceder a la función **nuevo** en el menú archivo, agregamos al menos un usuario con lo cual se habilita la función guardar la ejecutamos, se especifica el nombre y se acepta.

MANUAL DE OPERACIÓN
PSSM Duo

INTRODUCCIÓN

El sistema de control de acceso PSSM Duo, puede ser considerado como una interfase entre el controlador de un ascensor y los lectores de tarjetas que este posea, así su función es proveer o negar acceso, basando en criterios previamente definidos como horarios y pisos de acceso para cada usuario. El sistema esta conformado por dos tipos de módulos, que en total conforman un conjunto de 8 tarjetas electrónicas, el presente manual de usuario esta enfocado en la operación de dichas tarjetas, su instalación y detalles técnicos que ayudaran a reconocer posibles errores del sistema.

1. CONECTORES Y ELEMENTOS IMPORTANTES

Las tarjetas electrónicas pertenecientes a los módulos de control y de salidas poseen diferentes conectores y otros elementos que ayudan a su operación e interconexión.

2. MÓDULO DE CONTROL

Los elementos que forman parte del módulo de control se encuentran esquematizados en la

Figura 15, donde se observa su distribución dentro de la placa y las referencias para cada uno de ellos.

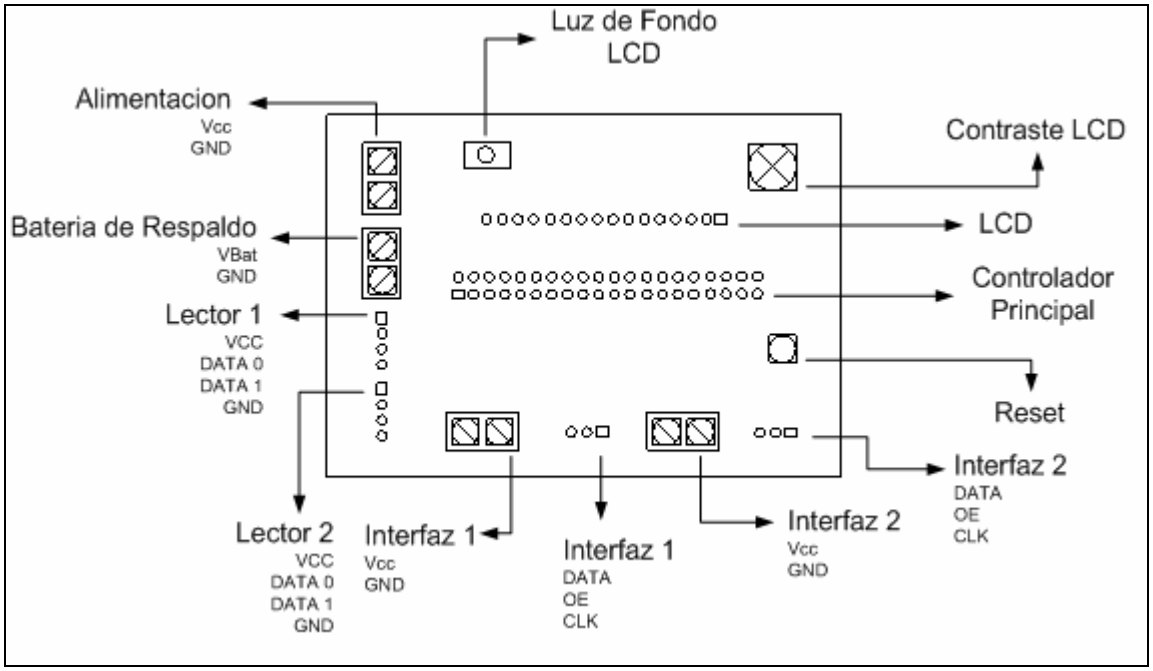


Figura 15. Módulo de control

2.1 ALIMENTACIÓN.

La bornera de alimentación recibe la fuente de poder que alimenta al sistema (5 Vdc).

2.2 BATERÍA DE RESPALDO.

En esta bornera se encuentra al entrada a la batería de respaldo para los datos del sistema, en caso de que esta no se encuentre conectada, pueden perderse registros de memoria que no hayan alcanzado a ser almacenadas en memoria no volátil, y el reloj de tiempo real que se maneja se desiguala cada vez que el sistema pierde energía.

2.3 LECTOR 1 y 2.

Los lectores de tarjetas se conectan mediante headers 1x4, cada pin se encuentra representado en la

Figura 15.

2.4 INTERFAZ 1 y 2.

Se conforma de dos conectores para cada interfase, la bornera transporta la energía para los módulos de salida, mientras que el header transporta las señales de datos y control que activan dichos módulos.

2.5 CONTRASTE LCD.

Este potenciómetro permite regular el contraste del LCD que posee el sistema.

2.6 LUZ DE FONDO LCD.

Es un switch de dos posiciones que permite habilitar o deshabilitar la luz de fondo del LCD cuando sea necesario.

2.7 LCD

En este header 1x16 se conecta el LCD del sistema.

2.8 CONTROLADOR PRINCIPAL

El controlador del sistema RCM 3700 se encuentra conectado a este header 2x20, en la Figura 16 se muestra dicho controlador, el cual posee un conector RJ-45, que permite la conexión des sistema PSSM Duo con un PC.

2.9 RESET

Este pulsador permite reiniciar el sistema en caso de producirse un mal funcionamiento en este.

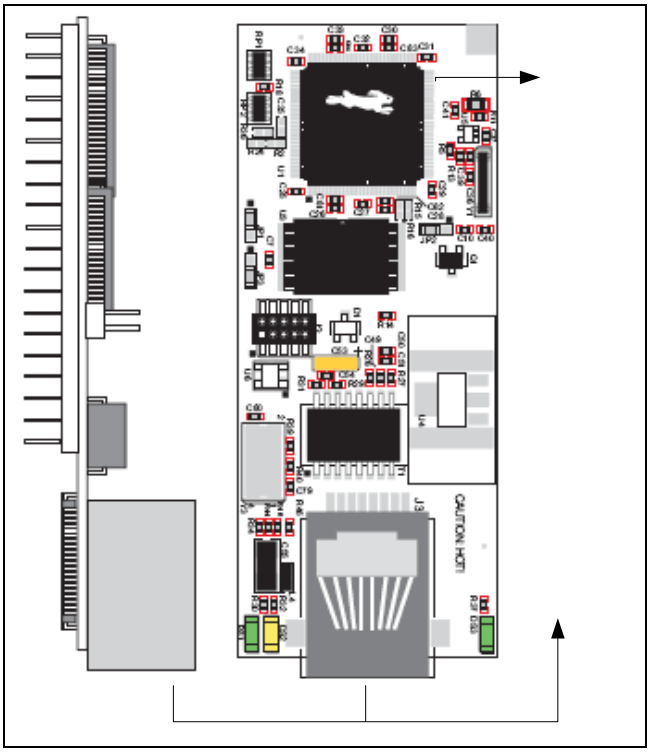


Figura 16. Controlador Principal RCM 3700

3. MÓDULO DE SALIDA

El módulo de salida se interactúa tanto con el controlador del ascensor como con el con el módulo de control y con otros módulos de salida con el fin de expandir su capacidad de salidas.

La

Figura 16, muestra la distribución de los elementos y conectores que forman parte del módulo de salida del sistema PSSM Duo.

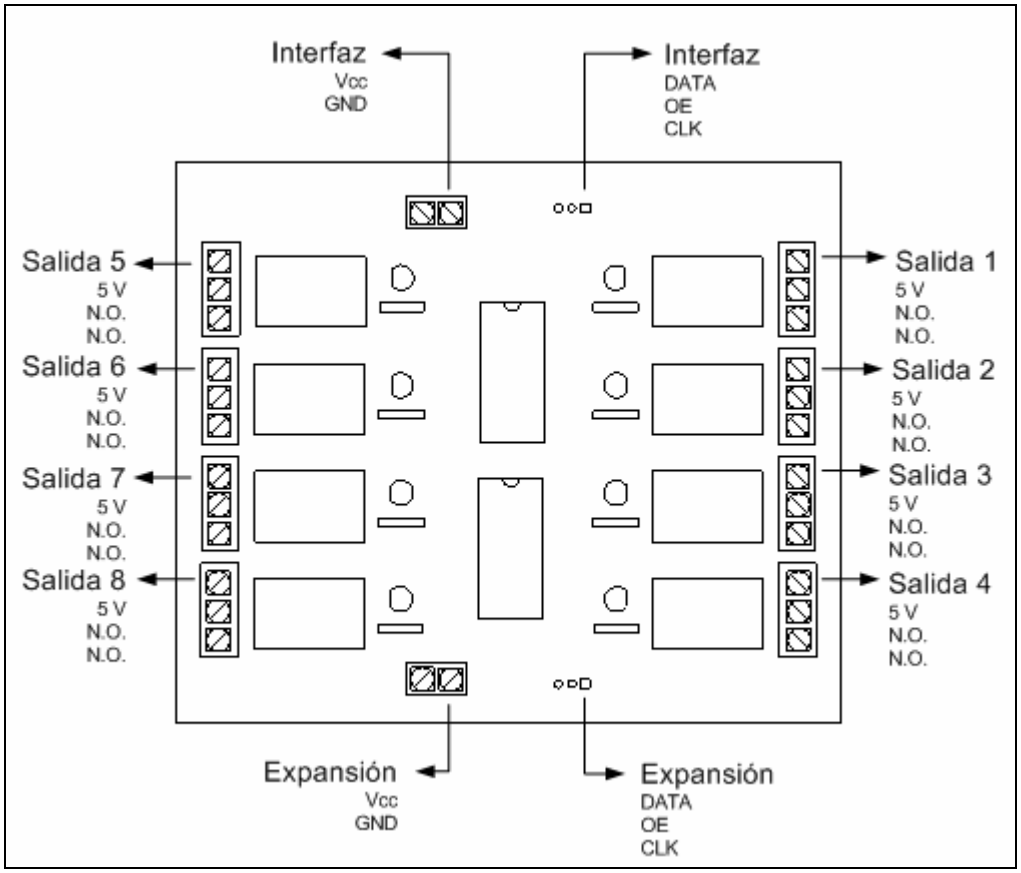


Figura 17. Módulo de Salida

3.1 INTERFAZ

Posee dos conectores, la bornera ingresa la energía al módulo, mientras que los headers reciben las señales de datos y control.

3.2 EXPANSIÓN

El puerto de expansión realiza la misma función que los puertos **INTERFAZ 1 y 2** del módulo de control, permite enviar las señales de control, datos y la energía para el siguiente módulo de salida, permitiendo así expandir el sistema.

3.3 SALIDA 1 – 8.

Son borneras 1x3, utilizadas para activar las interfases del controlador dentro del ascensor, tiene la flexibilidad de poseer una salida de 5 V y un contacto normalmente

abierto, permitiendo realizar deferentes tipos de conexiones dependiendo de las necesidades que sea presentadas.

4. CONEXIONES

Es sistema de control de acceso PSSM Duo, permite manejar hasta dos interfases independientes, con sus respectivos lectores y módulos de salida. Dado la modularidad y flexibilidad del sistema se puede conectar a cada uno de los módulos de manera distinta, dependiendo de las funciones que el sistema deberá realizar.

4.1 CONTROL DE ACCESO CON UNA SOLO INTERFAZ

La **Figura 18** muestra un diagrama de conexiones para que el sistema PSSM Duo opere con una sola interfaz. El módulo principal, debe ser alimentado con 5 V así como ser provisto de una batería de respaldo de 3.3 V, solo se utiliza un lector de tarjetas, teniendo en cuanta que en el header de conexión para el segundo lector se debe realizar un puente entre los pines Vcc, Data 0 y Data 1, con el fin de evitar falsas lecturas, finalmente solo se ocupa una interfaz de salida (la correspondiente al lector conectado).

Cabe acotar que la conexión de los módulos de salida con el controlador, es similar que la conexión para expansión entre módulos de salida.

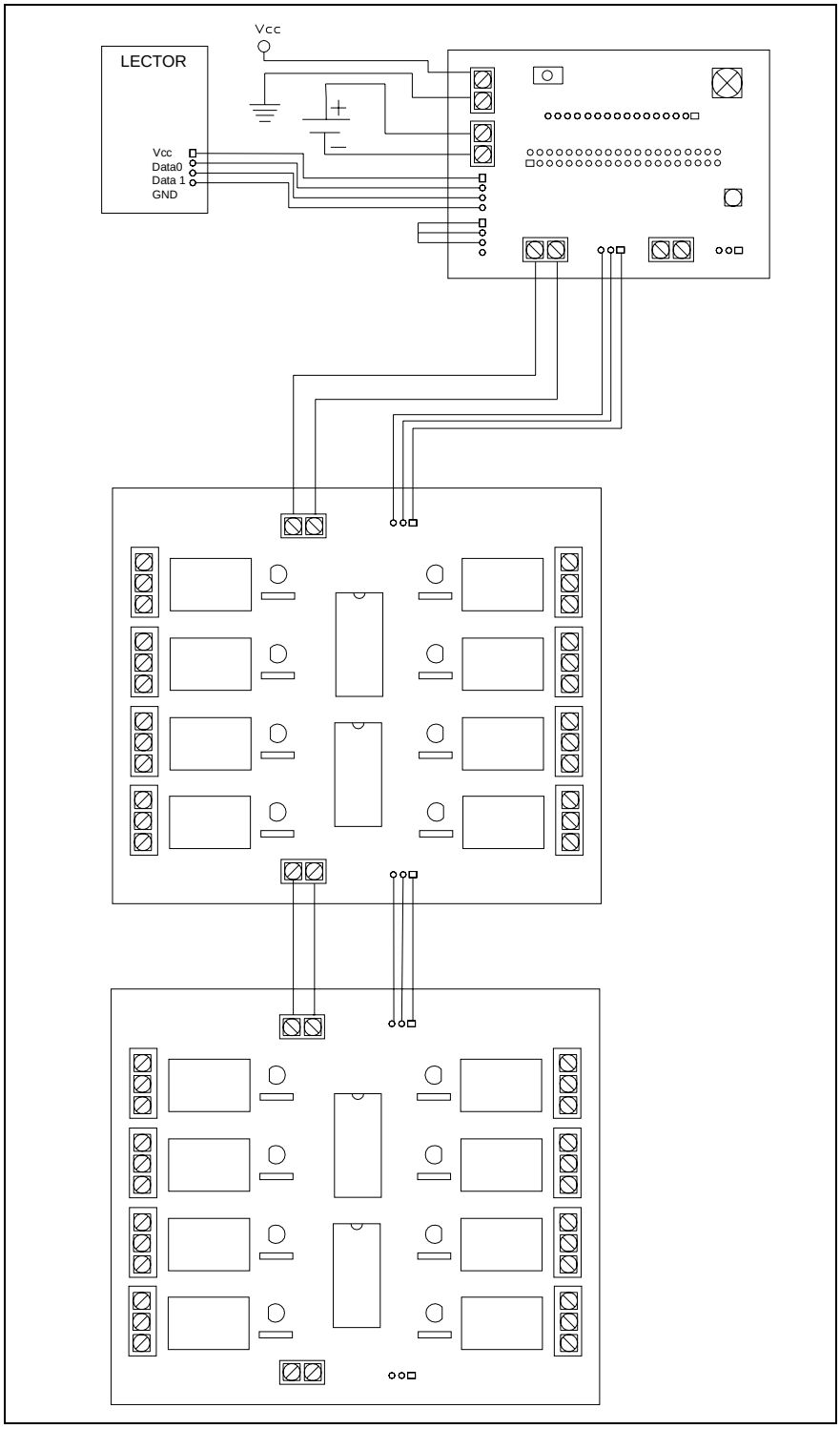


Figura 18. Control de acceso con una sola interfaz

4.2 CONTROL DE ACCESO CON DOS INTERFASES

En la

Figura 19, se encuentra un diagrama de conexiones para el sistema de control de acceso con dos interfases; como en el caso anterior la alimentación y la batería de respaldo para el módulo de control se conectan de la misma manera, pero esta vez se utilizan dos lectores y dos interfases de salida con conexiones similares al primer caso.

