TRABAJO ESPECIAL DE GRADO

DISEÑO E IMPLEMENTACIÓN DE UNA INTERFAZ DE SEGURIDAD PARA EL CONTROL DE ACCESIBILIDAD DE UN ASCENSOR

Presentado ante la Ilustre
Universidad Central de Venezuela
por el Br. Gil A., Elis R.
para optar al Título de
Ingeniero Electricista

Caracas, 2011.

TRABAJO ESPECIAL DE GRADO

DISEÑO E IMPLEMENTACIÓN DE UNA INTERFAZ DE SEGURIDAD PARA EL CONTROL DE ACCESIBILIDAD DE UN ASCENSOR

Prof. Guía: Ing. Joao Nunes
Tutor Industrial: Ing. Gabriel Clemencot

Presentado ante la Ilustre
Universidad Central de Venezuela
Por el Br. Gil A., Elis R.
para optar al título de
Ingeniero Electricísta

Caracas, 2011

CONSTANCIA DE APROBACIÓN

Caracas, 6 de junio de 2011

Los abajo firmantes, miembros del Jurado designado por el Consejo de Escuela de Ingeniería Eléctrica, para evaluar el Trabajo Especial de Grado presentado por el Bachiller Elis Rafael Gil Alvarez, titulado:

"DISEÑO E IMPLEMENTACIÓN DE UNA INTERFAZ DE SEGURIDAD PARA EL CONTROL DE ACCESIBILIDAD DE UN ASCENSOR"

Consideran que el mismo cumple con los requisitos exigidos por el plan de estudios conducente al Título de Ingeniero Electricista en la mención Electrónica, y sin que ello signifique que se hacen solidarios con las ideas expuestas por el autor, lo declaran APROBADO.

Prof. José Alonso
Jurado

Prof. Tamara Pérez
Jurado

Prof. Joao L. Nunes
Profesor Guía

DEDICATORIA

El presente trabajo lo dedico con todo el afecto y agradecimiento a la memoria de mi abuelo y padre Fortunato Alvarez (mi tío), confiando en que Dios lo tenga en su santa gloria por siempre y que desde allí observe el éxito que ambos con ésta pequeña obra hemos alcanzado. Por siempre estarás presente en todos mis pasos y por tu inmensa labor y guía estaré eternamente agradecido contigo.

RECONOCIMIENTOS Y AGRADECIMIENTOS

Primeramente a Dios nuestro Señor por iluminarme y guiarme por el mejor de los caminos, ya que sin él nada sería posible,

A mi madre Rosiris por el inmenso apoyo que me ha dado no sólo en mi carrera sino durante toda mi vida brindándome confianza, amistad, seguridad y cuidado en todo momento.

A mi padre Elis por haberme dado esta gran oportunidad de estudio confiando en que esta meta alcanzada siempre sería lograda.

A mi abuela Juana que mañana, tarde y noche se ha esforzado en ser la mejor abuela, logrando ser más que eso una madre para mí.

A Johan, mi otro padre, por haberme enseñado y guiado por este camino de estudio y por la confianza puesta en mi persona.

A Yine por todo su apoyo y comprensión a lo largo de mi carrera, brindándome su amistad incondicional en todo momento.

A los demás miembros de mi familia fruto de la meta alcanzada, todo y más se los agradeceré por siempre.

A Joao mi tutor por su gran y valiosa colaboración a lo largo de todo el proyecto ya que sin su ayuda no se habría logrado en conjunto con mis compañeros de estudio, por su contribución y amistad incondicional en especial a mi hermano Gabriel, Javier y Juan, muchísimas gracias amigos míos.

Gil A., Elis R.

DISEÑO E IMPLEMENTACIÓN DE UNA INTERFAZ DE SEGURIDAD PARA EL CONTROL DE ACCESIBILIDAD DE UN ASCENSOR

Profesor Guía: Prof. Joao L. Nunes. Tutor Industrial: Ing. Gabriel Clemencot. Tesis. Caracas. U.C.V. Facultad de Ingeniería. Escuela de Ingeniería Eléctrica. Opción: Electrónica. Institución: Ascensores Servas S.A. 2011. 62 h. + anexos.

Palabras Clave: Control; Acceso; Ascensor; Microcontrolador.

Resumen. Se plantea el diseño e implementación de una interfaz para el control de acceso en un ascensor, desarrollado en las instalaciones de la empresa Ascensores Servas S.A., con sede en Caracas. Esta interfaz opera mediante el control de un microcontrolador dispuesto como maestro, permitiendo otorgar convenientemente el acceso hacia un piso por medio del ingreso de un código programado, bajo intervalos de horas prefijadas por un usuario, o bien por la ausencia del sistema de seguridad.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

Pág.

Cap. III:

Cap. IV:

SIGLAS

- AC: Corriente Alterna.
- BCD: Binary Code Decimal (Código Decimal Binario).
- CPU: Central Processing Unit (Unidad de Procesamiento Central).
- DC: Corriente Continua.
- DDRAM: Display Data Random Access Memory (Visualización de Datos de Memoria de Acceso Aleatorio).
- EEPROM: Electrically Erasable Programmable Read Only Memory (Memoria de Sólo Lectura Programable y Borrable Eléctricamente).
- HMI: Human-Machine Interface (Interfaz Hombre-Máquina).
- Hz: Hertz.
- I2C: Inter-Integrated Circuit (Circuito Inter-Integrado).
- LCD: Liquid Chrytal Display (Pantalla de Cristal Líquido).
- MIPS: Millones de Instrucciones Por Segundo.
- PC: Program Counter (Contador del Programa).
- PCB: Printed Circuit Board (Placa de Circuito Impreso).
- PROM: Programmable Read Only Memory (Memoria de Sólo Lectura Programable).
- PSP: Parallel Slave Port (Puerto Paralelo Esclavo).
- RAM: Random Access Memory (Memoria de Acceso Aleatorio).
- RFID: Radio Frequency IDentification (Identificación Por Radio Frecuencia).
- RISC: Reduced Instruction Set Computer (Computador con Conjunto de Instrucciones Reducido).
- RTC: Real Time Clock (Reloj de Tiempo Real).
- S.A: Sociedad Anónima.
- SCA: Sistemas de Control de Acceso.
- TTL: Transistor-Transistor Logic (Lógica Transistor-Transistor).

INTRODUCCIÓN

A pesar de que los ascensores se encontraban en uso a partir del siglo III a.C., activados por energía humana, animal o por fuerza de agua, el ascensor moderno es en gran parte producto del siglo XIX. Estos sistemas fueron diseñados para movilizar, de forma vertical, personas o bienes entre diferentes alturas bien sea para ascender o descender. Debido a una gran demanda a causa de la construcción de edificios cada vez más altos, las personas se sentían menos inclinadas a subir escaleras, además que los grandes almacenes comenzaron a prosperar, por lo que surgió la necesidad de implementar un aparato para el traslado tanto de personas como de bienes con un mínimo de esfuerzo.