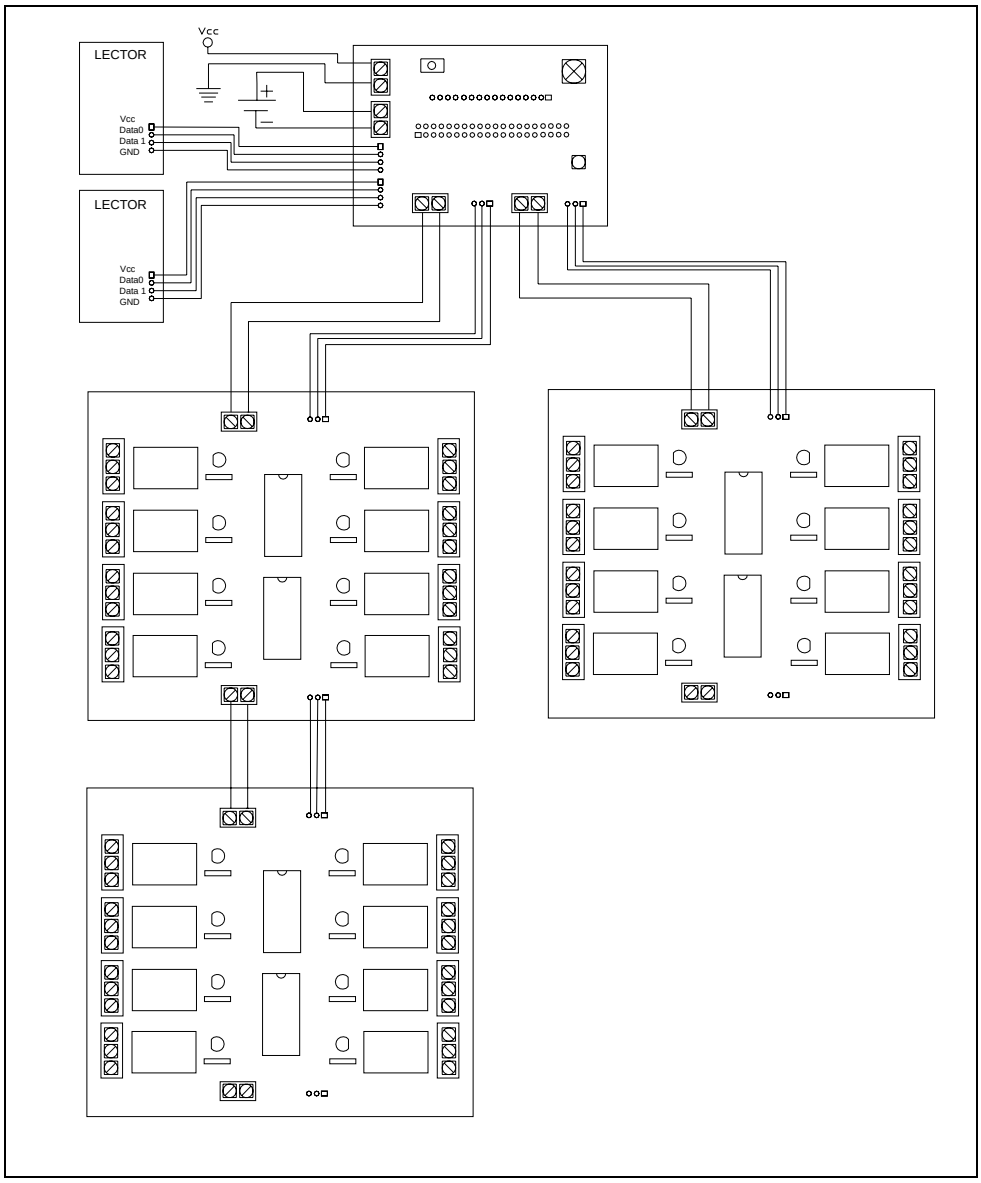


Figura 19. Control de acceso con dos interfases

5. MENSAJES DEL SISTEMA

El sistema de control de acceso PSSM Duo, presenta mensajes de funcionamiento, alertas y errores del mismo mediante el LCD del módulo de control, en la **Tabla 1**, se resumen y describen dichos mensajes.

Tabla 1. Mensajes del sistema

Mensaje	Descripción y Recomendaciones
INI. Error Wiegand 0	Error al inicializar la interfase para el lector 1, se recomienda revisar las conexiones o en último caso reemplazar el controlador.
INI. Error Wiegand 1	Error al inicializar la interfase para el lector 2, se recomienda revisar las conexiones o en último caso reemplazar el controlador.
INI. Error Flash Serial	Error al inicializar la memoria flash serial del controlador principal, el daño es irreparable y se debe cambiar el controlador principal.
Base de Datos Llena!	La base de datos de registros de acceso del controlador se encuentra llena y se sobrescribe, es recomendable descargar la base da datos al PC con el fin de no perder información antigua.
Usuario fuera de horario!	El controlador ha reconocido la existencia de la tarjeta presentada por el usuario, pero se encuentra fuera de su horario de acceso.
COHECO Cia. Ltda. Bienvenido…	El sistema ha reconocido, validado y permitido el acceso del usuario que ha presentado una tarjeta en un lector.
No pertenece A este lector!	La tarjeta presentada por el usuario existe en memoria y se encuentra dentro del horario de acceso, pero no pertenece al lector donde fue presentada.
Interfaz ocupada intente otra vez!	El sistema ha reconocido, validado y aceptado el acceso del usuario que ha presentado una tarjeta en un lector, sin embargo la interfaz de salida se encuentra ocupada y no se puede dar el acceso.
Tarjeta no registrada!	La tarjeta presentada en el lector no se encuentra registrada.
Fecha y hora actualizadas	El sistema se encuentra conectado a al PC y se ha actualizado la fecha y hora.
No hay espacio en memoria!	Se ha intentado almacenar un nuevo usuario cuando no existe espacio en la memoria del controlador.
Usuario ya	Se ha intentado grabar un usuario que ya existe en

esta en memoria!	memoria.
Usuario almacenado!	Se ha almacenado con éxito un nuevo usuario en el controlador.
Usuario no esta en memoria!	Se ha intentado borrar un usuario inexistente.
Usuario eliminado!	Se ha borrado un usuario de la memoria del controlador.
Base de datos ha sido eliminada	La base de registros de acceso ha sido eliminada del controlador.
Usuarios listo a enviarse	El controlador se encuentra listo para enviar toda la base de datos de los usuarios que posee en memoria.
Modificado tiempo de activación	Se ha modificado el tiempo de activación de las salidas de alguna interfaz.
Registros enviandose…	La base de datos de registros de acceso se está enviando al PC.
Recibiendo usuarios…	El controlador esta recibiendo un bloque de usuarios desde el PC.
PSSM DUO INICIALIZANDO…	El sistema esta arrancando, carga base de datos de usuarios en RAM, inicializa memoria flash e interfases de entrada salida.
COHECO Cia.Ldta. PSSM DUO	El sistema se ha inicializado correctamente y esta listo para operar.

ANEXO 2
Código del programa en Visual Basic

FORMULARIO ONLINE

```
'-------------------------------------------------------
'NAMSPACE, IMPORTACIONES
'-------------------------------------------------------
'Permite interoperabilidad con C# y visual C++
'Habilita opciones para declaracion de estructuras modificadas para que
'Funcionen como uniones
Imports System
Imports System.Runtime.InteropServices
Imports system.data

Public Class Online
    'Variables globales
    'VAriable auxiliar para la selección de días
    Dim auxdias() As Boolean = {True, True, True, True, True, True, True}
    'Variables de conexión TCP
    Public port As Int32
    Public server As String

    'Selecccciona o limpia de acuerdo a las opciones definidas en los controles
    'La matriz de horarios
    Private Sub Hseleccion_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Hseleccion.Click
        'valor de set o reset en la selección
        Dim aux As Boolean = True
        'validacion de hora
        If Hinicial.Value > Hfinal.Value Then
            MsgBox("La hora final no puede ser" & vbCrLf & "menor que la hora inicial", 16, "Error")
        Else
            'seleccion de horas
            For i As Integer = Hinicial.Value To Hfinal.Value
                If Hdias.GetItemChecked(0) Then
                    Hlunes.SetItemChecked(i, aux)
                End If
                If Hdias.GetItemChecked(1) Then
                    Hmartes.SetItemChecked(i, aux)
                End If
                If Hdias.GetItemChecked(2) Then
                    Hmiercoles.SetItemChecked(i, aux)
                End If
                If Hdias.GetItemChecked(3) Then
                    Hjueves.SetItemChecked(i, aux)
                End If
                If Hdias.GetItemChecked(4) Then
                    Hviernes.SetItemChecked(i, aux)
                End If
                If Hdias.GetItemChecked(5) Then
                    Hsabado.SetItemChecked(i, aux)
                End If
                If Hdias.GetItemChecked(6) Then
                    Hdomingo.SetItemChecked(i, aux)
```

```
            End If
        Next

    End If
End Sub

 'CODIGO DE LIMPIEZA DE TODA LA MATRIZ DE HORARIO
Private Sub Hlimpiar_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Hlimpiar.Click
    For i As Integer = 0 To 23
        Hlunes.SetItemChecked(i, False)
        Hmartes.SetItemChecked(i, False)
        Hmiercoles.SetItemChecked(i, False)
        Hjueves.SetItemChecked(i, False)
        Hviernes.SetItemChecked(i, False)
        Hsabado.SetItemChecked(i, False)
        Hdomingo.SetItemChecked(i, False)
    Next
End Sub

'CÓDIGO DE SELECCIÓN DE TODA LA MATRIZ DE HORARIO
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button2.Click
    For i As Integer = 0 To 23
        Hlunes.SetItemChecked(i, True)
        Hmartes.SetItemChecked(i, True)
        Hmiercoles.SetItemChecked(i, True)
        Hjueves.SetItemChecked(i, True)
        Hviernes.SetItemChecked(i, True)
        Hsabado.SetItemChecked(i, True)
        Hdomingo.SetItemChecked(i, True)
    Next
End Sub

'CONECTA CON UN SOCKET Y  ENVÍA UN ARREGLO DE BYTES AL CONTROLADOR
 Public Function tcpconect(ByVal func As Byte, ByVal datos() As Byte) As Byte()
    'inicio de una instancia para captura de excepciones
    'Try
    ' se crea un TcpClient.
    ' Importante este TcpClient debe conectarse a un TcpServer,
    ' La dirección a conectarse es la combinación especificada
    ' por el servidor y el puerto
    Dim port As Int32 = 13000
    Dim server As String = "192.168.0.3"
    Dim client As New System.Net.Sockets.TcpClient
    'Se conecta con el servidor
    client.Connect(server, port)
    'Se establece tiempos de Timeout para escritura y lectrura
    client.ReceiveTimeout = 20000
    client.SendTimeout = 20000
    'Buffer para lectura y escritura
    Dim data(256) As Byte
    'Se carga la funci[on en el primer byte
    data(0) = func
    'Se carga los bytescon información útil
    For i As Integer = 1 To datos.Length
```

```
            data(i) = datos(i - 1)
        Next
        'Se adiciona el caracter de fin de línea para que el servidor
        'conozca el fin de la transmisión
        data(datos.Length() + 1) = 13
        ' Obtener un stream de cliente para leer y escribir.
        ' Enviar el mensaje al TcpServer conectado.
        If client.GetStream.CanWrite And client.GetStream.CanRead Then
            'Se escribe la cadena de bytes DATA en el STREAM
            client.GetStream.Write(data, 0, datos.Length + 1)
            ' Se encera el Buffer para almacenar la respuesta de bytes.
            For i As Integer = 0 To 255
                data(i) = 0
            Next
            ' Lectura del primer batch de la respuesta en bytes del TcpServer.
            client.GetStream.Read(data, 0, data.Length)
            'Traduce el arreglo de bytes de la respuesta en una cadena  tipo string
            'Se cierra todo lo relacionado con el TcpClient
            client.GetStream.Close()
            client.Close()
        End If
        Return (data)
    End Function

    'Sale de la aplicación
    Private Sub SalirToolStripMenuItem1_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles SalirToolStripMenuItem1.Click
        Me.Close()
    End Sub

    'Muestra el formulario de conexión y prueba la conexión con el controlador
    Private Sub ActualizarHoraYFechaToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e
    As System.EventArgs) Handles ActualizarHoraYFechaToolStripMenuItem.Click, conectar_boton.Click
        Form2.Show()
    End Sub

     'Inicialización del formulario
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    MyBase.Load
        Hpisos.Items.Clear()
        For i As Integer = 1 To 32
            Hpisos.Items.Add(i)
        Next
    End Sub

    'Abre un archivo con información de las tarjetas
    Private Sub AbrirToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs)
        OpenFile.Filter = "Archivos COHECO (*.dat)|*.dat"
        OpenFile.FilterIndex = 1
        OpenFile.ShowDialog()
    End Sub

    'Selección de horario lunes
    Private Sub Label1_DoubleClick(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Handles Label1.DoubleClick
```

```
        For i As Integer = 0 To 23
            Hlunes.SetItemChecked(i, auxdias(0)) 'comprueba día lunes
        Next
        auxdias(0) = Not auxdias(0)
End Sub

 'Selección de horario martes
Private Sub Label2_DoubleClick(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Label2.DoubleClick
        For i As Integer = 0 To 23
            Hmartes.SetItemChecked(i, auxdias(1)) 'comprueba día lunes
        Next
        auxdias(1) = Not auxdias(1)
End Sub

'Selección de horario miércoles
 Private Sub Label3_DoubleClick(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Label3.DoubleClick
        For i As Integer = 0 To 23
            Hmiercoles.SetItemChecked(i, auxdias(2)) 'comprueba día lunes
        Next
        auxdias(2) = Not auxdias(2)
End Sub

 'Selección de horario jueves
 Private Sub Label4_DoubleClick(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Label4.DoubleClick
        For i As Integer = 0 To 23
            Hjueves.SetItemChecked(i, auxdias(3)) 'comprueba día lunes
        Next
        auxdias(3) = Not auxdias(3)
End Sub

'Selección de horario viernes
Private Sub Label5_DoubleClick(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Label5.DoubleClick
        For i As Integer = 0 To 23
            Hviernes.SetItemChecked(i, auxdias(4)) 'comprueba día lunes
        Next
        auxdias(4) = Not auxdias(4)
End Sub

'Selección de horario sábado
Private Sub Label6_DoubleClick(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Label6.DoubleClick
        For i As Integer = 0 To 23
            Hsabado.SetItemChecked(i, auxdias(5)) 'comprueba día lunes
        Next
        auxdias(5) = Not auxdias(5)
End Sub

'Selección de horario domingo
 Private Sub Label7_DoubleClick(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Label7.DoubleClick
        For i As Integer = 0 To 23
            Hdomingo.SetItemChecked(i, auxdias(6)) 'comprueba día lunes
```

```
        Next
        auxdias(6) = Not auxdias(6)
    End Sub

'Selección de todos los pisos
Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button3.Click
        For i As Integer = 0 To Hpisos.Items.Count() - 1
            Hpisos.SetItemChecked(i, True)
        Next
    End Sub


'Limpia todos los pisos
Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button4.Click
        For i As Integer = 0 To Hpisos.Items.Count() - 1
            Hpisos.SetItemChecked(i, False)
        Next
    End Sub

'Muestra el diálogo acerca de...
    Private Sub AcercaDeToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles AcercaDeToolStripMenuItem.Click
        AboutBox1.Show()
    End Sub

'Obtiene y forma una cadena de bytes con toda la información de un usuario
Public Function cadena_wiegand() As Byte()
        Dim usuario1(36) As Byte
        Dim wieaux As UInt16
        Dim wiegandA As Niveles
        '12 bits LSB
        wieaux = UInt16.Parse(Cwiegand.Text)
        wieaux = wieaux And &HFFF
        wiegandA.pisos = wieaux
        usuario1(0) = wiegandA.piso1
        usuario1(1) = wiegandA.piso2
        usuario1(2) = wiegandA.piso3
        usuario1(3) = wiegandA.piso4

        '12 bits MSB
        wieaux = UInt16.Parse(Cwiegand.Text)
        wieaux = wieaux \ 4096
        wieaux = wieaux Or (UInt16.Parse(Cfactibilidad.Text) * 16)
        wiegandA.pisos = wieaux
        usuario1(4) = wiegandA.piso1
        usuario1(5) = wiegandA.piso2
        usuario1(6) = wiegandA.piso3
        usuario1(7) = wiegandA.piso4


        'Obtencion de valores de acceso a pisos
        Dim aux2 As Niveles 'union usada para descomposición en bytes
        Dim aux3 As Integer
        aux2.pisos = 0
```

```
        For j As Integer = 0 To 31
            aux3 = 0
            If Hpisos.GetItemChecked(j) Then
                aux3 = 1
            End If
            aux2.pisos = aux2.pisos + aux3 * Math.Pow(2, j)
        Next

        usuario1(8) = aux2.piso1
        usuario1(9) = aux2.piso2
        usuario1(10) = aux2.piso3
        usuario1(11) = aux2.piso4

        'Obtencion de valores en la matriz de horario
        For i As Integer = 0 To 23
            usuario1(i + 12) = 0
            If Hdomingo.GetItemChecked(i) Then
                usuario1(i + 12) = 1
            End If
            If Hlunes.GetItemChecked(i) Then
                usuario1(i + 12) = usuario1(i + 12) + 2
            End If
            If Hmartes.GetItemChecked(i) Then
                usuario1(i + 12) = usuario1(i + 12) + 4
            End If
            If Hmiercoles.GetItemChecked(i) Then
                usuario1(i + 12) = usuario1(i + 12) + 8
            End If
            If Hjueves.GetItemChecked(i) Then
                usuario1(i + 12) = usuario1(i + 12) + 16
            End If
            If Hviernes.GetItemChecked(i) Then
                usuario1(i + 12) = usuario1(i + 12) + 32
            End If
            If Hsabado.GetItemChecked(i) Then
                usuario1(i + 12) = usuario1(i + 12) + 64
            End If
        Next
        'Definición de módulo a grabar
        If wiegand1.Checked Then
            usuario1(36) = 1
        End If
        If wiegand2.Checked Then
            usuario1(36) = 2
        End If
        If wiegand3.Checked Then
            usuario1(36) = 3
        End If

        Return (usuario1)
    End Function

'Borra un usuario del controlador
Private Sub Button7_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
USBorrar.Click, erase_us_boton.Click
```

```
        If comprueba_conexion() Then
            If Cwiegand.Text.Length = 0 Then
                Cwiegand.Text = "0"
            End If
            If Cfactibilidad.Text.Length = 0 Then
                Cfactibilidad.Text = "0"
            End If
            'Valida wiegand
            If Integer.Parse(Cwiegand.Text) > 65535 Or Integer.Parse(Cwiegand.Text) < 1 Then
                MsgBox("Debe ingresar un valor numérico entre 0 y 65535", 16, "Error código de tarjeta")
            Else
                'Valida código de factibilidad
                If Integer.Parse(Cfactibilidad.Text) > 255 Then
                    MsgBox("Debe ingresar un valor numérico entre 0 y 255", 16, "Error Código de
factibilidad")
                Else
                    If MsgBox("¿Está seguro de borrar el usuario?", MsgBoxStyle.Question +
MsgBoxStyle.OkCancel, "Borrar Usuario") = MsgBoxResult.Ok Then
                        Dim borrar() As Byte = wiegand(Cwiegand.Text, Cfactibilidad.Text)
                        Dim resp() As Byte = tcpconnect(12, borrar)
                        Select Case resp(1)
                            Case 0
                                MsgBox("Usuario Borrado", MsgBoxStyle.Information, "Grabación")
                            Case 1
                                MsgBox("Usuario no existe", MsgBoxStyle.Critical, "Grabación")
                        End Select
                    End If
                End If
            End If
        End If
    End Sub

'Modifica un usuario del controlador
Private Sub Button6_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
USModificar.Click, edit_us_boton.Click
        If comprueba_conexion() Then
            'Validación de las cadenas
            If Cwiegand.Text.Length = 0 Then
                Cwiegand.Text = "0"
            End If
            If Cfactibilidad.Text.Length = 0 Then
                Cfactibilidad.Text = "0"
            End If
            'Valida wiegand
            If Integer.Parse(Cwiegand.Text) > 65535 Or Integer.Parse(Cwiegand.Text) < 1 Then
                MsgBox("Debe ingresar un valor numérico entre 0 y 65535", 16, "Error código de tarjeta")
            Else
                'Valida código de factibilidad
                If Integer.Parse(Cfactibilidad.Text) > 255 Then
                    MsgBox("Debe ingresar un valor numérico entre 0 y 255", 16, "Error Código de
factibilidad")
                Else
                    If MsgBox("¿Está seguro de modificar el usuario?", MsgBoxStyle.Question +
MsgBoxStyle.OkCancel, "Borrar Usuario") = MsgBoxResult.Ok Then
                        'Obtiene el código wiegand en 2 arreglos de 13 bits cada uno
```

```
                'Solo obtiene c[odigo wiegand a diferencia de cadena_wiegand que responde con toda
la cadena
                Dim borrar() As Byte = wiegand(Cwiegand.Text, Cfactibilidad.Text)
                'borra el usuario
                Dim resp() As Byte = tcpconnect(12, borrar)
                'Obtiene los mensajes de respuesta
                Select Case resp(1)
                   Case 0
                      'graba el usuario con la nueva información
                      resp = tcpconnect(11, cadena_wiegand())
                      Select Case resp(1)
                         Case 0
                            MsgBox("El usuario ha sido modificado", MsgBoxStyle.Information,
"Modificar")
                      End Select
                   Case 1
                      MsgBox("Usuario no existe", MsgBoxStyle.Critical, "Modificar")
                End Select
             End If
          End If
       End If
    End If
End Sub


 'Graba un usuario en el controlador
Private Sub USGrabar_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
USGrabar.Click, add_us_boton.Click
    If comprueba_conexion() Then
       If Cwiegand.Text.Length = 0 Then
          Cwiegand.Text = "0"
       End If
       If Cfactibilidad.Text.Length = 0 Then
          Cfactibilidad.Text = "0"
       End If
       'Valida wiegand
       If Integer.Parse(Cwiegand.Text) > 65535 Or Integer.Parse(Cwiegand.Text) < 1 Then
          MsgBox("Debe ingresar un valor numérico entre 0 y 65535", 16, "Error código de tarjeta")
       Else
          'Valida código de factibilidad
          If Integer.Parse(Cfactibilidad.Text) > 255 Then
             MsgBox("Debe ingresar un valor numérico entre 0 y 255", 16, "Error Código de
factibilidad")
          Else

             'obtine el código wiegand en 2 arreglos de 13 bits de las cadenas de facibilidad y wiegand
             Dim borrar() As Byte = wiegand(Cwiegand.Text, Cfactibilidad.Text)
             'Envía los datos al controlador  mediante la función tcpconect1
             Dim resp() As Byte = tcpconnect(11, cadena_wiegand())
             'Obtiene mensajes de acuerdo a la respuesta del controlador
             Select Case resp(1)
                Case 0
                   MsgBox("Grabación exitosa", MsgBoxStyle.Information, "Grabación")
                Case 1
                   MsgBox("Memoria llena, no se puede grabar un usuario nuevo",
MsgBoxStyle.Critical, "Grabación")
```

```
                Case 2
                    MsgBox("Usuario duplicado no será grabado", MsgBoxStyle.Critical, "Grabación")
                End Select

            End If
        End If
    End If
End Sub

 'Muestra el diálogo para grabar usuarios
Private Sub GrabarUsuariosToolStripMenuItem1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MenuGrabarUs.Click, usuarios_grabar.Click
    If comprueba_conexion() Then
        graba_usuarios.Show()
    End If
 End Sub

'Muestra el diálogo para descargar usuarios
Private Sub DescargarUsuariosToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MenuDescargarUs.Click, usuarios_descargar.Click
    If comprueba_conexion() Then
        Login1.func = 1
        Login1.ShowDialog()
    End If
 End Sub


'Borra todos los usuarios
Private Sub BorrarTodosLosUsuariosToolStripMenuItem_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles MenuBorrarUs.Click, usuarios_eliminar.Click
    If comprueba_conexion() Then
        If MsgBox("Se eliminarán todos los usuarios del controlador, ¿Está seguro de continuar?",
MsgBoxStyle.Question + MsgBoxStyle.OkCancel, "Eliminar Usuarios") = MsgBoxResult.Ok Then
            borrar_todo(1)
            MsgBox("Todos los usuarios han sido borrados del controlador", MsgBoxStyle.Information,
"Eliminar Usuarios")
        End If
    End If
End Sub

'Borra toda la base de datos
Private Sub BorrarTodaLaBaseDeDatosToolStripMenuItem_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MenuBorraBase.Click, base_borrar.Click
    If comprueba_conexion() Then
        If MsgBox("Se eliminará toda la base de datos del controlador, ¿Está seguro de continuar?",
MsgBoxStyle.Question + MsgBoxStyle.OkCancel, "Eliminar base de datos") = MsgBoxResult.Ok Then
            borrar_todo(2)
            MsgBox("La base de datos fue borrada exitosamente", MsgBoxStyle.Information, "Borrar
Base de datos")
        End If
    End If

End Sub

'Muestra el diálogo para descargar toda la base de datos
```

```
Private Sub DescargarTodaLaBaseDeDatosToolStripMenuItem_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MenuDescargaBase.Click, base_descargar.Click
        If comprueba_conexion() Then
            descarga_datos.Show()
        End If
End Sub

'Actualiza la hora y fecha en el controlador
Private Sub ActualizarHoraYFechaToolStripMenuItem1_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles ActualizarHoraYFechaToolStripMenuItem1.Click, fecha_boton.Click
        If comprueba_conexion() Then
            'se obtiene una cadena con la informacion de hora y fecha actual
            Dim fecha() As Byte = {Now.Second(), Now.Minute(), Now.Hour(), Now.Day(), Now.Month(),
(Now.Year() - 1900), Now.DayOfWeek()}
            'Se envía los datos de hora y fecha al controlador con la función 10
            Dim respuesta() As Byte = tcpconnect(10, fecha)

            If respuesta(1) = 0 Then
                MsgBox("Hora y fecha igualada", MsgBoxStyle.Information, "Hora y Fecha")
            End If
        End If
 End Sub

    'Modifica el tiempo de activación de las salidas del módulo
    Private Sub TiempoDeActivaciónToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles TiempoDeActivaciónToolStripMenuItem.Click, activacion_boton.Click
        If comprueba_conexion() Then
            tiempo_activacion.Show()
        End If
End Sub

'Abre el formmulario Offline
Private Sub OfflineToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles OfflineToolStripMenuItem.Click
        Me.Hide()
        offline_form.Show()
    End Sub

 'Abre el formulario para consultar la base de datos
 Private Sub ConsultarBase_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles ConsultarBase.Click, boton_consultar.Click
        Consulta_base.ShowDialog()
End Sub

Private Sub Online_Shown(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MyBase.Shown
        Login1.func = 0
        Login1.Show()
 End Sub

Private Sub GeneralToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles GeneralToolStripMenuItem.Click
        modifica_password.func = 0
        modifica_password.ShowDialog()
 End Sub
```

```
Private Sub DescargaDeUsuariosToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles DescargaDeUsuariosToolStripMenuItem.Click
        modifica_password.func = 1
        modifica_password.ShowDialog()
    End Sub

End Class
```