A causa de tal demanda, un estadounidense de Vermont llamado Elisha Otis presento una demostración de un sistema elevador de pasajeros, resultando para la comunidad un diseño satisfactorio. Estos ascensores eran accionados generalmente por una máquina de vapor ya fuera de manera indirecta a través de un tipo de tracción hidráulica o bien de forma directa.

Posteriormente, los ascensores fueron elaborados con partes mecánicas, eléctricas y electrónicas que funcionaban conjuntamente, logrando un medio seguro de movilidad. Fundamentalmente se instalan dos tipos de ascensores: El electromecánico y el hidráulico, o propiamente llamado oleodinámico.

Seguidamente con el despliegue de la tecnología surge la necesidad de dotar a los ascensores con cierta inteligencia a fin de que mantenga llamadas en espera, reconozca la existencia de algún evento de emergencia, presente prioridad de llamada a uno o varios pisos, entre otras características con la ayuda de un microcontrolador o en su defecto un PLC.

La realización de este trabajo formula una propuesta de accesibilidad por medio de un ascensor partiendo del ingreso de códigos a través de un teclado, del ingreso de horas predeterminadas o bien trabajando como un ascensor de servicio privado.

Para llevar a cabo esta propuesta, se desarrolla este trabajo el cual se encuentra dividido en cuatro capítulos desarrollados de la siguiente manera:

El **Capítulo 1** plantea el problema por el cual se desarrolla en presente trabajo, así como el objetivo general y los objetivos específicos del mismo, los cuales permiten definir el alcance de este trabajo.

En el **Capítulo 2** se documentan los aspectos teóricos de las tecnologías vinculadas a los sistemas de seguridad más frecuentes, detallando las características de cada estándar.

El **Capítulo 3** presenta los módulos y periféricos del sistema diseñado, resaltando su composición así como las características y configuraciones empleadas para su mejor desempeño. De esta manera se exhiben en forma de bloques funcionales cada uno de los módulos que constituyen la interfaz para su mejor entendimiento, así como también se muestra una descripción general de la interfaz y la del módulo de aplicación.

En el **Capítulo 4** se da a conocer la metodología de programación empleada, y la estructura contenida en cada sub-bloque del programa elaborado descrita mediante diagramas de flujo con su respectiva explicación.

Finalmente se establecen conclusiones y recomendaciones que, basadas en la realización de pruebas de campo, refuerzan la implementación de sistemas de

seguridad y accesibilidad asociados a los ascensores de la compañía interesada en la elaboración del proyecto en cuestión.

CAPÍTULO I

PLANTEAMIENTO GENERAL

1.1 Planteamiento del Problema

La compañía ASCENSORES SERVAS S.A. tiene la necesidad de implementar un sistema que permita el control de ciertas operaciones en sus ascensores, entre las cuales se destacan el movimiento de cabina, manejo de llamadas y acceso a los pisos.

Para solventar la problemática presentada por ésta compañía al momento de controlar las operaciones ejercidas por un ascensor, se decidió realizar la migración hacia la inclusión de una interfaz electrónica que permita el control de tales operaciones sin recurrir a la reprogramación propia del mismo.

Para la implementación de la interfaz de control se realizó un estudio al entorno del ascensor buscando en todo momento aprovechar la infraestructura existente, es decir, la cabina junto al tablero de control, así como también los respectivos actuadores, de manera de acoplar junto a estos elementos la interfaz a implementar.

El sistema a implementar debe ser capaz de controlar el tiempo de apertura de puertas así como el control de acceso a los pisos según un horario preestablecido, ofrecer prioridad a un número de pisos de acuerdo al flujo de personas, fijar el ascensor en un piso predeterminado debido a la ocurrencia de alguna emergencia y prestar servicio de control de pisos por contraseña.

En base a lo antes planteado y buscando obtener información necesaria para el despliegue satisfactorio del nuevo sistema de control, se plantea la implementación de un hardware que permita satisfacer los requerimientos exigidos y generar un software que se comporte de forma idónea según tales requerimientos.

1.2 OBJETIVOS DEL PROYECTO

1.2.1 Objetivo General

Diseñar e implementar una interfaz de control electrónica para manejo de accesibilidad de un ascensor.

1.2.2 Objetivos Específicos

- Realizar una serie de estudios que permitan conocer el comportamiento de los elementos que componen el sistema.
- Seleccionar un hardware adecuado que permita satisfacer los requerimientos para el control del sistema.
- Diseñar un software asociado que permita satisfacer los requerimientos de control del sistema.
- Implementar el sistema definitivo.
- Realizar el análisis de resultados correspondiente a la implementación realizada.

CAPÍTULO II

BASES TEÓRICAS

2.1 Sistemas de Control de Acceso [1]

Los Sistemas de Control de Acceso (SCA) son una popular solución de seguridad para pequeñas o grandes empresas, cuya finalidad es permitir convenientemente el acceso a personas que tengan autorización para ingresar o egresar de un recinto o área determinada.

Inicialmente el acceso hacia un determinado lugar dentro de las instalaciones de una empresa estaba supervisado por guardias o vigilantes, los cuales se encontraban ubicados en áreas cuyos niveles de seguridad debían ser superiores, y a su vez las puertas limitantes entre zonas poseían trancas de madera o metal, candados con llaves de metal, cerraduras o cerrojos mecánicos (manuales), cadenas con candados o bien llaves especiales, permitiendo que tales niveles de seguridad fueran lógicamente en aumento.

Ésta medida de seguridad se adecuaba a los requerimientos básicos de cada compañía, permitiendo tener controlada cada una de sus áreas dada la poca cantidad de empleados que estas manejaban. Posteriormente las necesidades de cada compañía fueron cambiando, exigiendo así un número mayor de personal, lo cual implicaba de forma directa un aumento de los niveles de seguridad y con ello dar lugar a una nueva generación de SCA.

Posteriormente, el avance de la tecnología ah permitido integrar una serie de SCA de diversas índoles según el nivel de seguridad requerido, logrando adaptarse de manera eficaz en cada entorno laboral. Ahora bien, una de las principales características que se debe tener en cuenta es la facilidad de uso el cual no debería requerir de una formación especial para su utilización ya que los empleados o usuarios pueden estar utilizando el SCA varias veces al día. Entre los sistemas de control de acceso más utilizados actualmente se encuentran:

- Tarjetas de proximidad.
- Tarjetas de banda magnética.
- Tarjetas con código de barras.
- Control remoto.
- Lectores biométricos.
- Teclados.

2.1.1 Tarjetas de Proximidad [2]

Las tarjetas de proximidad son dispositivos que envían señales de radiofrecuencia a un lector diseñado para tal fin cuando se encuentran en su cercanía, cuya finalidad es la de transmitir la identidad del usuario para el cual la misma fue diseñada. Estos dispositivos también son denominados RFID (Radio Frequency IDentification).

Estos dispositivos son denominados de proximidad ya que la información de la tarjeta puede ser transmitida sin tocar la lectora. El lector RFID o transceptor está compuesto por tres elementos los cuales son la antena, el transceptor y el decodificador.