FORMULARIO OFFLINE

```
Public Class offline_form
    'Estructura que almacena la información completa de un usuario
    'Se necesita una declaraci[on local para que pueda ser accedida desde otor diálogo
    Public Structure usuario2
        Public wiegand As Int64
        Public wieH As Int16
        Public wieL As Int16
        Public facti As Byte
        Public pisos As Int64
        Public modulo As Byte
        Public hora() As Byte
    End Structure


Public activo, modificado As Boolean 'Variables auxiliares que permiten conocer si se tiene abierto un
archivo
    Public lista As New List(Of usuario2)
    Public usuaux As usuario2
    Dim archivo As String = ""
    Public funcion As Integer = 0

 'limpia el listview
Private Sub Form3_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
        activo = False
        modificado = False
        boton_guardar.Enabled = False
        GuardarTool.Enabled = False
        ListaUsuarios.Items.Clear()
    End Sub

'Agrega un usuario al controlador
Private Sub Agregar_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Agregar.Click, boton_add.Click
        Dim item As ListViewItem
        Dim cadena As String
```

```
        funcion = 0
        If agrega_modifica.ShowDialog() = Windows.Forms.DialogResult.OK Then
            cadena = ""
            item = New ListViewItem((ListaUsuarios.Items.Count + 1).ToString, 0)

            'Añadimos el subitem del módulo al que pertenece
            cadena = usuaux.modulo.ToString()
            If usuaux.modulo = 3 Then
                cadena = "1 y 2"
            End If
            item.SubItems.Add(cadena)
            cadena = ""
            'Añadimos el subitem del código de factibilidad
            item.SubItems.Add(usuaux.facti.ToString())
            'Añadimos el subitem del código de tarjeta
            item.SubItems.Add(usuaux.wiegand.ToString())
            'Añadimos el subitem de código de pisos
            item.SubItems.Add(usuaux.pisos.ToString())
            'Obtenemos los códigos de hora
            For i As Integer = 0 To 23
                cadena = cadena + usuaux.hora(i).ToString + ","
            Next
            'Añadimos el subitem de códigos de horas
            item.SubItems.Add(cadena)
            'Añadimos el nuevo item al final del listview
            ListaUsuarios.Items.Add(item)
            'Añade el nuevo usuario al final de la lista
            lista.Add(usuaux)
            modificado = True 'Existen modificaciones en el archivo
            boton_guardar.Enabled = True
            GuardarTool.Enabled = True
        End If
    End Sub

'Abre un archivo de usuarios
Private Sub AbrirToolStripMenuItem1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles AbrirToolStripMenuItem1.Click, boton_abrir.Click
        Dim ID As Integer
        Dim aux As usuario2
        Dim cadena As String
        Dim num_ar As Integer
        ReDim aux.hora(23)

        'Si existen usuarios en el listview
        If lista.Count And modificado Then
            If MsgBox("¿Desea guardar los cambios realizados?", MsgBoxStyle.Question +
MsgBoxStyle.YesNo, "Guardar") = MsgBoxResult.Yes Then
                GrabarUsuarios()
            End If
        End If

        'Parametros del OpenFileDialog
        AbrirArchivo.Filter = "Archivos COHECO (*.coh)|*.coh"
        AbrirArchivo.FileName = ""
        AbrirArchivo.FilterIndex = 1
```

```
        'Si abrimos un archivo adeacuado
        If AbrirArchivo.ShowDialog() = System.Windows.Forms.DialogResult.OK Then
            'Nuevo item para el listview
            Dim item As ListViewItem

            ListaUsuarios.Items.Clear()    'Limpia todos los regitros del listview
            lista.Clear()

            ID = 1 'Inicializamos índice de record

            num_ar = FreeFile()
            'Abrimos el archivo en modo random
            FileOpen(num_ar, AbrirArchivo.FileName, OpenMode.Random)

            Me.Cursor = Cursors.WaitCursor 'Cambia al cursor de espera
            Me.Refresh()
            'Mientras no se llegue al fin de archivo (END OF FILE)
            Do While Not EOF(num_ar)
                cadena = "" 'Limpia la cadena auxiliar
                FileGet(num_ar, aux, ID)   ' Lee el siguiente record.
                'Asignamos un nuevo ListViewItem de acuerdo al índice de record
                item = New ListViewItem(ID.ToString(), 0)

                cadena = ""
                item = New ListViewItem((ListaUsuarios.Items.Count + 1).ToString, 0)
                'Añadimos el subitem del módulo al que pertenece
                cadena = aux.modulo.ToString()
                If aux.modulo = 3 Then
                    cadena = "1 y 2"
                End If
                item.SubItems.Add(cadena)
                cadena = ""
                'Añadimos el subitem del código de factibilidad
                item.SubItems.Add(aux.facti.ToString())
                'Añadimos el subitem del código de tarjeta
                item.SubItems.Add(aux.wiegand.ToString())
                'Añadimos el subitem de código de pisos
                item.SubItems.Add(aux.pisos.ToString())
                'Obtenemos los códigos de hora
                For i As Integer = 0 To 23
                    cadena = cadena + aux.hora(i).ToString + ","
                Next
                'Añadimos el subitem de códigos de horas
                item.SubItems.Add(cadena)
                'Añadimos el nuevo item al listview y a la lista
                ListaUsuarios.Items.Add(item)
                lista.Add(aux)
                ID = ID + 1
            Loop
            Me.Cursor = Cursors.Default
            FileClose(num_ar)
            modificado = False 'Archivo abierto y no está modificado
            boton_guardar.Enabled = False
            GuardarTool.Enabled = False
        End If
    End Sub
```

```
'Muestra el formulario Online
Private Sub ConectadoToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ConectadoToolStripMenuItem.Click
    Me.Close()
    Online.Show()
End Sub

'Manejador del evento columnclick para ordenar usuarios segun la columna
Private Sub ListaUsuarios_ColumnClick(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.ColumnClickEventArgs) Handles ListaUsuarios.ColumnClick
    ListaUsuarios.ListViewItemSorter = New ListViewItemComparer(e.Column)
End Sub

'Modifica un usuario seleccionado de la lista
Private Sub Modificar_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Modificar.Click, boton_edit.Click
    Dim indices As ListView.SelectedListViewItemCollection = ListaUsuarios.SelectedItems   'Obtiene
los items seleccionados
    Dim cadena As String

    If ListaUsuarios.Items.Count Then
      If indices.Count Then  'si existe un elemento en la lista procede a modificarlo
        Dim index As Integer = Integer.Parse(indices.Item(0).SubItems(0).Text) - 1 'Convierte a
entero el ID del item seleccionado
        Dim index1 As ListView.SelectedIndexCollection = ListaUsuarios.SelectedIndices
'Obtiene los indices de los items modificados
        Dim index2 As Integer = index1(0)
        funcion = 1 'le indica al diálogo que debe cargar la información a modificar y bloquea
        'los campos de factibilidad y número de tarjeta
        usuaux = lista(index) 'obtiene el item de la lista que será modificado
        'Abre el diálogo de modficación y ibtiene la respuesta
        If agrega_modifica.ShowDialog() = Windows.Forms.DialogResult.OK Then
          cadena = ""
          'modifica el usuario en la lista
          lista(index) = usuaux
          'modifica el usuario en el listview
          cadena = usuaux.modulo.ToString()
          If usuaux.modulo = 3 Then
            cadena = "1 y 2"
          End If
          ListaUsuarios.Items(index2).SubItems(1).Text = cadena
          ListaUsuarios.Items(index2).SubItems(2).Text = usuaux.facti.ToString()
          ListaUsuarios.Items(index2).SubItems(3).Text = usuaux.wiegand.ToString()
          ListaUsuarios.Items(index2).SubItems(4).Text = usuaux.pisos.ToString()
          cadena = ""
          For i As Integer = 0 To 23
            cadena = cadena + usuaux.hora(i).ToString + ","
          Next
          ListaUsuarios.Items(index2).SubItems(5).Text = cadena
          modificado = True
          boton_guardar.Enabled = True
          GuardarTool.Enabled = True
        End If
        ListaUsuarios.HideSelection = False 'Muestra el item seleccionado en el listview
        funcion = 0     'Indica que el diálogo está listo para agregar o modificar
```

```
            'funcion=0 -> agegar, funcion=1 -> eliminar
        Else
            MsgBox("Seleccione un elemento de la lista para modificar", MsgBoxStyle.Exclamation,
"Modificar usuario")
        End If
    Else
        MsgBox("No existen elementos en la lista para modificar", MsgBoxStyle.Exclamation,
"Modificar usuario")
    End If
End Sub

'Elimina un usuario de la lista
Private Sub Eliminar_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Eliminar.Click, boton_erase.Click
    Dim indices As ListView.SelectedListViewItemCollection = ListaUsuarios.SelectedItems 'Obtiene
los items seleccionados
    If ListaUsuarios.Items.Count Then
        If indices.Count Then   'Si existe al menos un item seleccionado
            If MsgBox("¿Está seguro de eliminar este usuario?", MsgBoxStyle.Exclamation +
MsgBoxStyle.OkCancel, "Eliminar usuario") = MsgBoxResult.Ok Then
                lista.RemoveAt(Integer.Parse(indices.Item(0).SubItems(0).Text) - 1) 'Remueve el item de
la lista
                'Que corresponde al texto existente en la columna 0 "ID"
                ListaUsuarios.Items.Remove(indices(0)) 'Remueve el item del listview
                ListaUsuarios.ListViewItemSorter = New ListViewItemComparer(0) 'Ordena los items por
ID
                'Vuelve a enumerar los elementos existentes en el lsitview
                For i As Integer = 0 To ListaUsuarios.Items.Count() - 1
                    ListaUsuarios.Items(i).SubItems(0).Text = (i + 1).ToString 'Asigna un nuevo ID a todos
los elementos de la lista
                Next
                modificado = True
                boton_guardar.Enabled = True
                GuardarTool.Enabled = True
            End If
        Else
            MsgBox("Seleccione un elemento de la lista para eliminar", MsgBoxStyle.Exclamation,
"Eliminar Usuario")
        End If
    Else
        MsgBox("No existen elementos en la lista para eliminar", MsgBoxStyle.Exclamation, "Eliminar
Usuario")
    End If
    ListaUsuarios.HideSelection = False
  End Sub

Private Sub GuardarToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles GuardarTool.Click, boton_guardar.Click
    If lista.Count And modificado Then
        GrabarUsuarios()
        modificado = False
        boton_guardar.Enabled = False
        GuardarTool.Enabled = False
    End If
End Sub
```

```
'Muestra las opciones para grabar los cambios en caso de cerrar sin grabar
Private Sub Form3_FormClosing(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.FormClosingEventArgs) Handles MyBase.FormClosing
        If lista.Count And modificado Then
            If MsgBox("¿Desea guardar los cambios realizados?", MsgBoxStyle.Question +
MsgBoxStyle.YesNo, "Guardar") = MsgBoxResult.Yes Then
                GrabarUsuarios()
            End If
        End If
        Online.Show()
 End Sub

'Función para grabar un usuarios
Private Sub GrabarUsuarios()
        GuardarArchivo.Filter = "Archivo Usuarios COHECO(*.coh)|*.coh"
        GuardarArchivo.Title = "Guardar Usuarios"
        GuardarArchivo.FilterIndex = 1
        Dim num_ar As Integer

        GuardarArchivo.ShowDialog()
        If GuardarArchivo.FileName <> "" Then
            If My.Computer.FileSystem.FileExists(GuardarArchivo.FileName) Then
                My.Computer.FileSystem.DeleteFile(GuardarArchivo.FileName)
            End If
            num_ar = FreeFile() 'Busca el siguiente número de archivo libre
            FileOpen(num_ar, GuardarArchivo.FileName, OpenMode.Random)
            'Graba las estructuras con información editada
            For n As Integer = 0 To lista.Count - 1
                FilePut(num_ar, lista(n))
            Next
            FileClose(num_ar)

        End If
End Sub

'Función de búsqueda en la lista modificada
    Public Function existelista(ByVal fact As Byte, ByVal wieg As Int64) As Boolean
        For i As Integer = 0 To lista.Count - 1
            If lista(i).facti = fact And lista(i).wiegand = wieg Then
                Return (True)
            End If
        Next
        Return (False)
 End Function

 Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
        Dim lista1 As New List(Of usuario)
        Dim aux As usuario
        ReDim aux.hora(23)
        Dim wieaux As Int64
        For i As Integer = 1 To 2000
            aux.facti = 1
            aux.modulo = 1
            aux.pisos = 1
            aux.wiegand = i
```

```
            wieaux = aux.wiegand
            wieaux = wieaux And &HFFF
            aux.wieL = wieaux

            '12 bits MSB
            wieaux = aux.wiegand
            wieaux = wieaux \ 4096
            wieaux = wieaux Or (aux.facti * 16)
            aux.wieH = wieaux
            lista1.Add(aux)
        Next
        GuardarArchivo.Filter = "Archivo Usuarios COHECO(*.coh)|*.coh"
        GuardarArchivo.Title = "Guardar Usuarios"
        GuardarArchivo.FilterIndex = 1
        GuardarArchivo.ShowDialog()
        If GuardarArchivo.FileName <> "" Then
            If My.Computer.FileSystem.FileExists(GuardarArchivo.FileName) Then
                My.Computer.FileSystem.DeleteFile(GuardarArchivo.FileName)
            End If
            FileOpen(1, GuardarArchivo.FileName, OpenMode.Random)
            'Graba las estructuras con información editada
            For n As Integer = 0 To lista1.Count - 1
                FilePut(1, lista1(n))
            Next
            FileClose(1)
        End If
    End Sub

Private Sub AbrirToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles AbrirToolStripMenuItem.Click, boton_nuevo.Click
        If lista.Count Then
            If modificado Then
                If MsgBox("¿Desea guardar los cambios realizados?", MsgBoxStyle.Question +
MsgBoxStyle.YesNo, "Guardar") = MsgBoxResult.Yes Then
                    GrabarUsuarios()
                End If
            End If
            lista.Clear()
            ListaUsuarios.Items.Clear()
        End If
        'limpia la lista y el listview
    End Sub
End Class
```

FORMULARIO CONSULTA BASE

```
'-------------------------------------------------------
'NAMSPACE, IMPORTACIONES
'-------------------------------------------------------
Imports System.IO
Public Class Consulta_base
   Dim lista As New List(Of reg)   'Lista que contiene los registros del archivo
   Dim ID As Int64         'Número que indica el total de registros en el archivo
   Dim cadena() As String  'Arreglo que contiene las cadenas con las consultas


Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Exportar.Click, ExportarTool.Click, boton_exportar.Click, ExportarToolMenu.Click
      Dim saveFileDialog1 As New SaveFileDialog()
      saveFileDialog1.Filter = "Rich Text Format(*.rtf)|*.rtf"
      saveFileDialog1.Title = "Exportar Consulta"
      saveFileDialog1.FilterIndex = 1
      saveFileDialog1.ShowDialog()

      'En caso de elegir un nombre válido para el archivo se procede a grabarlo
      If saveFileDialog1.FileName <> "" Then
         'Si ya existe el archivo este perderá toda la información y será sobreescrito
         If My.Computer.FileSystem.FileExists(saveFileDialog1.FileName) Then
            My.Computer.FileSystem.DeleteFile(saveFileDialog1.FileName)
         End If
         texto.SaveFile(saveFileDialog1.FileName)
      End If
End Sub


Private Sub AbrirToolStripMenuItem1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles AbrirToolStripMenuItem1.Click, boton_abrir.Click

      Dim aux As reg
      Dim num_ar As Integer
      AbrirArchivo.Filter = "Archivos de datos COHECO (*.cod)|*.cod"
      AbrirArchivo.FileName = ""
      AbrirArchivo.FilterIndex = 1

      'Si abrimos un archivo adeacuado
      If AbrirArchivo.ShowDialog() = System.Windows.Forms.DialogResult.OK Then
         ReDim cadena(80000)
         lista.Clear()

         boton_exportar.Enabled = False
         ExportarTool.Enabled = False
         archivo_box.Text = AbrirArchivo.FileName
         ID = 1
         lista.Clear()
         num_ar = FreeFile() 'Obtiene el próximo número de archivo libre
         FileOpen(num_ar, AbrirArchivo.FileName, OpenMode.Random)
```

```
        'Mientras no se llegue al fin de archivo (END OF FILE)
        Me.Cursor = Cursors.WaitCursor  'Establece cursor Wait
        texto.Clear() 'Limpia es cuadro de texto
        Me.Refresh()
        Do While Not EOF(num_ar)
           FileGet(num_ar, aux, ID)   ' Lee el siguiente record.
           lista.Add(aux)             'Adiciona el registro a la lista
           ID = ID + 1
        Loop
        Me.Cursor = Cursors.Default     'Establece el cursor por defecto
        FileClose(num_ar)
        ReDim Preserve cadena(ID)
        'Habilita los controles de consulta
        Panel1.Enabled = True
        Panel2.Enabled = True
        Panel3.Enabled = True
        Panel4.Enabled = True
        Consultar.Enabled = True
        boton_consultar.Enabled = True
        ConsultarTool.Enabled = True
    End If
End Sub


Private Sub Form1_FormClosing(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.FormClosingEventArgs) Handles MyBase.FormClosing
    Me.Dispose()
 End Sub

Private Sub Usuario_ch_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Usuario_ch.CheckedChanged
    If Usuario_ch.Checked Then
        Cfactibilidad.Enabled = True
        Cwiegand.Enabled = True
        Label_codigo.Enabled = True
        Label_wiegand.Enabled = True
    Else
        Cfactibilidad.Enabled = False
        Cwiegand.Enabled = False
        Label_codigo.Enabled = False
        Label_wiegand.Enabled = False
    End If
End Sub

Private Sub Modulo_ch_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Modulo_ch.CheckedChanged
    If Modulo_ch.Checked Then
        modulo_comb.Enabled = True
    Else
        modulo_comb.Enabled = False
    End If
    modulo_comb.SelectedIndex = 0
End Sub

Private Sub Fecha_ch1_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Fecha_ch1.CheckedChanged
```

```
        If Fecha_ch1.Checked Then
            fecha_1P.Enabled = True
            fecha_ch2.Enabled = True
        Else
            fecha_1P.Enabled = False
            fecha_ch2.Enabled = False
            fecha_ch2.Checked = False
        End If
    End Sub

Private Sub fecha_ch2_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles fecha_ch2.CheckedChanged
        If fecha_ch2.Checked Then
            fecha_2P.Enabled = True
        Else
            fecha_2P.Enabled = False
        End If
    End Sub

Private Sub hora_ch1_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles hora_ch1.CheckedChanged
        If hora_ch1.Checked Then
            horai.Enabled = True
            hora_ch2.Enabled = True
        Else
            horai.Enabled = False
            hora_ch2.Enabled = False
            hora_ch2.Checked = False
        End If
    End Sub

    Private Sub hora_ch2_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles hora_ch2.CheckedChanged
        If hora_ch2.Checked Then
            horaf.Enabled = True
        Else
            horaf.Enabled = False
        End If
    End Sub

'Función que permite consultar de acuerdo a las selecciones en el cuadro
Private Sub Consultar_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Consultar.Click, ConsultarTool.Click, boton_consultar.Click
        Dim k As Int64 = 1
        texto.Clear()
        'Valida los campos antes de consultar
        If validar() Then
            For j As Int64 = 0 To ID - 2
                If comparar(lista(j)) Then
                    'Da formato a la cadena
                    cadena(k) = (k.ToString + "." + vbTab + vbTab + _
                                lista(j).modulo.ToString + vbTab + _
                                lista(j).facti.ToString + vbTab + vbTab + _
                                lista(j).wiegand.ToString + vbTab + vbTab + _
                                lista(j).anio.ToString + "/" + lista(j).mes.ToString + "/" + lista(j).dia.ToString +
vbTab + _
```

```
                        lista(j).horas.ToString + ":" + lista(j).minutos.ToString + ":" +
lista(j).segundos.ToString + "    " + vbTab)
                k = k + 1
            End If
        Next
        'Cadena de encabezado
        cadena(0) = "ID          Módulo    Facti.    Cod. Wiegand    Fecha          Hora
"
        'Habilita botones de exportación
        Exportar.Enabled = True
        boton_exportar.Enabled = True
        ExportarTool.Enabled = True
        ExportarToolMenu.Enabled = True

        If k = 1 Then
            cadena(0) = "NO EXISTEN REGISTROS QUE COINCIDAN CON LOS CAMPOS"
            Exportar.Enabled = False
            boton_exportar.Enabled = False
            ExportarTool.Enabled = False
            ExportarToolMenu.Enabled = False
        End If
        ReDim Preserve cadena(k)
        texto.Lines = cadena
        ReDim cadena(80000)
    End If
End Sub
'Función que valida los campos de consulta
Public Function validar() As Boolean
    If Usuario_ch.Checked Then
        If Cwiegand.Text.Length = 0 Then
            Cwiegand.Text = "0"
        End If
        If Cfactibilidad.Text.Length = 0 Then
            Cfactibilidad.Text = "0"
        End If
        'Valida wiegand
        If Integer.Parse(Cwiegand.Text) > 65535 Or Integer.Parse(Cwiegand.Text) < 1 Then
            MsgBox("Debe ingresar un valor numérico entre 0 y 65535", 16, "Error código de tarjeta")
            Return (False)
        End If
        'Valida código de factibilidad
        If Integer.Parse(Cfactibilidad.Text) > 255 Then
            MsgBox("Debe ingresar un valor numérico entre 0 y 255", 16, "Error Código de factibilidad")
            Return (False)
        End If
    End If

    'Valida fecha
    If fecha_ch2.Checked Then
        If fecha_1P.Value > Fecha_2P.Value Then
            MsgBox("La fecha final debe ser posterior a la fecha inicial", 16, "Error Fecha")
            Return (False)
        End If
    End If

    'Valida hora
```

```
    If hora_ch2.Checked Then
        If horai.Value > horaf.Value Then
            MsgBox("La hora final debe ser posterior a la hora inicial", 16, "Error Hora")
            Return (False)
        End If
    End If

    Return (True)
End Function
'Función de comparación para las consultas
Private Function comparar(ByVal x As reg) As Boolean
    Dim fecha As New Date
    Dim aux As String
    'Compara factibilidad y usuario
    If Usuario_ch.Checked Then
        If x.facti <> Integer.Parse(Cfactibilidad.Text) Then
            Return (False)
        End If
        If x.wiegand <> Int64.Parse(Cwiegand.Text) Then
            Return (False)
        End If
    End If

    'Compara módulo
    If Modulo_ch.Checked Then
        If modulo_comb.SelectedIndex = 0 And x.modulo <> 1 Then
            Return (False)
        End If
        If modulo_comb.SelectedIndex = 1 And x.modulo <> 2 Then
            Return (False)
        End If
    End If

    aux = x.anio.ToString + "/" + x.mes.ToString + "/" + x.dia.ToString
    fecha = CType(aux, Date)

    If fecha_ch2.Checked Then
        If fecha > Fecha_2P.Value Or fecha < fecha_1P.Value Then
            Return (False)
        End If
    Else
        If Fecha_ch1.Checked Then
            If x.anio <> fecha_1P.Value.Year Then
                Return (False)
            End If
            If x.mes <> fecha_1P.Value.Month Then
                Return (False)
            End If
            If x.dia <> fecha_1P.Value.Day Then
                Return (False)
            End If
        End If

    End If

    'compara hora
```

```
        If hora_ch2.Checked Then
            If x.horas > horaf.Value Or x.horas < horai.Value Then
                Return (False)
            End If
        Else
            If hora_ch1.Checked Then
                If x.horas <> horai.Value Then
                    Return (False)
                End If
            End If
        End If

        Return (True)

    End Function
End Class
```

DIALOGO DESCARGA DATOS

```
'-------------------------------------------------------
'NAMSPACE, IMPORTACIONES
'-------------------------------------------------------

Imports System.Windows.Forms
Imports System.Data
Imports System.Data.SqlClient


Public Class descarga_datos
    Dim numreg As Int32
    Dim salir As Boolean = False


Private Sub OK_Button_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles OK_Button.Click
        Dim buff(1) As Byte
        Dim llamadas, wiegan1, wiegan2 As Int64
        Dim i, j, k As Integer
        Dim rx(numreg \ 20 + 1) As datos 'buffer de RX que recibe hasta 20 registros a la vez
        Dim rx1(numreg) As reg 'arreglo de estructuras que obtiene los datos del buffer de RX
        Dim union As Campos
        Dim fecha(numreg) As Date
        Dim num_ar As Integer
        Me.DialogResult = System.Windows.Forms.DialogResult.OK

        'Si existen usuarios se inicia la descarga
        If numreg Then
            'Número de llamadas a la función 18 para descargar registros del controlador
            llamadas = numreg \ 20
```

```
If numreg Mod 20 Then
   llamadas = llamadas + 1
End If
descargados.Visible = True
descargados.Text = "Descargados 0 usuarios de " + numreg.ToString
salir = True
For i = 1 To llamadas

   'Obtiene los registros usando la función 15
   rx(i - 1).data = tcpconnect(18, buff)
   'Modifica el valor del progress bar
   Dim aux As Integer
   aux = i * 20
   If i * 20 > numreg Then
      aux = (i - 1) * 20 + numreg Mod 20
   End If
   Progresodescarga.Value = aux
   descargados.Text = "Descargados " + aux.ToString + " registros de " + numreg.ToString

   'Permite manejar el boton cancelar mientras se descarga la base de datos
   Application.DoEvents()

   If Not salir Then
      Progresodescarga.Value = 0
      Exit For 'Permite detener la descarga
   End If
Next

'Creación del arreglo de estructuras para grabar en el archivo,
'Cuando se ha descargado todos los usuarios
If (i - 1) = llamadas Then

   For m As Integer = 1 To numreg

      'Obtención del índice de buffer de llegada
      '20 usuarios equivalen a un usuario

      j = m \ 20 - 1
      If m Mod 20 Then
         j = j + 1
      End If

      'offset dentro del buffer de llegada para poder leer los 20 registros
      k = (m - 1 - j * 20) * 16

      'Obtención de la información de cada registro
      'Obtención del código wiegand + factibilidad
      'Wiegand 12 LSB
      union.dato1 = rx(j).data(k + 1)
      union.dato2 = rx(j).data(k + 2)
      union.dato3 = rx(j).data(k + 3)
      union.dato4 = rx(j).data(k + 4)
      'Variable auxiliar para la obtención del código de factibilidad y número de tarjeta
      wiegan1 = union.dato
      'Almacena los 12 bits LSB
```

```
        'Wiegand 12 MSB
        union.dato1 = rx(j).data(k + 5)
        union.dato2 = rx(j).data(k + 6)
        union.dato3 = rx(j).data(k + 7)
        union.dato4 = rx(j).data(k + 8)
        'Variable auxiliar para la obtención del código de factibilidad y número de tarjeta
        wiegan2 = union.dato

        'número de tarjeta
        rx1(m - 1).wiegand = (wiegan2 And &HF) * 4096 Or wiegan1

        'Código de factibilidad
        rx1(m - 1).facti = (wiegan2 And &HFF0) / 16

        'Obtención del código de segundos
        rx1(m - 1).segundos = rx(j).data(k + 9)


        'Obtención del código de minutos
        rx1(m - 1).minutos = rx(j).data(k + 10)

        'Obtención de horas
        rx1(m - 1).horas = rx(j).data(k + 11)

        'Obtención día del mes
        rx1(m - 1).dia = rx(j).data(k + 12)

        'Obtención del mes
        rx1(m - 1).mes = rx(j).data(k + 13)

        'Obtención del año
            rx1(m - 1).anio = rx(j).data(k + 14) + 1900 'Corrección de compatibilidad con el módulo
rabbit

        'Obtención día de la semana
        rx1(m - 1).dayweek = rx(j).data(k + 15)

        'Obtención del modulo de funcionamiento
        rx1(m - 1).modulo = rx(j).data(k + 16)
    Next

    Me.TopMost = False 'Permite que otras ventanas aparezacan sobre el diálogo
    'Crea un saveFileDialog, establece los filtros y muestra el diálogo
    Dim saveFileDialog1 As New SaveFileDialog()
    saveFileDialog1.Filter = "Archivo registros COHECO(*.cod)|*.cod"
    saveFileDialog1.Title = "Guardar Registros"
    saveFileDialog1.FilterIndex = 1
    saveFileDialog1.ShowDialog()

    'En caso de elegir un nombre válido para el archivo se procede a grabarlo
    If saveFileDialog1.FileName <> "" Then
        'Si ya existe el archivo este perderá toda la información y será sobreescrito
        If My.Computer.FileSystem.FileExists(saveFileDialog1.FileName) Then
            My.Computer.FileSystem.DeleteFile(saveFileDialog1.FileName)
        End If
```

```
                num_ar = FreeFile() 'Obtiene el próximo número de archivo libre
                'Se abre el archivo en modo random
                FileOpen(num_ar, saveFileDialog1.FileName, OpenMode.Random)
                'Graba las estructuras con información descargada del controlador
                Me.Refresh()
                Me.Cursor = Cursors.WaitCursor
                For n As Integer = 1 To numreg
                   FilePut(num_ar, rx1(n - 1))
                Next
                FileClose(num_ar)
                Me.Cursor = Cursors.Default
            End If
            If MsgBox("¿Desea eliminar la base de datos del controlador?", MsgBoxStyle.Question +
MsgBoxStyle.OkCancel, "Eliminar base de datos") = MsgBoxResult.Ok Then
                borrar_todo(2)
                MsgBox("La base de datos fue borrada exitosamente", MsgBoxStyle.Information,
"Eliminar Base de datos")
            End If
            Me.Dispose()
            Me.Close()
         End If
         salir = False
      End If
   End Sub

Private Sub Cancel_Button_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Cancel_Button.Click
      Me.DialogResult = System.Windows.Forms.DialogResult.Cancel
      If salir Then
         salir = False 'detiene la descarga
      Else
         Me.Close()
      End If
End Sub


Private Sub Dialog3_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
      Dim nada(1) As Byte
      'Obtiene el número de registros en el controlador
      Dim resp() As Byte = tcpconnect(17, nada)
      'Unión usada para leer el número de registros en el controlador
      'Dim aux As numero16
      Dim aux1 As Campos
      'Obtención del número de 32 bits
      aux1.dato1 = resp(1)
      aux1.dato2 = resp(2)
      aux1.dato3 = resp(3)
      aux1.dato4 = resp(4)
      Progresodescarga.Value = 0
      If aux1.dato Then
         descarga.Text = "Existen " + aux1.datoD.ToString() + " registros en el controlador"
         Progresodescarga.Maximum = aux1.datoD
         OK_Button.Enabled = True
      Else
         descarga.Text = "No existen registros en el controlador"
```

```
        End If
        numreg = aux1.datoD
    End Sub
End Class
```

DIALOGO DESCARGA DATOS

```
Imports System.Windows.Forms
Imports System
Imports System.Runtime.InteropServices


Public Class descarga_usuarios
    'Numero de usuarios a descargar
    Dim numusu As Int16
    Dim salir As Boolean = False
    'Descarga los usuarios del controlador
    Private Sub OK_Button_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles OK_Button.Click
        Dim buff(1) As Byte
        Dim llamadas, wiegan1, wiegan2 As Int64
        Dim i, j, k As Integer
        Dim rx(numusu \ 8 + 1) As datos 'buffer de RX que recibe hasta 8 usuarios a la vez
        Dim rx1 As usuario
        Dim lista As New List(Of usuario)
        Dim union As Campos
        Me.DialogResult = System.Windows.Forms.DialogResult.OK
        'Si existen usuarios se inicia la descarga
        If numusu Then
            'Número de llamadas a la función 15 para descargar usuarios del controlador
            llamadas = numusu \ 8
            If numusu Mod 8 Then
                llamadas = llamadas + 1
            End If
            descargados.Visible = True
            descargados.Text = "Descargados 0 usuarios de " + numusu.ToString
            salir = True
            Cancel_Button.Focus()
            For i = 1 To llamadas
                'Obtiene un buffer de hasta 8 usuarios usando la función 15
                rx(i - 1).data = tcpconnect(15, buff)
                'Modifica el valor del progress bar
                Dim aux As Integer
                aux = i * 8
                If aux > numusu Then
                    aux = (i - 1) * 8 + numusu Mod 8
                End If
                Progresodescarga.Value = aux
                descargados.Text = "Descargados " + aux.ToString + " usuarios de " + numusu.ToString

                'Permite manejar el boton cancelar mientras se descarga la base de datos
                Application.DoEvents()
                If Not salir Then
                    Progresodescarga.Value = 0 'Detiene la descarga
```

```
            Exit For 'Sale del bucle
        End If
    Next

'Creación del arreglo de estructuras para grabar en el archivo,
    'Cuando se ha descargado todos los usuarios
    If (i - 1) = llamadas Then

        For m As Integer = 1 To numusu
'inicialización del arreglo horas dentro de la estructura
            ReDim rx1.hora(23)
            'Obtención del índice de buffer de llegada
            '8 usuarios equivalen a un índice

            j = m \ 8 - 1
            If m Mod 8 Then
                j = j + 1
            End If

'offset dentro del buffer de llegada para poder leer los 8 usuarios
            k = (m - 1 - j * 8) * 37

            'Obtención de la información de cada usuario
            'Obtención del código wiegand + factibilidad
            'Wiegand 12 LSB
            union.dato1 = rx(j).data(k + 1)
            union.dato2 = rx(j).data(k + 2)
            union.dato3 = rx(j).data(k + 3)
            union.dato4 = rx(j).data(k + 4)
'Variable auxiliar para la obtención del código de factibilidad y número de tarjeta
            wiegan1 = union.dato
            'Almacena los 12 bits LSB
            rx1.wieL = union.datoS
            'Wiegand 12 MSB
            union.dato1 = rx(j).data(k + 5)
            union.dato2 = rx(j).data(k + 6)
            union.dato3 = rx(j).data(k + 7)
            union.dato4 = rx(j).data(k + 8)
 'Variable auxiliar para la obtención del código de factibilidad y número de tarjeta
            wiegan2 = union.dato
            rx1.wieH = union.datoS

            'número de tarjeta
            rx1.wiegand = (wiegan2 And &HF) * 4096 Or wiegan1
            'Código de factibilidad
            rx1.facti = (wiegan2 And &HFF0) / 16

            'Obtención del código de pisos
            union.dato1 = rx(j).data(k + 9)
            union.dato2 = rx(j).data(k + 10)
            union.dato3 = rx(j).data(k + 11)
            union.dato4 = rx(j).data(k + 12)
            rx1.pisos = union.dato

            'Obtención de horas de acceso
            For l As Integer = 0 To 23
```

```
                    rx1.hora(l) = rx(j).data(k + 13 + l)
                Next
                'Obtención del modulo de funcionamiento
                rx1.modulo = rx(j).data(k + 37)
                lista.Add(rx1)

            Next
            'elimina los elelementos no usados en la lista
            lista.TrimExcess()
            'Ordena la lista antes de ser grabada
            'lista.Sort(AddressOf CompUsuarios)

'Crea un saveFileDialog, establece los filtros y muestra el diálogo
            Dim saveFileDialog1 As New SaveFileDialog()
            saveFileDialog1.Filter = "Archivo Usuarios COHECO(*.coh)|*.coh"
            saveFileDialog1.Title = "Guardar Usuarios"
            saveFileDialog1.FilterIndex = 1
            saveFileDialog1.ShowDialog()
            Dim num_ar As Integer
'En caso de elegir un nombre válido para el archivo se procede a grabarlo
            If saveFileDialog1.FileName <> "" Then
                If My.Computer.FileSystem.FileExists(saveFileDialog1.FileName) Then
                    My.Computer.FileSystem.DeleteFile(saveFileDialog1.FileName)
                End If
                'Obtiene el próximo número de archivo libre
                num_ar = FreeFile()
                FileOpen(num_ar, saveFileDialog1.FileName, OpenMode.Random)
                'Refresca la pantalla y cambia el cursor wait
                Me.Refresh()
                Me.Cursor = Cursors.WaitCursor
'Graba las estructuras con información descargada del controlador
                For n As Integer = 1 To numusu
                    FilePut(num_ar, lista(n - 1))
                Next
                FileClose(num_ar)
                'Cambia el cursor al valor por defecto
                Me.Cursor = Cursors.Default
            End If
            'Libera los recursos usados por el diálogo
            Me.Dispose()
            'Cierra el diálogo
            Me.Close()
        End If
        salir = False
    End If
End Sub

'Detiene la descarga y sale del diálogo
Private Sub Cancel_Button_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Cancel_Button.Click
    'Me.DialogResult = System.Windows.Forms.DialogResult.Cancel
    If salir Then
        salir = False   'Detiene la descarga
        descargados.Visible = False
    Else
        Me.Close()
```

```
        End If
    End Sub

    'Inicializa el formulario y pregunta al controlador sobre el número total de usuarios
    Private Sub Dialog1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
        'Variable auxiliar sin aplicación
        Dim nada(1) As Byte
        'Obtiene el número de usuarios en el controlador
        Dim resp() As Byte = tcpconnect(14, nada)
        'Unión usada para leer el número de usuarios en el controlador
        'Dim aux As numero16
        Dim aux1 As Campos

        'Obtención del número de 16 bits
        aux1.dato1 = resp(1)
        aux1.dato2 = resp(2)

        Progresodescarga.Value = 0
        If aux1.dato Then
            descarga.Text = "Existen " + aux1.dato.ToString() + " usuarios en el controlador"
            Progresodescarga.Maximum = aux1.dato
            OK_Button.Enabled = True
        Else
            descarga.Text = "No existen usuarios en el controlador"
        End If
        numusu = aux1.dato
        'numusu = 0
End Sub
End Class
```