La función que ejecuta es la siguiente: El lector envía periódicamente señales a fin de verificar si existe alguna tarjeta en su cercanía. Cuando capta la señal de una

tarjeta extrae la información de identificación contenida en la misma y se la pasa al subsistema de procesamiento de datos. El mismo verifica los datos y procede a ceder el acceso o no de acuerdo a la información procesada.

2.1.2 Tarjetas de Banda Magnética [3]

Este tipo de tarjetas son aquellas que poseen una banda de color negra o marrón en su parte posterior, la cual está compuesta por partículas ferromagnéticas incrustadas en una matriz de resina, y que almacenan cierta cantidad de información mediante una codificación determinada.

Las partículas que conforman la banda magnética pueden ser magnetizadas en dirección norte o sur para dar origen a su codificación, permitiendo grabar información en la banda. Esta información puede ser leída mediante contacto físico, pasándola a través de un dispositivo de lectura/escritura gracias al fenómeno de la inducción magnética, con lo que posteriormente la información contenida en la banda magnética puede ser modificada.

Una vez pasada la tarjeta a través del dispositivo de lectura/escritura, la información contenida en la tarjeta es decodificada y procesada, permitiendo dar acceso o no al solicitante según la información suministrada.

2.1.3 Tarjetas con Código de Barras [4]

Este tipo de tarjetas son aquellas que poseen un código de barras ubicado al frente o al dorso de la misma, la cual contiene una clave de manera codificada para acceder a cierta cantidad de información.

Su representación está dada mediante una disposición de líneas paralelas ordenadas de forma vertical, con distinto grosor y espaciado, los cuales corresponden

a una clave codificada que permite dar acceso a un registro de una determinada base de datos, en donde realmente reside la información.

El lector infrarrojo es pasado sobre el código de barras permitiendo su escaneo, una vez conseguida su lectura, la misma es enviada a una base de datos la cual contiene la información de cada usuario. De acuerdo a tal información, el acceso al usuario es concedido o rechazado, permitiendo posteriormente realizar un nuevo escaneo.

2.1.4 Control Remoto

Este sistema de control de acceso es un dispositivo que trabaja mediante ondas de radiofrecuencia determinadas. Este tipo de dispositivos es controlado a distancia lo cual hace posible que sea fácilmente manipulable con una sola mano, y generalmente diseñados con 2 o 3 botones para diversas operaciones según su aplicación.

Al ser pulsado el botón para dar acceso, una señal eléctrica alimenta un resonador (cristal) que se encuentra en su interior, el mismo transmite un pulso de ondas a una determinada frecuencia. Las ondas son captadas y posteriormente decodificadas por un módulo receptor accionando de esta manera una serie de elementos electrónicos que permiten dar acceso o no al usuario.

A diferencia de las tarjetas, en este dispositivo no se guarda ningún tipo de información referida al usuario, lo cual resulta ser básicamente una llave para el acceso a distancia.

2.1.5 Lectores Biométricos [5]

Este tipo de sistemas comprende el control de acceso de personas mediante la utilización de rasgos personales como lo son la geometría de la mano (HandKey), la voz, huella digital, la retina, reconocimiento facial o bien el reconocimiento de firma del usuario. Este tipo de sistemas posee una gran ventaja respecto a los demás, y la misma radica en que cada persona puede ser identificada en forma eficaz dado que tales rasgos son inviolables, intransferibles e infalsificables, haciéndolos independientes de cualquier tipo de documento, además de no tener la necesidad de llevar consigo alguna llave o tarjeta la cual podría extraviar.

La eficacia de este tipo de SCA se basa en el algoritmo utilizado para generar un Template (algoritmo codificado), el cual sea tan complejo que la generación del mismo y su resguardo consuman pocos recursos del sistema como lo son el tiempo y espacio dedicados en la memoria. De igual forma, dicho sistema debe ser eficiente a tal grado que al querer identificar a un determinado usuario este sea capaz de reconocerlo, sin importar la posición del cuerpo en cada registro, pues de no ser así la razón de rechazo será muy elevada.

2.1.6 Teclados

Los teclados numéricos como sistemas de control de acceso son muy populares en estos días. Estos dispositivos son un periférico de entrada que utiliza una disposición de botones o teclas que, acoplados a elementos electrónicos, pueden enviar información a un computador o controlador para ser procesada.

La programación que posee este tipo de dispositivos es de forma local o bien mediante software externo. Estos sistemas que permiten la programación localmente son denominados sistemas inteligentes, llamados de esta manera debido a que bajo el teclado poseen una circuitería que comprende desde el cableado hasta su propio

microprocesador, el cual permite controlar el ingreso y egreso hacia áreas restringidas o de alta seguridad permitiendo la activación de cerraduras, molinetes, barreras, entre otros. El mismo no requiere de otras interfaces para su programación, por lo que al pulsar de manera correcta una serie de teclas o bien mediante una llave especial se puede acceder a su opción de reprogramación. Su capacidad de almacenamiento varía según la empresa diseñadora, llegando a capacidades de almacenamiento de hasta 8000 registros, pudiendo particionar esta memoria entre códigos habilitados y eventos almacenados según su necesidad.

Por su parte, los teclados numéricos programados mediante software externo son dependientes de una tarjeta de control externa, alojada de manera distante. Son llamados de software externo ya que los mismos requieren de otros periféricos de control para ser programados. Sus capacidades de almacenamiento dependen de igual manera del microprocesador con el que se encuentre implementado, pudiendo controlar, además del acceso hacia áreas de mayor seguridad, una serie de periféricos asociados a otros eventos como lo son el control de alarmas, cámaras, sensores de presencia, movimiento de cabina de un ascensor, entre otros.

2.2 Sistemas de Servicio Privado

Los sistemas de servicio privado son una solución de seguridad generalmente empleados en casas o en edificaciones en donde cada piso posee un propietario general, cuya finalidad es permitir convenientemente la accesibilidad de personas con movilidad reducida, que tengan autorización para ingresar o egresar de una determinada instalación por medio de una clave de acceso ingresada manualmente o por medio de un dispositivo que la suministre.

Estos servicios privados hacen uso de los sistemas de control de acceso, particularizándose en el entorno en donde se llevará a cabo tal instalación. De forma independiente al SCA empleado, su función radica en agregar identificación personal,

lo que significa realizar la llamada hacia el piso deseado con tan sólo el discado del código de seguridad que este tiene asignado, permitiendo la parada del ascensor en la planta establecida y dando la posibilidad de eliminar la botonera de la cabina del ascensor.

CAPÍTULO III

3.1 DISEÑO DEL HARDWARE DE CONTROL DE ACCESO

El hardware diseñado está compuesto de una tarjeta conformada por una serie de componentes con funciones y propósitos diferentes. Por medio de una serie de conectores con entradas y salidas seleccionadas estratégicamente, éste módulo se acopla tanto a la botonera del ascensor como a su respectiva tarjeta de control principal para formar un producto determinado, ya sea el presentado en éste documento o cualquier otro del que se desee disponer para proyectos futuros. Éste módulo está diseñado para ser reutilizable e intercambiable entre ascensores diferentes, siempre que cumplan con los parámetros establecidos por la compañía en cuestión. A continuación se describen las características del sistema. La Figura 1 muestra un diagrama de bloques de las partes que conforman la interfaz.