ANEXO 3
Código del programa en Dynamic C

```
/*************************************************************
   CONTROL DE ACCESO-TESIS 2006 COHECO
     Lectura y almacenamiento de usuarios
     En memoria, instancias de presentación
     De tarjetas de acceso, comunicación
     TCP/IP

     Tarjeta Utilizada-> RCM 3700
*************************************************************/

//Set a default of declaring all local variables "auto" (on stack)
//#class auto

//.....................Configuracion TCP.....................................
#define TCPCONFIG 1
#define PORT 13000

#define SOCK_BUF_SIZE 800 //tamaño del buffer para el TCP (entrada y salida juntos)
#define MAX_SOCKETS 1

#memmap xmem
#use "dcrtcp.lib"

#define MAX_BUFSIZE 400//tamaño maximo de los bufer de entrada o salida

//.....................Configuracion Flash Serail.............................
//setup serial flash chip select for PE6 (RCM3700)
#define SF_SPI_CSPORT PEDR
#define SF_SPI_CSSHADOW PEDRShadow
#define SF_SPI_CSDD PEDDR
#define SF_SPI_CSDDSHADOW PEDDRShadow
#define SF_SPI_CSPIN 6


#define J_USUARIOS_PAG 7    //Numero de usuarios por pagina
#define J_USUARIOS 4011             //Numero de usuarios
#define J_PAGINAS 572               //Numero paginas
#define J_PAG_LIBRE 573             //Pagina Libre

#define J_PAG1_BASE 574      //Primera pagina para la Base de Datos
#define J_NBASE_PAG 20              //Numero de tarjetas por pagina, para la Base de Datos


#use "sflash.lib"
//.....................................................................

//.....................Configuracion Weigand.................................
#define WIEGAND_VERBOSE  // Get the library to print messages for us.
//#define WIEGAND_DEBUG     // Allow Dynamic C debugging of the
#define p_Nbits 26 //numero de bits de la tarjeta magnética.

#use "wiegand.lib"
//.....................................................................

// Socket para manejar TCP
tcp_Socket Socket;
```

```
//Estructura para detallar los usuarios
typedef struct
{
        unsigned long pisos, tarjeta[2];
        char horario[24];
  char n_wiegand;
} USUARIO;

//Estructura para detallar la Base de Datos
typedef struct
{
        unsigned long fecha, tarjeta[2];
  char n_wiegand;
} BASE_DATOS;

//Estructura para Almacenar variables criticas del sistema en Flash
typedef struct
{
        int j_n;  //Numero de usuarios almacenados
        int pag_base;           //Pagina en la cual se escribira la Base de Datos;
        char base_llena;        //Indica si la base de datos esta llena
  char tiempo_w[2];     //indica el tiempo que la interfase 1 o 2 estara activada
  int n_vacios;         //numero de espacios vacios entre los usuarios en la flash

} PG_LIBRE_VAR;

//Variables Globales
protected USUARIO j_usuario;                    //Detale de cada Usuario (lectura)

protected shared BASE_DATOS j_baseRAM[J_NBASE_PAG];   //Arreglo de estructuras
protected PG_LIBRE_VAR pg_libre;                //Variables criticas del sistema
int sf_pagesize; //Tamaño de la Pagina
protected int j_ibase;     //Indice del arreglo de estructuras para la Base de Datos


protected int n_usuario_send; //Indica cuantos usuarios o datos se han enviado o recibido
protected int n_usuarios_tot; //Numero de usuarios toal a recibir

int tiempo; //indica el tiempo que el mensaje en el LCD aparece

struct tm   rtc, horas;     //Etsructura del reloj de tiempo real

static unsigned long physaddrUsers;                     // physical memory address to write to

protected shared long dirUsuarios[J_USUARIOS];

CoData Salidas0, Salidas1, hora, imp_lcd; // Tareas especiales del sistema

//Prototipos de funciones
cofunc int p_weigandResultado(int p_lector);
cofunc int p_wiegandLeer();
void weigandOK(unsigned long p_res[]);

int pagina_flash(int pos);
```

```
speed void leer_flash(int pos);
void escribir_flash(int pos);
speed int buscar_flash(int n, unsigned long tarjeta[]);

void leer_base(int pos);
void escribir_base(int pag);
void imprimir_base(int pag);
void actualizar_pg_libre();

cofunc int Func_tcp(tcp_Socket *s, char *buf);
cofunc int conexion_tcp(tcp_Socket *s, int puerto);

void lcd_write(int data);
void lcd_puts(char * s);
void lcd_goto(unsigned char pos);
void lcd_clear();
void init_LCD();
void lcd_imprimir(int pos1, char *msg1, int pos2, char *msg2, int t);

void cargarRAM();
//>>>>>>>>>>>>>>FUNCION PRINCIPAL>>>>>>>>>>>>>>>>>>>>>>>>>>
int main()
{
  char j_key;     //Tecla presionada-Seleccion de Accion
  char j_op[5];   //Tecla presionada-Confirmacion Borrar
  int j_i;        //Contador general
  char j_aux[30];                 //Auxiliar para ingreso de datos
  int j_encontrado;       //Usuario encontrado
        unsigned long j_tarjet[2];          //Tarjeta encontrada
  unsigned int p_i; //contador auxiliar
  int p_bit;
        unsigned long p_pisos,aux;
  char p_horario;

        int p_rc;

  char fecha_hora[17];

  _sysIsSoftReset();      // restore any protected variables


//INICIALIZACION DE PUERTOS_____________________________________

  WrPortI(SPCR, &SPCRShadow, 0x084); //PortA Salida
  WrPortI(PADR ,&PADRShadow, 0x00);  //PortA=0
  WrPortI(PBDDR, &PBDDRShadow, 0xFF);      // PortB outputs
  WrPortI(PBDR ,&PBDRShadow, 0x00);  //PortB0
  BitWrPortI(PBDR, &PBDRShadow,1,3);//Output Enable1=1 (salidas deshabilitadas)
  BitWrPortI(PGDDR, &PGDDRShadow,1,7);//PG7 OUT
  BitWrPortI(PGDCR, &PGDCRShadow,0,7);//Salida normal
  WrPortI(PFDDR ,&PFDDRShadow, 0xFF);  //Puerto F como salidas
  WrPortI(PFFR ,&PFFRShadow, 0x00);  //Puerto F
  WrPortI(PFDCR ,&PFDCRShadow, 0x00); //como E/S normales
        BitWrPortI(PFDR, &PFDRShadow,1,6);//Output Enable2=1 (salidas deshabilitadas)

 //Inicializacion Interfases
```

```
          //Interfaas0
  BitWrPortI(PBDR, &PBDRShadow,1,3);//Output Enable=1
  p_pisos=0;
  for (p_i=0;p_i<32;p_i++)
  {
    p_bit= bit(&p_pisos,31-p_i);
    BitWrPortI(PBDR,&PBDRShadow,p_bit,5);//data
    BitWrPortI(PBDR, &PBDRShadow,0,2);//clock
    aux=MS_TIMER;
    while(MS_TIMER<=aux+1);
    BitWrPortI(PBDR, &PBDRShadow,1,2);//clock
  }
   BitWrPortI(PBDR, &PBDRShadow,0,3);//Output Enable=0

   //Interfas1
  BitWrPortI(PFDR, &PFDRShadow,1,6);//Output Enable = 1
  p_pisos=0;
  for (p_i=0;p_i<32;p_i++)
  {
    p_bit= bit(&p_pisos,31-p_i);
    BitWrPortI(PFDR,&PFDRShadow,p_bit,4);//data
    BitWrPortI(PFDR, &PFDRShadow,0,5);//clock
    aux=MS_TIMER;
    while(MS_TIMER<=aux+1);
    BitWrPortI(PFDR, &PFDRShadow,1,5);//clock
  }
  BitWrPortI(PFDR, &PFDRShadow,0,6);//Output Enable =0


  // clockDoublerOn();// trabaja a 22 Mhz(NO FUNCIONA CON DEBUG!!)
   init_LCD();

//INICIALIZACION DE TCP_____________________________________

  sock_init();
  sock_mode(&Socket,TCP_MODE_ASCII);

//INICIALIZACION DE PUERTO WEIGAND_________________________

  p_rc=wiegand_init(0, 0, 0, 0, 0, p_Nbits);    //Inicializa WEIGAND

            //  data0--> PE0
                                  //  data1--> PE4
  if (p_rc)
  {

     lcd_clear();
     lcd_goto(0x01);
     lcd_puts(" INI. Error");
     lcd_goto(0x41+0x3);
     lcd_puts("WIEGAND 0");


     exit(1);
  }//if rc
```

```
   p_rc=wiegand_init(1, 0, 0, 0, 0, p_Nbits);    //Inicializa WEIGAND

            //  data0--> PE1
                                  //  data1--> PE5
  if (p_rc)
  {

     lcd_clear();
     lcd_goto(0x01);
     lcd_puts(" INI. Error");
     lcd_goto(0x41+0x3);
     lcd_puts("WIEGAND 1");
     exit(1);
  }//if rc


//INICIALIZACION DE FLASH SERIAL Y RAM_____________________________



          sfspi_init();        //Inicializa el controlador SPI para el uso con flash serial
          if(sf_init())
          {

      lcd_clear();
      lcd_goto(0x01);
      lcd_puts(" INI. Error");
      lcd_goto(0x41+0x2);
      lcd_puts("Flash Serial");
             exit(1);
           }
           else
           {
   /* Variables auxiliares en FLASH en la pagina J_PAG_LIBRE
                    1. j_n              Numero de usuarios almacenados (NO real)
       2. pag_base                      Numero de pagina a almacenar la base de datos
       3. base_llena        Bandera que indica si la Base de datos esta llena
       4. tiempo_w[2]                   Tiempo de activacion de las salidas
       5. n_vacios                      Espacios vacios en la memoria de usuarios
   */

     //j_n, pag_base, base_llena
     sf_pageToRAM((long)J_PAG_LIBRE);
                    sf_readRAM((char *)&pg_libre, 0, sizeof(pg_libre));


   if(j_ibase<0 || j_ibase>= (int)J_NBASE_PAG) j_ibase=0; //en caso de que se haya perdido la bateria

                                                          //de back-up (se llega a perder maximo 20 usuarios)


     cargarRAM();//Carga en RAM los datos de los usuarios almacenados en flash

     //pg_libre.j_n=4000;
     //pg_libre.n_vacios=0;
     // pg_libre.pag_base=4095;
```

```
    // actualizar_pg_libre();
    // j_ibase=17;


   }
//FIN---INICIALIZACION FLASH SERIAL_____________________

  for (;;)//_______BUCLE INFINITO_____________________
  {

          costate imp_lcd//_____MENSAJE LCD_____________________
    {             //DELAY PARA PRESENTAR EL LCD


       CoPause(&hora);

       waitfor(DelaySec(tiempo));

                    CoBegin(&hora);

    }//_________________MENSAJE LCD_____________________________

    costate hora always_on
    {
         tm_rd(&horas); //caraga la hora actual en la estructura
         lcd_clear();

         lcd_puts("COHECO Cia.Ldta.");
         sprintf(fecha_hora,"%02d/%02d/%02d   %02d:%02d",horas.tm_year-
100,horas.tm_mon,horas.tm_mday,horas.tm_hour,horas.tm_min);
         lcd_goto(0x40);
         lcd_puts(fecha_hora);
         waitfor(DelaySec(2));

         if(pg_libre.base_llena=='S')
         {
           lcd_clear();
           lcd_goto(0x01);
          lcd_puts("Base de Datos");
           lcd_goto(0x45);
           lcd_puts("Llena!");
           waitfor(DelaySec(2));
         }

    }

    costate//_________LECTURA DE TARJETAS________
    {
      //waitfor(!( isCoRunning(&Salidas0) || isCoRunning(&Salidas1) ) ); // no lee una nueva tarjeta
hasta que la antarior se haya realizado
       wfd p_wiegandLeer();

    }//________FIN DE LECTURA DE TARJETAS________

    costate//_________MANEJO DE TCP_______________
    {
```

```
                wfd conexion_tcp(&Socket, PORT);
            }//_________FIN MANEJO DE TCP_____________________


costate//_________PILA TCP_____________________
{
                // majeja la pila del TCP
                tcp_tick(NULL);
            }//_________FIN PILA TCP_____________________


 costate Salidas0//__Validacion de horarios(lector 0)____
{


    tm_rd(&rtc); //caraga la hora actual en la estructura

  p_pisos= j_usuario.pisos;
  p_horario=j_usuario.horario[rtc.tm_hour];


  BitWrPortI(PBDR, &PBDRShadow,1,3);//Output Enable =1

  if( !bit(&p_horario,rtc.tm_wday) )// no esta dentro del horario
  {

    lcd_clear();
    lcd_imprimir(2,"Usuario fuera",3,"de horario!",3);

  }else //Esta dentro del horario
  {
      BitWrPortI(PBDR, &PBDRShadow,1,3);//Output Enable=1
      for (p_i=0;p_i<32;p_i++)
            {
              p_bit= bit(&p_pisos,31-p_i);

              BitWrPortI(PBDR,&PBDRShadow,p_bit,5);//data


              BitWrPortI(PBDR, &PBDRShadow,0,2);//clock

              waitfor(IntervalMs(1)); //   delay
              BitWrPortI(PBDR, &PBDRShadow,1,2);//clock
            }


     BitWrPortI(PBDR, &PBDRShadow,0,3);//Output Enable=0
          waitfor(DelaySec((long)pg_libre.tiempo_w[0]));//tiempo de activacion
          BitWrPortI(PBDR, &PBDRShadow,1,3);//Output Enable=1

      p_pisos=0;
      for (p_i=0;p_i<32;p_i++)
            {
              p_bit= bit(&p_pisos,31-p_i);
```

```
                    printf(" %d",p_bit);
                    BitWrPortI(PBDR,&PBDRShadow,p_bit,5);//data


                    BitWrPortI(PBDR, &PBDRShadow,0,2);//clock

                    waitfor(IntervalMs(1)); //   delay
                    BitWrPortI(PBDR, &PBDRShadow,1,2);//clock
                   }

          BitWrPortI(PBDR, &PBDRShadow,0,3);//Output Enable=0


               }

}

 costate Salidas1//__Validacion de horarios(lector 1)____
 {


       tm_rd(&rtc); //caraga la hora actual en la estructura

    p_pisos= j_usuario.pisos;
    p_horario=j_usuario.horario[rtc.tm_hour];



    BitWrPortI(PFDR, &PFDRShadow,1,6);//Output Enable =1

    if( !bit(&p_horario,rtc.tm_wday) )// no esta dentro del horario
    {

      lcd_clear();
      lcd_imprimir(2,"Usuario fuera",3,"de horario!",3);

    }else //Esta dentro del horario
    {

        BitWrPortI(PFDR, &PFDRShadow,1,6);//Output Enable = 1
      for (p_i=0;p_i<32;p_i++)
                {
                  p_bit= bit(&p_pisos,31-p_i);

                  BitWrPortI(PFDR,&PFDRShadow,p_bit,4);//data


                  BitWrPortI(PFDR, &PFDRShadow,0,5);//clock

                  waitfor(IntervalMs(1)); //   delay
                  BitWrPortI(PFDR, &PFDRShadow,1,5);//clock
                 }


                BitWrPortI(PFDR, &PFDRShadow,0,6);//Output Enable =0
                waitfor(DelaySec((long)pg_libre.tiempo_w[1]));//tiempo de activacion
```

```
                        BitWrPortI(PFDR, &PFDRShadow,1,6);//Output Enable = 1

                p_pisos=0;
                for (p_i=0;p_i<32;p_i++)
                        {
                          p_bit= bit(&p_pisos,31-p_i);

                          BitWrPortI(PFDR,&PFDRShadow,p_bit,4);//data


                          BitWrPortI(PFDR, &PFDRShadow,0,5);//clock

                          waitfor(IntervalMs(1)); //   delay
                          BitWrPortI(PFDR, &PFDRShadow,1,5);//clock
                        }

                BitWrPortI(PFDR, &PFDRShadow,0,6);//Output Enable =0


                      }

     }//________FIN DE VALIDACION DE HORARIOS________


   }//___________for(;;) fin BUCLE INFINITO______________
}// fin main
//>>>>>>>>> FIN DE FUNCION PRINCIPAL >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>



//<<<<<<<<<<<<<<<<<<<<<FUNCIONES<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

//*************** FUNCION DETECTAR DATOS EN LAS INTERFASES WEIGAND
cofunc int p_wiegandLeer()
{

  int p_rc;

                      p_rc=wiegand_ready();

           if (!p_rc) return 0;
     else
     {

           if(bit(&p_rc,0))
           wfd p_weigandResultado(0);

        if(bit(&p_rc,1))
           wfd p_weigandResultado(1);

      }//if rc else

          return (1);
}
```

```
//*************** FUNCION PARA LEER DATOS DE LAS INTERFASES WEIGAND
cofunc int p_weigandResultado(int p_lector)
{
          int p_rc;
  char n_wiegand;
  unsigned long p_res[2], p_raw_res[2];


                        p_rc=wiegand_result(p_lector, p_raw_res, p_res);
      if (p_rc == WIEGAND_OK)
      {

        if (buscar_flash(pg_libre.j_n,p_res))
        {

          n_wiegand=j_usuario.n_wiegand;

          if (p_lector==0 && (bit(&n_wiegand,0)==1)  )// verifica si el pertenece al lector 0
          {
                if(!isCoRunning(&Salidas0))//si la interfaz no esta activada
            {
                        CoBegin(&Salidas0);// activa la tarea salidas
              j_baseRAM[j_ibase].n_wiegand=1;// activacion del lector 1
              lcd_imprimir(0,"COHECO Cia.Ltda.",2,"Bienvenido...",3);
              j_baseRAM[j_ibase].n_wiegand=p_lector+1;
                        weigandOK(p_res);

            }else
              lcd_imprimir(0,"Interfaz ocupada",0,"intente otra vez",3);


          }else
          {

                if (p_lector==1 && (bit(&n_wiegand,1)==1) )// verifica si el pertenece al lector 1
                  {
              if(!isCoRunning(&Salidas1)) //si la interfaz no esta activada
                {
                      CoBegin(&Salidas1);// activa la tarea salidas
                      j_baseRAM[j_ibase].n_wiegand=2;// activacion del lector 2
                      lcd_imprimir(0,"COHECO Cia.Ltda.",2,"Bienvenido...",3);
                j_baseRAM[j_ibase].n_wiegand=p_lector+1;
                                weigandOK(p_res);

              }else
                        lcd_imprimir(0,"Interfaz ocupada",0,"intente otra vez",3);

                  }else
              lcd_imprimir(1,"No pertenece a",2,"este lector!",3);

          }// if plector=0..... else


          }else
          {
```

```
              lcd_clear();
              lcd_imprimir(3,"Tarjeta no",2,"registrada!",3);

            }//if res.... else

          } //if weigand_ok

                        return (1);
}

//*************** FUNCION PARA EJECUTAR ACCIONES CUANDO UN USUARIO ENTRA AL
SISTEMA
void weigandOK(unsigned long p_res[])
{

  //Almacenamiento en Base de Datos>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
   j_baseRAM[j_ibase].tarjeta[0]=p_res[0];
   j_baseRAM[j_ibase].tarjeta[1]=p_res[1];
   j_baseRAM[j_ibase].fecha=SEC_TIMER;    // guarada la hora actual


   j_ibase++;

   if(pg_libre.pag_base <= sf_blocks && j_ibase == (int)J_NBASE_PAG)//Almacenar!!
   {
      escribir_base(pg_libre.pag_base);
      j_ibase=0;
      pg_libre.pag_base++;
      actualizar_pg_libre();
   }

   if(pg_libre.pag_base >= sf_blocks && j_ibase == 0)
   {
      pg_libre.base_llena='S';
      actualizar_pg_libre();
      j_ibase=0;  //Indice de la Base de Datos
      pg_libre.pag_base = (int)J_PAG1_BASE;
      actualizar_pg_libre();
   }
         //Fin Almacenamiento
}

//<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<


//*************** FUNCION OBTEBER LA PAGINA DE UN USUARIO DE LA MEMORIA FLASH
int pagina_flash(int pos)
{
   //La funcion devuelve el valor de la Pagina de la posicion deseada
   //pos: Posicion Deseada
   int pag;          //Pagina
   int i;    //Contador general

   for(i=1, pag=-1; i<=(int)J_USUARIOS; i=i+(int)J_USUARIOS_PAG, pag++)
   {
         if(pos >= (i-(int)J_USUARIOS_PAG) && pos < i)
```

```
        break;
  }
  return(pag);
}


//*************** FUNCION PARA LEER UN USUARIO DE LA MEMORIA FLASH
speed void leer_flash(int pos)
{
        //pos: Posicion en la que se va a Leer

  /*
  int pag;        //Pagina

  pag = pagina_flash(pos);
  sf_pageToRAM(pag);
        sf_readRAM((char *)&j_usuario, (pos-1-pag*(int)J_USUARIOS_PAG)*sizeof(j_usuario),
sizeof(j_usuario));
  */

  xmem2root(&j_usuario,dirUsuarios[pos-1],sizeof(USUARIO));

}
//*************** FUNCION PARA ESCRIBIR UN USUARIO EN LA MEMORIA FLASH
void escribir_flash(int pos)
{
        //pos: Posicion en la que se va a Escribir
  int pag;        //Pagina

  pag = pagina_flash(pos);
  sf_pageToRAM(pag);
        sf_writeRAM((char *)&j_usuario, (pos-1-pag*(int)J_USUARIOS_PAG)*sizeof(j_usuario),
sizeof(j_usuario));
        sf_RAMToPage(pag);
  sf_pageToRAM(pag);
  sf_readRAM((char *)&j_usuario, (pos-1-pag*(int)J_USUARIOS_PAG)*sizeof(j_usuario),
sizeof(j_usuario));

  root2xmem(dirUsuarios[pos-1],&j_usuario,sizeof(USUARIO));

}

//*************** FUNCION PARA BUSCAR UN USUARIO EN LA MEMORIA FLASH
speed int buscar_flash(int n, unsigned long tarjeta[])
{
  //La funcion devuelve el valor de la Posicion del Usuario buscado
  //n: Numero actual de Usuarios
  //tarjeta: Numero de la tarjeta a ser Buscada


  int pag;        //Pagina
  int i;          //Contador general

          for(i=1; i<=n; i++)
          {
              leer_flash(i);
```

```
                if(j_usuario.tarjeta[0] == tarjeta[0] && j_usuario.tarjeta[1] == tarjeta[1])
                {
                  return(i);  //Tarjeta i encontrada
                }


            }


  return(0);        //Tarjeta no encontrada


}

//<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

//*************** FUNCION PARA ESCRIBIR BASE DE DATOS EN LA MEMORIA FLASH
void escribir_base(int pag)
{
        //pag: Pagina en que se Escribira

  sf_pageToRAM(pag);
        sf_writeRAM((char *)j_baseRAM, 0, sizeof(j_baseRAM));
        sf_RAMToPage(pag);
}

//*************** FUNCION PARA IMPRIMIR UNA PAGINA DE LA BASE DE DATOS
void imprimir_base(int pag)
{
        //pag: Pagina en que se Imprimira
  int i;

  sf_pageToRAM(pag);
        sf_readRAM((char *)j_baseRAM, 0, sizeof(j_baseRAM));

  for(i=0; i<(int)J_NBASE_PAG; i++)
  {
    mktm( &rtc, j_baseRAM[i].fecha );  // transforma los segundos a estructura

  }
}

//*************** FUNCION PARA ACTUALIZAR LOS VALORES DE LA PAGINA LIBRE
void actualizar_pg_libre()
{
        sf_writeRAM((char *)&pg_libre,0, sizeof(pg_libre));
        sf_RAMToPage((long)J_PAG_LIBRE);

        sf_pageToRAM((long)J_PAG_LIBRE);
        sf_readRAM((char *)&pg_libre,0, sizeof(pg_libre));
}

//*************** FUNCION PARA RESPONDER A LAS PETICIONES DE COMUNICACION
cofunc int Func_tcp(tcp_Socket *s, char *buf)
{
```

```
  int i,j,aux;
  int j_encontrado;         //Usuario encontrado
          unsigned long j_tarjet[2];            //Tarjeta encontrada

  char respuesta[MAX_BUFSIZE];

  BASE_DATOS base[J_NBASE_PAG]; //arreglo auxiliar para enviar la base de datos al PC
  USUARIO usuarios[J_USUARIOS_PAG];//arreglo auxiliar para la recepcion de usuarios

  union//Union para la recepcion de datos (usuarios)
          {
            char bytes[8];
            unsigned long numero[2];
    int entero;
          } campos;

  //HACER PRUEBA CON UNA UNION EN LA ESTRUCTURA!!!!!!!!!!!

          switch(buf[0])
  {
          case 10://Igualar reloj
          rtc.tm_sec=buf[1];
          rtc.tm_min=buf[2];
          rtc.tm_hour=buf[3];
          rtc.tm_mday=buf[4];
          rtc.tm_mon=buf[5];
          rtc.tm_year=buf[6];
          rtc.tm_wday=buf[7];

          tm_wr(&rtc); //iguala el reloj de tiempo real
                                      SEC_TIMER = mktime(&rtc);  //iguala la variable de tiempo del
sistema

          lcd_imprimir(2,"Fecha y hora",1,"actualizadas.",3);

          respuesta[0]=10;
          respuesta[1]=0;
          sock_fastwrite(s, respuesta, 2); //Indica la operacion se realizo correctamente

        break;

        case 11: //Recepcion de los datos de datos de un usuario (codigo, pisos, horario)


            //Busca espacio en memoria
            if((pg_libre.j_n-pg_libre.n_vacios)==(int)J_USUARIOS)
            {

              lcd_imprimir(1,"No hay espacio",2,"en memoria!",3);
              respuesta[0]=11;
              respuesta[1]=1;
              sock_fastwrite(s, respuesta, 2);  // respuesta al PC
              break;
            }

            if(pg_libre.n_vacios>0)
```

```
  {
                    j_tarjet[0] = 0;
          j_tarjet[1] = 0;
          j_encontrado = buscar_flash(pg_libre.j_n, j_tarjet);//Busca un espacio libre
  }else
         j_encontrado=0;


//recibe los campos de la tarjeta wiegand
for (i=1; i<=8; i++)
            campos.bytes[i-1]=buf[i];

j_tarjet[0]=campos.numero[0];
j_tarjet[1]=campos.numero[1];

//Verifica si el usuario ya existe
if(buscar_flash(pg_libre.j_n,j_tarjet))
{

  lcd_imprimir(1,"Usuario ya esta",2,"en memoria!",3);
  respuesta[0]=11;
  respuesta[1]=2;
  sock_fastwrite(s, respuesta, 2);  // respuesta al PC
  break;
}else
{
    j_usuario.tarjeta[0]=campos.numero[0];
  j_usuario.tarjeta[1]=campos.numero[1];
}

//recibe los campos de los pisos
campos.bytes[0]=buf[9];
campos.bytes[1]=buf[10];
campos.bytes[2]=buf[11];
campos.bytes[3]=buf[12];

j_usuario.pisos=campos.numero[0];

//recibe los campos de los horarios
for(i=13; i<=36; i++)
 j_usuario.horario[i-13]=buf[i];

 // recibe el lector al que pertenece
j_usuario.n_wiegand=buf[37];

//Almacenamiento en memoria Flash

  if(j_encontrado > 0)
  {

    escribir_flash(j_encontrado);//Escritura en FLASH
    pg_libre.n_vacios--;


  }else // si no hay espacio libres crea uno al final
  {
```

```
            //Actualiza j_n el flash
        pg_libre.j_n++;
        escribir_flash(pg_libre.j_n);//Escritura en FLASH
      }
      actualizar_pg_libre();

      lcd_imprimir(4,"Usuario",2,"almacenado.",3);

      respuesta[0]=11;
      respuesta[1]=0;
      sock_fastwrite(s, respuesta, 2); //Indica la operacion se realizo correctamente
    break;

            case 12: //Borra un usuario

     //recibe los campos de la tarjeta wiegand
            for (i=1; i<=8; i++)
               campos.bytes[i-1]=buf[i];

           j_tarjet[0]=campos.numero[0];
           j_tarjet[1]=campos.numero[1];

           //Verifica si el usuario no existe
      j_encontrado=buscar_flash(pg_libre.j_n,j_tarjet);

           if(!j_encontrado)
           {

             lcd_imprimir(1,"Usuario no esta",2,"en memoria!",3);
             respuesta[0]=12;
             respuesta[1]=1;
             sock_fastwrite(s, respuesta, 2);  // respuesta al PC
             break;
      }else
      {
        j_usuario.tarjeta[0]=0;
        j_usuario.tarjeta[1]=0;
        j_usuario.pisos=0;
        escribir_flash(j_encontrado);//Escritura en FLASH
        pg_libre.n_vacios++;
        actualizar_pg_libre();
        respuesta[0]=12;
             respuesta[1]=0; // ok
             sock_fastwrite(s, respuesta, 2);  // respuesta al PC
        lcd_imprimir(4,"Usuario",3,"eliminado!",3);

      }
    break;
  case 13: // Borrar todo el bloque de memoria

     if(buf[1]==1)//Borra USUARIOS
     {
        pg_libre.j_n=0;          //Numero actual de Usuarios
        pg_libre.n_vacios=0;
        actualizar_pg_libre();
```

```
            lcd_imprimir(0,"Los usuarios han",0,"sido eliminados",3);
          }

          if(buf[1]==2)// Borra e inicia base de datos
          {
              pg_libre.base_llena='N';
              actualizar_pg_libre();

              j_ibase=0;    //Indice de la Base de Datos

              pg_libre.pag_base = (int)J_PAG1_BASE;
              actualizar_pg_libre();
              lcd_imprimir(0,"Base de datos ha",1,"sido eliminada",3);

          }


           respuesta[0]=13;
                  respuesta[1]=0; // ok
                  sock_fastwrite(s, respuesta, 2);  // respuesta al PC
          break;

         case 14: //Numero de usuarios e inicializacion para envio de usuarios

           campos.entero=pg_libre.j_n-pg_libre.n_vacios;
           n_usuario_send=0;
           respuesta[1]=campos.bytes[0];
           respuesta[2]=campos.bytes[1];
           respuesta[0]=14;
                  sock_fastwrite(s, respuesta, 3);  // respuesta al PC

            lcd_imprimir(0,"Usuarios listos",1,"a enviarse...",3);
         break;
       case 15://Peticion de envio de los datos de un usuario

           respuesta[0]=15;
           j_tarjet[0]=0;