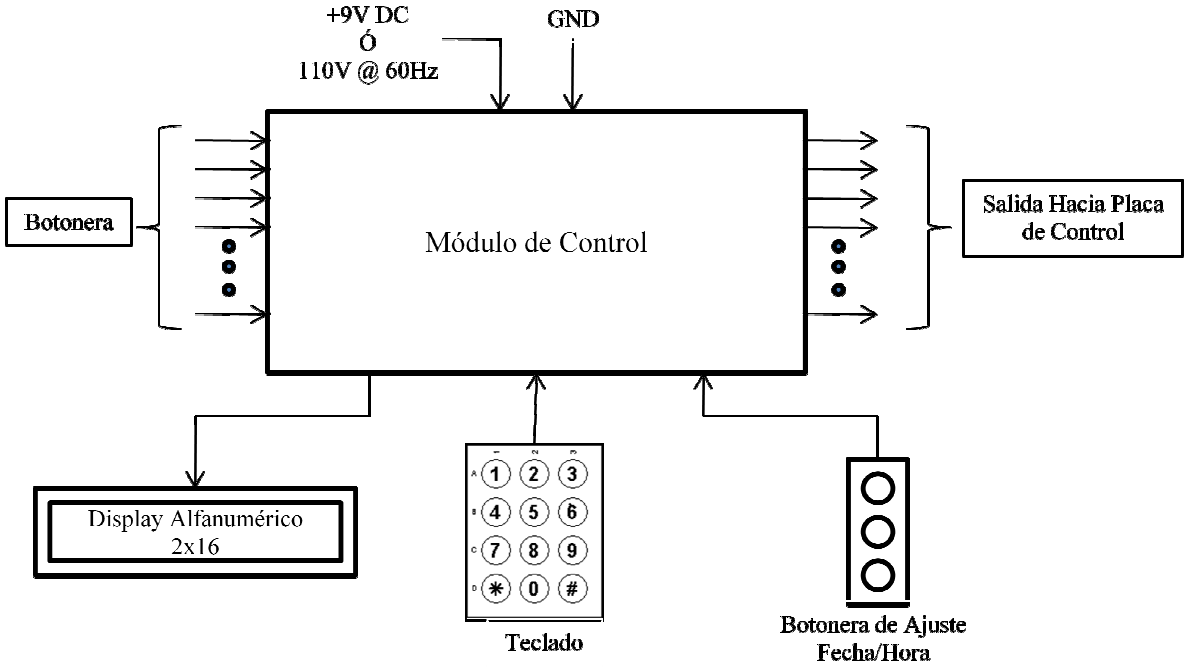

Figura 1: Diagrama de Bloques del Hardware del Sistema.

3.1.1 Módulo de Control

El módulo de control posee una serie de dispositivos e interfaces periféricas de uso más frecuente en la implementación de un proyecto cualquiera. Dispone de un microcontrolador maestro de la familia PIC18, entradas/salidas digitales y osciladores para el microcontrolador, una memoria EEPROM externa y un reloj de tiempo real (RTC); contiene además una serie de Shift Register, y adicionalmente se ha habilitado el puerto I2C y la interrupción externa del microcontrolador. El RTC así como la EEPROM externa hacen uso del puerto I2C habilitado. El microcontrolador puede ser programado mediante el uso de un conector destinado para tal fin. La Figura 2 muestra el diagrama de bloques del módulo de control.

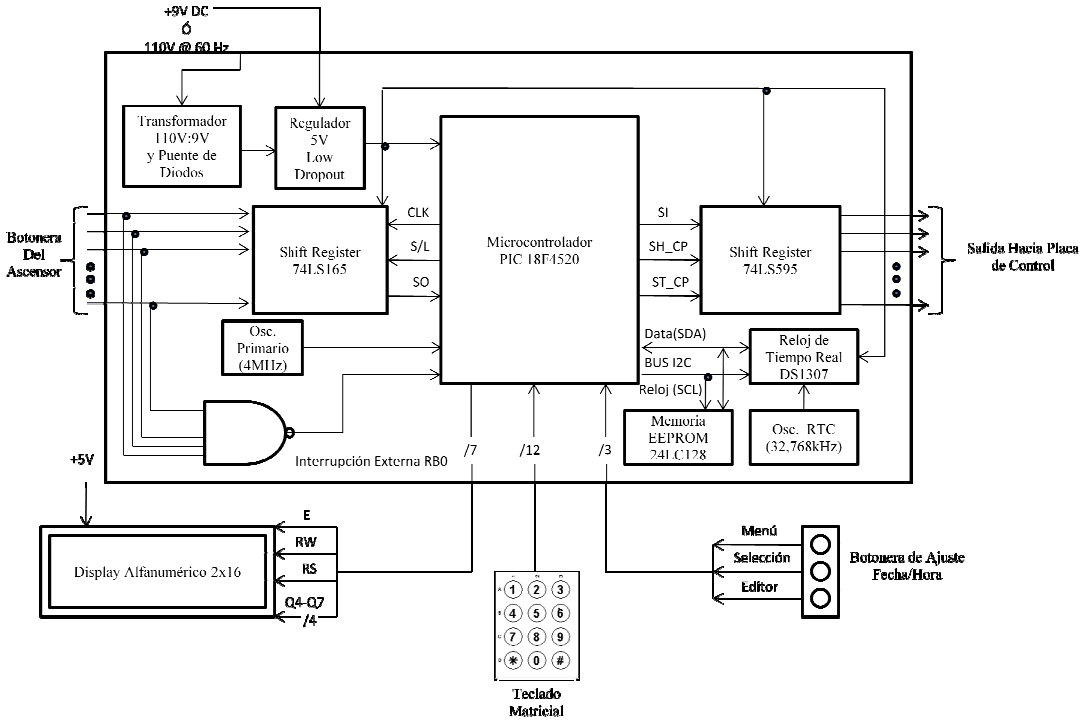

Figura 2: Diagrama de Bloques del Módulo de Control.

3.1.1.1 Microcontrolador PIC18

Se ha dispuesto de un microcontrolador PIC18F4520 de la compañía Microchip como elemento de control del sistema. La arquitectura de los microcontroladores pertenecientes a esta familia sigue el modelo Harvard de 8 bits, cuya tecnología RISC y tecnología CMOS lo distinguen de los demás. Debido a estas características este controlador es altamente efectivo en el uso de memoria de datos y programa, y por lo tanto en velocidad de ejecución. La Figura 3 muestra la distribución de los pines del microcontrolador. Dicho controlador ha sido programado mediante el dispositivo Pickit 2 de la casa Microchip.

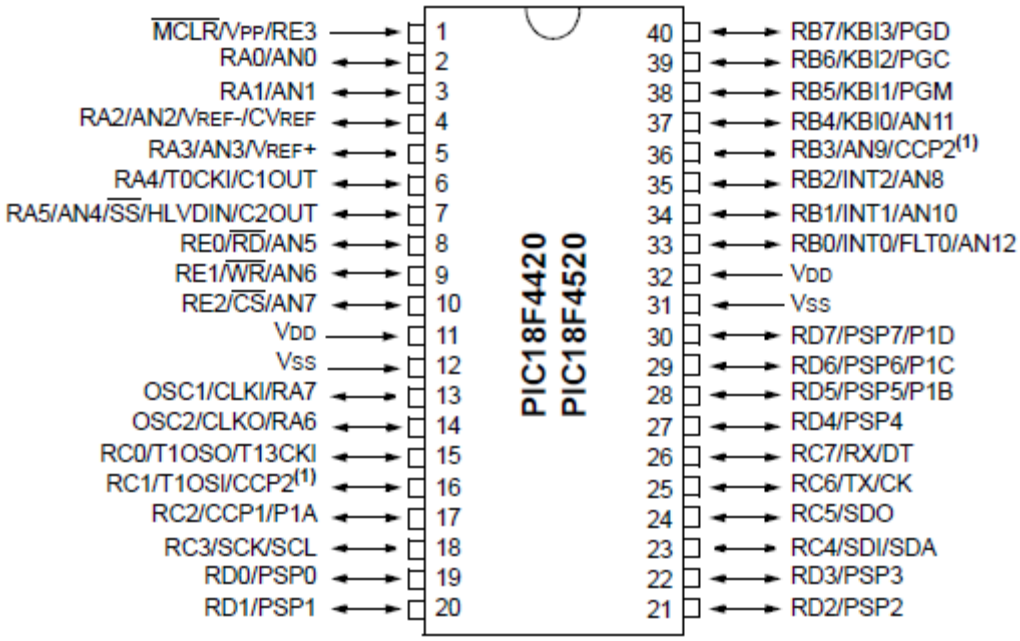

Figura 3: Distribución de pines del microcontrolador PIC18F4520.

Características Generales:

- Memoria Flash de programa de 32 kBytes agrupados de a 2 Bytes.
- Memoria de datos (RAM) de 1536 x 8 Bytes.

- Memoria de datos EEPROM de 256 x 8 Bytes.
- Stack de hardware de treinta y un (31) niveles.
- Protección programable de código.
- Posee veinte (20) fuentes de interrupción, con niveles de prioridad.
- Cuatro (4) timers, uno (1) de 8 bits y tres (3) de 16 bits.
- Desempeño de hasta 10 MIPS (Millones de Instrucciones Por Segundo).
- Alimentación de 2.0 a 5.5 Volt.

3.1.1.1.1 Estructura Interna

El microcontrolador PIC18F4520 posee la memoria de programa y la memoria de datos en forma independiente, a esto se le conoce como arquitectura Harvard. Este tipo de configuración tiene, entre otras cosas, la ventaja de acceder a las instrucciones de programa y de datos simultáneamente mediante buses diferentes, lo que mejora considerablemente la velocidad de ejecución de estos dispositivos.

3.1.1.1.2 Unidad de Control (CPU)

La unidad de control del microcontrolador PIC18F4520 presenta una arquitectura tipo Harvard de 8 bits, la cual procesa instrucciones a una rata que supera los 10MIPS a una frecuencia de trabajo de 40MHz. Este microcontrolador emplea por cada ciclo de instrucción cuatro (4) ciclos de reloj para ser ejecutadas, salvo las instrucciones de salto fijo o condicional, las cuales son ejecutadas en ocho (8) ciclos de reloj. Cada ciclo de instrucción la CPU lee la instrucción guardada en la memoria de programa apuntada por el registro Program Counter (PC), y simultáneamente ejecuta la instrucción anterior.

Este dispositivo presenta un registro de estados (STATUS) de 8 bits, donde cada bit (denominado bandera) es un indicador de estado de la CPU o bien del resultado de la última operación ejecutada.

El PIC18F4520 es un microcontrolador RISC (Reduced Instruction Set Computer), lo cual se refleja en su reducido repertorio de 75 instrucciones ortogonales, pudiendo extenderse a 83 si la opción “Set de Instrucciones Extendida” es habilitada, siendo las mismas rápidas en ejecución con una longitud fija de 16 bits.

3.1.1.1.3 Entradas/Salidas

Los pines de Entrada/Salida del microcontrolador PIC18F4520 presentan variedad de funcionalidades que varían según los requerimientos de diseño de acuerdo a buses de comunicación, comparadores, entradas del tipo analógica o digital, entre otras funciones. Estos pines manejan corrientes de hasta 25mA por pin.

El puerto B de este microcontrolador permite habilitar mediante software una serie de resistencias de PULL-UP, las cuales entre sus funciones permiten mejorar la conexión con interfaces externas, manejar niveles mayores de corriente, mejorar el comportamiento en el rango de temperaturas, mayor resistencia al ruido, entre otras.

3.1.1.1.4 Buses de Comunicación

El microcontrolador PIC18F4520 dispone de una serie de interfaces de comunicación para periféricos externos. A continuación se da mención de los puertos habilitados para este diseño así como los dispositivos que estos manejan:

- I2C (Inter-Integrated Circuit). Puerto I2C para comunicación con reloj de tiempo real (RTC) externo y memoria EEPROM externa.
- PSP (Parallel Slave Port). Puerto PSP para adquisición de data proveniente de teclado y control de pantalla LCD.

3.1.1.1.5 Interrupciones del Microcontrolador

El microcontrolador PIC18F4520 posee veinte (20) fuentes posibles de interrupción programables, con disposición de ocho (8) niveles de prioridad. Cuando ocurre alguna de las fuentes de interrupción asociadas a nivel alto, el vector de interrupciones se ubicará en la posición 0008h, y si la prioridad de la interrupción es baja, el salto producido se hallará en la posición 0018h.

Para este propósito es habilitada únicamente la interrupción externa RB0 cuyo pin es conectado a un dispositivo 74S133 acoplado a su vez a la botonera del ascensor, a fin de verificar la tecla presionada sin coincidir con el código de muestra de hora y fecha proveniente del RTC externo.

3.1.1.1.6 Interfaz de Programación del Microcontrolador

La interfaz de control posee un conector del tipo convencional (pines) de 5 terminales para la programación del microcontrolador. Para realizar la programación del microcontrolador empleado son utilizadas las señales de alimentación (VCC y GND), la señal de reset MCLR y las señales de bus de comunicación PGD y PGC correspondientes a datos y reloj, respectivamente. Por medio de un programador adecuado (Programador Pickit 2 de la compañía Microchip) es posible acceder a la memoria de códigos del dispositivo.