           j_tarjet[1]=0;

          for(j=0;j<8;j++)
         {

             if (n_usuario_send==pg_libre.j_n) break;

              do
                     {

                n_usuario_send++;
                       leer_flash(n_usuario_send);

                     }while( (j_usuario.tarjeta[0] == j_tarjet[0]) && (j_usuario.tarjeta[1] == j_tarjet[1]) &&
(n_usuario_send<pg_libre.j_n));



                     //Codigo Wiegand
```

```
                    campos.numero[0]=j_usuario.tarjeta[0];
                    campos.numero[1]=j_usuario.tarjeta[1];

                   for(i=1;i<=8;i++)
                      respuesta[i+j*37]=campos.bytes[i-1];

                   //Pisos
                   campos.numero[0]=j_usuario.pisos;

                   for(i=9;i<=12;i++)
                      respuesta[i+j*37]=campos.bytes[i-9];

                   //horarios
                   for(i=13;i<=36;i++)
                      respuesta[i+j*37]=j_usuario.horario[i-13];

                   //Lector
                   respuesta[37+j*37]=j_usuario.n_wiegand;

        }

         sock_fastwrite(s, respuesta, 37*j+1);  // respuesta al PC


                  break;
      case 16://Carga al controlador los tiempos de activacion de las interfases 1 y 2

           pg_libre.tiempo_w[buf[1]-1]=buf[2];
                     actualizar_pg_libre();

           respuesta[0]=16;
           respuesta[1]=0;//ok
           sock_fastwrite(s, respuesta, 2); // respuesta al PC

           lcd_imprimir(0,"Modficado tiempo",1,"de activacion.",3);
         break;
      case 17://Numero de datos e inicializacion para envio de base de datos
         respuesta[0]=17;

         if(pg_libre.base_llena=='N')
             campos.numero[0]=((unsigned long)pg_libre.pag_base-(unsigned
long)J_PAG1_BASE)*(unsigned long)J_NBASE_PAG + (unsigned long)j_ibase;//numero toatal de
registros
         else //si la base esta llena
           campos.numero[0]=((unsigned long)sf_blocks-(unsigned long)J_PAG1_BASE)*(unsigned
long)J_NBASE_PAG + (unsigned long)j_ibase;//numero toatal de resgistros

          for(i=1;i<=4;i++)
          respuesta[i]=campos.bytes[i-1];

           sock_fastwrite(s, respuesta, 5); // respuesta al PC

           n_usuario_send=0;// inicializa los datos

           lcd_imprimir(3,"Registros",1,"enviandose....",3);
```

```
          break;

        case 18://Envio de los registros de la base de tatos

          respuesta[0]=18;

        if(pg_libre.base_llena=='N') aux=pg_libre.pag_base;
        else aux=(int)sf_blocks;

        if(n_usuario_send<(aux-(int)J_PAG1_BASE))//envia los datos de la flash
        {
          sf_pageToRAM((long)n_usuario_send+(long)J_PAG1_BASE);
                                        sf_readRAM((char *)base, 0, sizeof(base));//craga pagina de
flash a ram (serial)

          for(i=0;i<(int)J_NBASE_PAG;i++)
          {
            //codigo de la tarjeta
            campos.numero[0]=base[i].tarjeta[0];
            campos.numero[1]=base[i].tarjeta[1];

            for(j=1;j<=8;j++)
                    respuesta[j+i*16]=campos.bytes[j-1];

            //Fecha y Hora
            mktm( &rtc, base[i].fecha);
            respuesta[9+i*16]=rtc.tm_sec;
                  respuesta[10+i*16]=rtc.tm_min;
                    respuesta[11+i*16]=rtc.tm_hour;
                    respuesta[12+i*16]=rtc.tm_mday;
                    respuesta[13+i*16]=rtc.tm_mon;
                    respuesta[14+i*16]=rtc.tm_year;
                    respuesta[15+i*16]=rtc.tm_wday;

            //Numero de lector
            respuesta[16+i*16]=base[i].n_wiegand;
            yield;
          }
          sock_fastwrite(s, respuesta, (int)J_NBASE_PAG*16+1); // respuesta al PC
          n_usuario_send++;


        }else // envia datos de la ram
        {
                        for(i=0;i<=j_ibase;i++)
                {
                  //codigo de la tarjeta
                  campos.numero[0]=j_baseRAM[i].tarjeta[0];
                  campos.numero[1]=j_baseRAM[i].tarjeta[1];

                  for(j=1;j<=8;j++)
                    respuesta[j+i*16]=campos.bytes[j-1];

                  //Fecha y Hora
                  mktm( &rtc, j_baseRAM[i].fecha);
                  respuesta[9+i*16]=rtc.tm_sec;
```

```
                respuesta[10+i*16]=rtc.tm_min;
                respuesta[11+i*16]=rtc.tm_hour;
                respuesta[12+i*16]=rtc.tm_mday;
                respuesta[13+i*16]=rtc.tm_mon;
                respuesta[14+i*16]=rtc.tm_year;
                respuesta[15+i*16]=rtc.tm_wday;

                //Numero de lector
                respuesta[16+i*16]=j_baseRAM[i].n_wiegand;
              }
              sock_fastwrite(s, respuesta, j_ibase*16+2); // respuesta al PC

  }
   break;

case 19: //Aviso para recibir Usuarios

   //lee numero de usuarios que se van a recibir
   campos.bytes[0]=buf[1];
          campos.bytes[1]=buf[2];

          n_usuarios_tot=campos.entero;

    n_usuario_send=0;
          respuesta[0]=19;
          respuesta[1]=0;//ok

          sock_fastwrite(s, respuesta, 2); // respuesta al PC

    lcd_imprimir(3,"Recibiendo",2,"usuarios....",3);

   break;

case 20: //Recepcion de usuarios

              for(j=0; j<(int)J_USUARIOS_PAG; j++)
              {

           //recibe los campos de la tarjeta wiegand
          for (i=1; i<=8; i++)
             campos.bytes[i-1]=buf[i+j*37];

          usuarios[j].tarjeta[0]=campos.numero[0];
          usuarios[j].tarjeta[1]=campos.numero[1];


          //recibe los campos de los pisos
          campos.bytes[0]=buf[9+j*37];
          campos.bytes[1]=buf[10+j*37];
          campos.bytes[2]=buf[11+j*37];
          campos.bytes[3]=buf[12+j*37];

          usuarios[j].pisos=campos.numero[0];

          //recibe los campos de los horarios
          for(i=13; i<=36; i++)
```

```
                    usuarios[j].horario[i-13]=buf[i+j*37];

                  // recibe el lector al que pertenece
                  usuarios[j].n_wiegand=buf[37+j*37];


                  }//for(j=0....
               //Almacenamiento en memoria Flash

                        n_usuario_send++;
            aux=(int) J_USUARIOS_PAG;

            if(n_usuario_send <= (n_usuarios_tot/aux))
                  {
                  pg_libre.j_n=n_usuario_send*7;

            }

            else
            {
                  pg_libre.j_n+=(n_usuarios_tot%(int)J_USUARIOS_PAG);

            }

            actualizar_pg_libre();
            sf_writeRAM((char *)usuarios,0 , sizeof(usuarios));
            sf_RAMToPage((long)(n_usuario_send-1));
            sf_pageToRAM((long)(n_usuario_send-1));
            sf_readRAM((char *)usuarios,0 , sizeof(usuarios));

            for(i=(n_usuario_send-1)*7,j=0;i<(n_usuario_send)*7;i++,j++)
                  root2xmem(dirUsuarios[i],&usuarios[j],sizeof(USUARIO));

         respuesta[0]=20;
               respuesta[1]=0;//ok

               sock_fastwrite(s, respuesta, 2); // respuesta al PC

      break;

  }//Switch buf[0]

      return (1);
}

 //<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

//*************** FUNCION PARA MANEJAR LA INFORMACION TCP
 // Realiza la funcion de manejo de informacion
cofunc int conexion_tcp(tcp_Socket *s, int puerto)
{
      char buf[MAX_BUFSIZE];

      int length;

      tcp_listen(s, puerto, 0, 0, NULL, 0);// escucha la llegada de una conexion en el puerto
```

```
  // espera por la coneccion
        while((-1 == sock_bytesready(s)) && (0 == sock_established(s)))
        // suelta el control para que otras tareas se ejecuaten
        yield;
        while(sock_established(s))
  {


    //space_avaliable = sock_tbleft(s);

                //if(space_avaliable > (MAX_BUFSIZE))// revisa si hay espacio
                  //        space_avaliable = (MAX_BUFSIZE);

                // obtener datos
                length = sock_fastread(s, buf, (int)MAX_BUFSIZE);

    if(length > 0)
    { //si se recibio datos

      wfd Func_tcp(s, buf);//Ejecuta acciones pediadas por el sistema remoto

                                    //sock_fastwrite(s, buf, length);
                }

                yield; // suelta el control a otras tareas
        }
        sock_close(s);

  //lcd_imprimir(2,"Conexion TCP",2,"finalizada.",3);


        return (1);
}

//*************** FUNCION PARA ESCRIBIR EN EL LCD
void lcd_write(int data)
{

  unsigned long aux;


  WrPortI(PADR ,&PADRShadow, data);  //PortA=data
  BitWrPortI(PGDR, &PGDRShadow,1,7);//EN=1 Enable en nivel H
  aux=MS_TIMER;
  while(MS_TIMER-aux<1);
  BitWrPortI(PGDR, &PGDRShadow,0,7);//EN=0 Enable en nivel L
 //        wait_lcd();

}
//*************** FUNCION PARA POSICIONAR CURSOR EN EL LCD
void lcd_goto(unsigned char pos)
{
        BitWrPortI(PBDR, &PBDRShadow,0,7);//RS=0 Trabajar en el Registro de Control
        lcd_write(0x80+pos);     //Escribe instruccion para posicion del Cursor
}
```

```
//*************** FUNCION PARA ESCRIBIR CADENAS EN EL LCD
void lcd_puts(char * s)
{
        while(*s)
        {
                BitWrPortI(PBDR, &PBDRShadow,1,7);//RS=1
                lcd_write(*s++);
        }
}
//*************** FUNCION PARA BORRAR EL LCD
void lcd_clear()
{
        BitWrPortI(PBDR, &PBDRShadow,0,7);//RS=0 Trabajar en el Registro de Control
        lcd_write(0x01); //Limpiar el LCD
  lcd_write(0x02);        //LCD a home

}
//*************** FUNCION PARA IMPRIMIR TODO EL LCD
void lcd_imprimir(int pos1, char *msg1, int pos2, char *msg2, int t)
{


    CoBegin(&imp_lcd);
    lcd_clear();

    lcd_goto(pos1);
    lcd_puts(msg1);
    lcd_goto(0x40+pos2);
    lcd_puts(msg2);

    tiempo=t;
}
void init_LCD() //*********INICIALIZACION LCD(Se ejecuta sola una vez al inicio)
{
    unsigned long aux;
                BitWrPortI(PBDR, &PBDRShadow,0,7);//RS=0 Trabajar en el Registro de Control
        BitWrPortI(PBDR, &PBDRShadow,0,4);//RW=0 Trabajar en Escritura
        BitWrPortI(PGDR, &PGDRShadow,0,7);//EN=0 Enable en nivel L
        aux=MS_TIMER;
        while(MS_TIMER<=aux+15);//Retraso recomendado por
        aux=MS_TIMER;
                        while(MS_TIMER<=aux+15);// las hojas tecnicas
                WrPortI(PADR ,&PADRShadow, 0x03);  //PortA=0x03 Alerta al LCD que va a recibir
instrucciones

        BitWrPortI(PGDR, &PGDRShadow,1,7);//EN=1 Enable en nivel H
        aux=MS_TIMER;
                        while(MS_TIMER<=aux+1); //Pulso (flanco descendente) en Enable
        BitWrPortI(PGDR, &PGDRShadow,0,7);//EN=0 Enable en nivel L

        aux=MS_TIMER;
                        while(MS_TIMER<=aux+5); //Retraso recomendado por
        aux=MS_TIMER;
                        while(MS_TIMER<=aux+5); // las hojas tecnicas
        BitWrPortI(PGDR, &PGDRShadow,1,7);//EN=1 Enable en nivel H
```

```
        aux=MS_TIMER;
                        while(MS_TIMER<=aux+1); //Pulso (flanco descendente) en Enable
        BitWrPortI(PGDR, &PGDRShadow,0,7);//EN=0 Enable en nivel L

        aux=MS_TIMER;
                        while(MS_TIMER<=aux+1);//Retraso recomendado por las hojas tecnicas
        BitWrPortI(PGDR, &PGDRShadow,1,7);//EN=1 Enable en nivel H
       aux=MS_TIMER;
                        while(MS_TIMER<=aux+1); //Pulso (flanco descendente) en Enable
        BitWrPortI(PGDR, &PGDRShadow,0,7);//EN=0 Enable en nivel L

       aux=MS_TIMER;
                        while(MS_TIMER<=aux+5);//Retraso recomendado por
        aux=MS_TIMER;
                        while(MS_TIMER<=aux+5); // las hojas tecnicas
                WrPortI(PADR ,&PADRShadow, 0x02);  //PortA=0x02 Cursor a Home (Posicion 0x00)
        BitWrPortI(PGDR, &PGDRShadow,1,7);//EN=1 Enable en nivel H
        aux=MS_TIMER;
                        while(MS_TIMER<=aux+1); //Pulso (flanco descendente) en Enable
        BitWrPortI(PGDR, &PGDRShadow,0,7);//EN=0 Enable en nivel L

        aux=MS_TIMER;
                        while(MS_TIMER<=aux+1);//Retraso recomendado por las hojas tecnicas
                BitWrPortI(PBDR, &PBDRShadow,0,7);//RS=0 Trabajar en el Registro de Control

        lcd_write(0x38);//DL:Interfase de 8 bits, N:2 líneas, caracter de 5x8 pixeles (function set)
                lcd_write(0x08);//D:display On, C: cursor On, B:blink On,(display control)
                lcd_write(0x06);//I/D:Incrementa contador de direcciones, S:display quieto (entry mode
set)
                lcd_write(0x0C);////D:display ON, C: cursor OFF, B:blink OFF,(display control)

        lcd_clear();
        lcd_puts("PSSM DUO");
        lcd_goto(0x40);
        lcd_puts("INICIALIZANDO...");
}
//<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

//*************** FUNCION PARA CARGAR LOS USUARIOS DE FLASH A RAM
void cargarRAM()
{

   int i,pag;
   for(i=0;i<J_USUARIOS;i++)
  {

    dirUsuarios[i]=xalloc(sizeof(USUARIO));
    pag=pagina_flash(i+1);
    sf_pageToRAM(pag);
                sf_readRAM((char *)&j_usuario, (i-pag*(int)J_USUARIOS_PAG)*sizeof(j_usuario),
sizeof(j_usuario));
    root2xmem(dirUsuarios[i],&j_usuario,sizeof(USUARIO));
  }

  lcd_imprimir(0,"COHECO Cia.Ldta.",4,"PSSM DUO",4);
}
```

FECHA DE ENTREGA

Ing. Víctor Proaño
COORDINADOR DE CARRERA

Giovanny Paul Padilla Cazar
AUTOR

Alejandro Sebastián Mera Balseca
AUTOR