3.1.1.2 Interfaz HMI: Conjunto Teclado-Pantalla

La interfaz hombre-máquina implementada para el diseño del proyecto consiste en dos dispositivos de entrada (a.- Teclado matricial de 4 filas por 3 columnas y b.- Teclado de 3 pulsadores) y uno de salida (Pantalla alfanumérica de 2 filas por 16 columnas) los cuales permiten el ingreso y visualización de configuración de piso y su respectivo código de seguridad u hora de acceso a ser guardados en la

memoria EEPROM externa, así como el ingreso de código de seguridad para el acceso hacia algún piso en particular si el caso lo amerita (teclado “a”). A su vez admite la ejecución de configuración para hora y fecha del RTC externo, como también el desplazamiento en el menú de opciones presentes en el diseño (teclado “b”).

El teclado matricial se encuentra conectado a cuatro (4) pines del microcontrolador (RD4-RD7), los cuales exponen las filas de este a niveles altos de tensión (“1” lógico). A su vez mediante otros tres (3) pines de entrada (RD3, RC5 y RC6) asociados a las columnas del teclado se consigue leer la muestra de una tecla presionada, realizando un barrido de dichas columnas. Cuando una tecla es pulsada se conecta una fila con una columna, lo que permite calcular su posición relativa para así obtener una muestra de la tecla presionada.

Por medio de la librería KBD_2.C dispuesta en el anexo 1 se confirman las teclas que han sido presionadas para ser tratadas por la aplicación. Algunos de estos pines del microcontrolador son compartidos entre el teclado matricial y la pantalla LCD, de forma tal que las señales correspondientes a la data transmitida hacia la LCD y a la exposición de las filas del teclado matricial hacia niveles lógicos altos (1 lógico) son correctamente multiplexadas. La Figura 4 muestra las conexiones asociadas al teclado matricial y a la pantalla LCD utilizada.

Por su parte, el teclado de 3 pulsadores se encuentra conectado a VCC en uno de sus extremos mientras que al otro extremo se ubica cada pulsador a un pin de entrada del microcontrolador utilizando para este propósito tres (3) de éstas entradas, cuyas funciones están asociadas a Menú, Enter y Editor, las cuales realizan el desplazamiento entre menús de pantallas o cancelación de aplicación (de acuerdo al modo ejecutado), entrada hacia menú, y edición de datos seleccionados, respectivamente. La Figura 5 muestra las conexiones asociadas al teclado de tres (3) pulsadores.

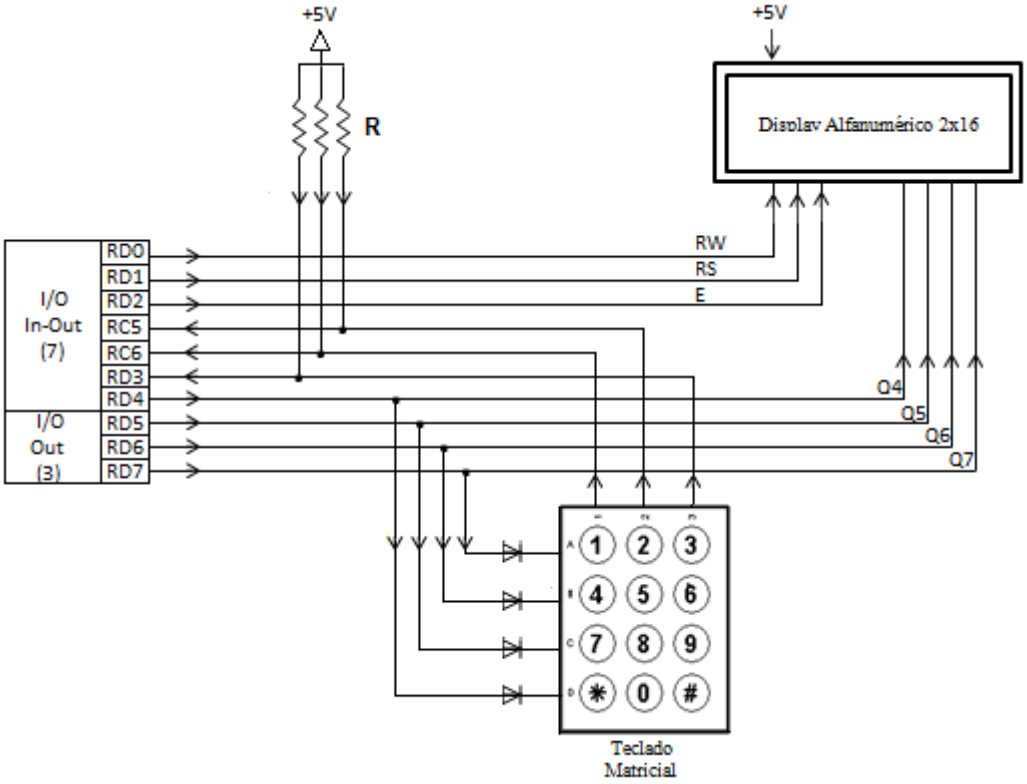

Figura 4: Conexiones del Teclado Matricial y Pantalla LCD para 4 bits de puerto.

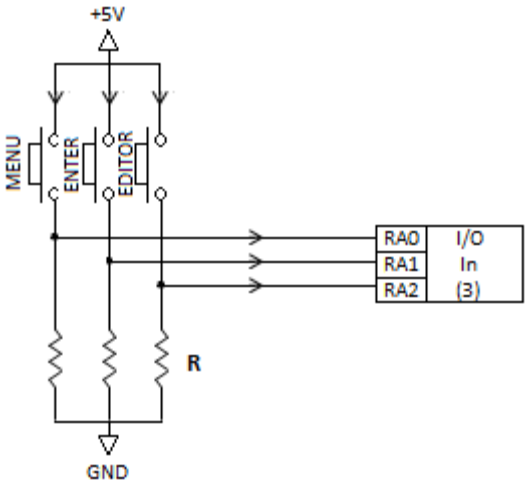

Figura 5: Conexiones del Teclado de 3 Pulsadores.

La pantalla implementada en la aplicación de la interfaz está basada en una programación a manera de utilizar un bus de datos de 4 bits, modo de trabajo en el cual la transferencia de información se efectúa por medio de los cuatro (4) bits más significativos, con tres (3) señales de control adicionales: Selección de Registro (RS), Lectura/Escritura (RW) y Habilitación de Comando (E). La señal RS permite seleccionar dato o comando para escribir o leer memoria de pantalla (presentación de caracteres de pantalla) o memoria de caracteres especiales (8 primeros caracteres del set). La señal RW permite la lectura de cursor o datos, y escritura o presentación de caracteres en pantalla. La señal E permite habilitar la ejecución de alguna instrucción, previamente programados los pines correspondientes a RS y RW.

3.1.1.3 Reloj de Tiempo Real (RTC) Externo

Para mostrar en pantalla la hora y fecha de forma actualizada en la pantalla alfanumérica LCD se dispone de un reloj de tiempo real (RTC) DS1307 desarrollado por Dallas Semiconductors, con 56 bytes de memoria RAM no volátiles cuya interfaz con el microcontrolador se realiza utilizando el bus I2C.

El DS1307 es un reloj-calendario en código decimal binario (BCD), programado para mantener la cuenta de los segundos, minutos, horas, fecha, mes y año, con compensación de años bisiestos cuya finalidad es mantener informado al microcontrolador del momento de ocurrencia de alguna actividad secundaria tal como dar acceso hacia algún piso de acuerdo a horario preestablecido. Este dispositivo se encuentra configurado para formato de 24 horas, y éste es conectado a una batería de 3 Volt (fuera de su alimentación principal) a fin de mantener los datos de hora y fecha aun cuando el resto del sistema se encuentre inoperativo o desenergizado. La Figura 6 muestra las conexiones del RTC externo.

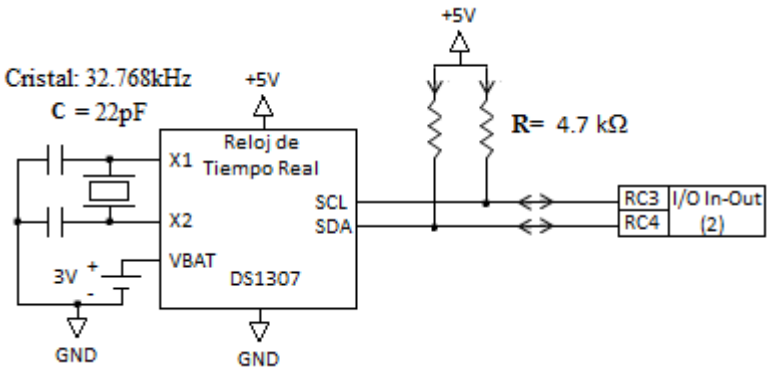

Figura 6: Conexión del RTC.

3.1.1.4 Memoria EEPROM Externa

Para memorizar los datos de hora y clave de acceso de un determinado número de pisos que deben quedar permanente registrados se dispone de una memoria EEPROM 24LC128 desarrollada por Microchip, con 128 kbits de memoria RAM no volátiles cuya interfaz con el microcontrolador se realiza utilizando el bus I2C.

Esta EEPROM 24LC128 es una memoria PROM borrable, pero a diferencia de ésta, se puede borrar mediante la aplicación de una pequeña corriente eléctrica. La misma permite la lectura y escritura de palabra por palabra, lo que la hace aun más práctica que el resto de las memorias. La EEPROM 24LC128 permite conexiones de hasta ocho (8) dispositivos semejantes en cascada, trabaja a una frecuencia máxima de reloj de 400kHz, permite almacenar datos por un periodo de 200 años y posee protección de escritura (WP), lo que evita errores de escritura por ruidos u otros entes de diversa índole.

La utilización de este dispositivo radica en el almacenamiento y retención de horas y códigos para el acceso hacia una serie de pisos destinados para tal fin, de forma tal que estos datos no sean eliminados en caso de ausencia de electricidad o

cuando el microcontrolador utilizado sea reprogramado. La Figura 7 muestra las conexiones de la memoria EEPROM externa.

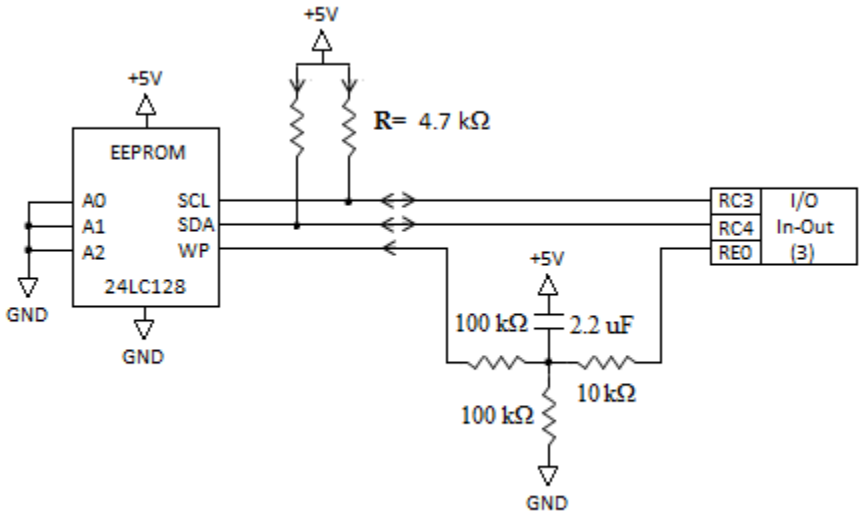

Figura 7: Conexiones de la EEPROM 24LC128.

3.1.1.5 Shift Register de Entrada

Para el manejo y control de diversas entradas provenientes de la botonera de cabina haciendo uso de tan sólo tres (3) pines del microcontrolador se dispone de dos (2) 74LS165, dispositivo TTL (Transistor Transistor Logic) el cual es un registro de desplazamiento (Shift Register) de 8 bits con acceso paralelo de ocho (8) entradas de datos individuales y salida de datos serial (SO). Este dispositivo además de las ocho (8) entradas de datos paralelas dispone de una (1) entrada del tipo serial la cual le permite hacer conexiones en cascada a fines de aumentar el número de pisos involucrados y una (1) salida serial negada; este es controlado mediante tres (3) señales adicionales: Reloj (CLK), Inhibidor de Reloj (INH) y Habilitador de Carga (S/L).

La señal S/L permite cargar los ocho (8) bits de entrada paralela mientras el pin de la señal INH se encuentra en alto, tiempo en que la señal CLK no afecta a la salida, una vez que el pin correspondiente a la señal INH es puesto en bajo, la entrada paralela saldrá en serie en orden D0 a D7 por ambas salidas.

Cada pin de entrada de estos dispositivos se encuentra conectada al cátodo de un diodo 1N4007 debido a que las señales provenientes de la botonera del ascensor son activadas con nivel de tensión bajo (“0” lógico). La Figura 8 muestra las conexiones de los Shift Register 74LS165.

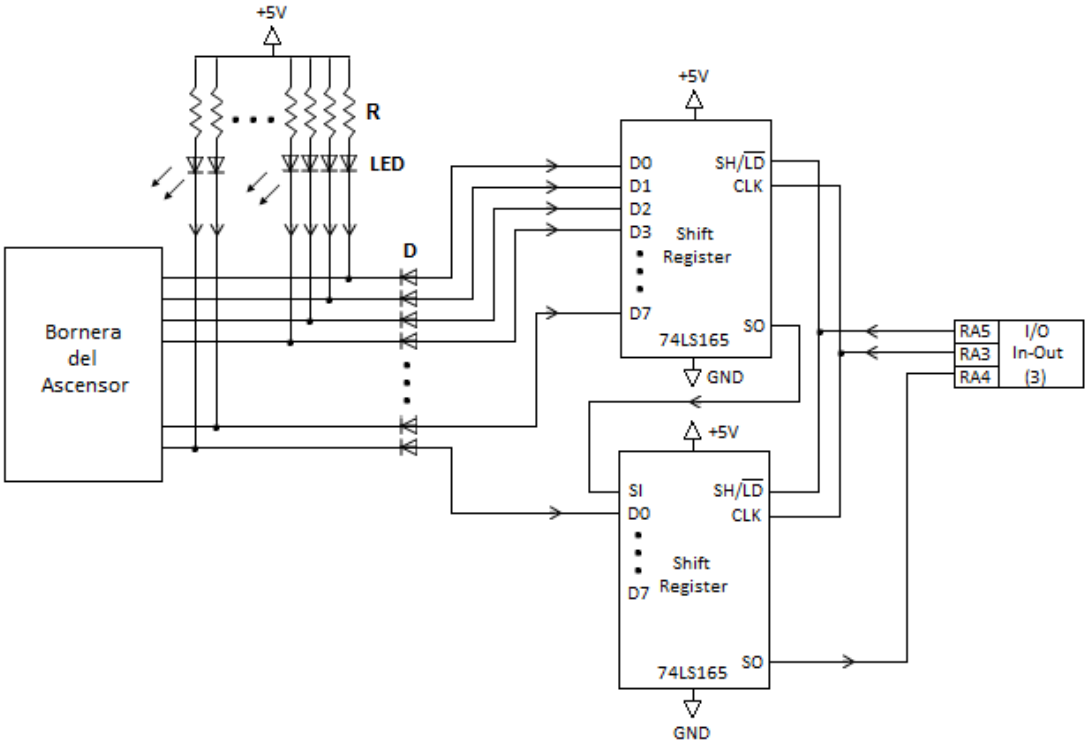

Figura 8: Conexiones de los Shift Register 74LS165.

3.1.1.6 **Shift Register de Salida**

Para el manejo y control de diversas salidas dirigidas hacia la tarjeta de control principal del ascensor haciendo uso de tan solo tres (3) pines del microcontrolador se dispone de dos (2) 74LS595, dispositivo TTL el cual es un registro de desplazamiento de 8 bits cuya entrada es del tipo serial (SI), y salida paralela o serial de datos individuales, el cual se puede expandir para añadir más pisos a los que se desee llamar. Este dispositivo además de las ocho (8) salidas de datos paralelas dispone de salida tres estados latchedas, lo que quiere decir que la misma puede ser de nivel alto (“1” lógico), de nivel bajo (“0” lógico) o de alta impedancia. El mismo es controlado mediante dos (2) señales adicionales: Reloj de Desplazamiento (SH_CP) y Reloj de Cierre de Dato (ST_CP).

Con cada pulso de la señal SH_CP se envían los datos de forma serial a través de los diferentes registros de desplazamiento para ser transmitidos, mientras que un impulso sobre la señal ST_CP escribe estos datos a través de la memoria intermedia del 74LS595 en el registro de salida, acoplando el valor que se desea transmitir a dicha salida. La Figura 9 muestra las conexiones de los Shift Registers 74LS595.

3.1.1.7 Módulo de Alimentación

La interfaz de control dispone de dos modos de alimentación, el primero es por medio de una batería cuyo rango de tensión se encuentre entre los 9 y 12 V preferiblemente, y el segundo que admite alimentación desde la línea eléctrica (110V @ 60Hz) a través de un transformador reductor del tipo 110V:9V, seguido de un puente rectificador de onda completa, para luego ser conectados a un regulador en cuya salida admitirá una tensión de 5 V. La Figura 10 muestra el diagrama de bloques del circuito de alimentación.

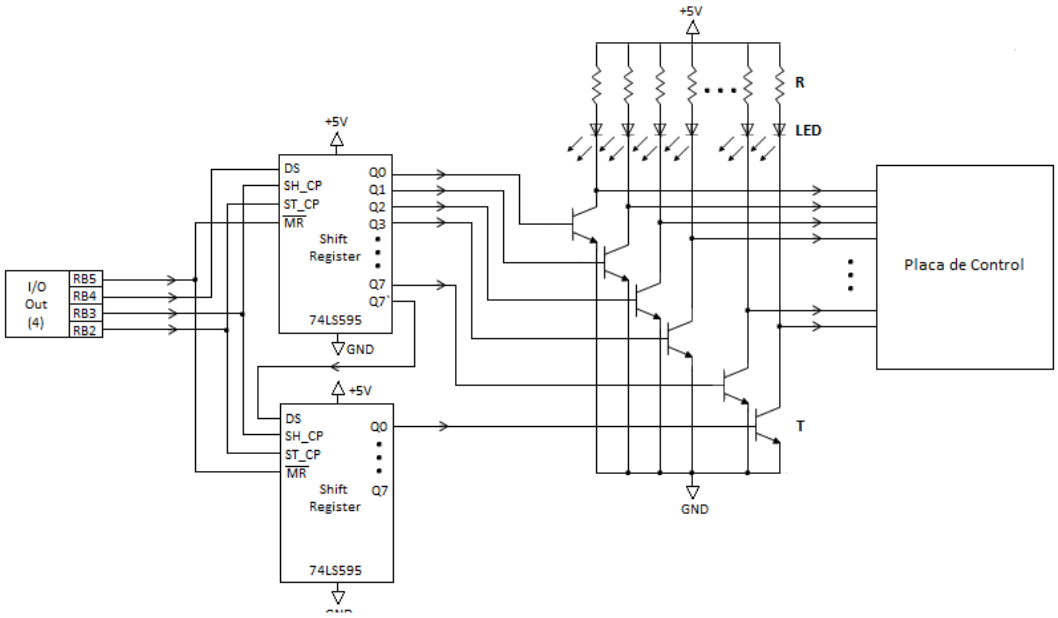

Figura 9: Conexiones de los Shift Register 74LS595.

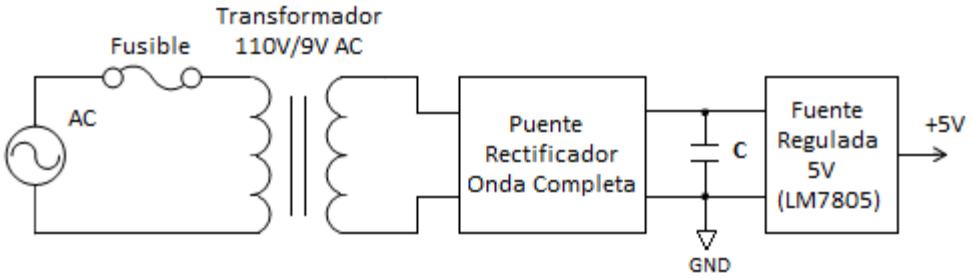

Figura 10: Diagrama de Bloques del Circuito de Alimentación.

3.1.2 Descripción General de La Interfaz

La interfaz en cuestión consiste en un sistema de control ubicado entre la botonera de cabina de un ascensor y su tarjeta principal de control, cuya aplicación se encuentra diseñada y elaborada para el público en general de manera que permita adaptar funciones extras al ascensor sin intervenir en la ejecución de los servicios para el cual fue confeccionado. Dicha interfaz debe montarse en un cajetín ubicado en

la cabina propiamente dicha, como se estila para cualquier ascensor público convencional y su funcionamiento debe ser independiente de la programación que presente la tarjeta maestra del mismo. Sin embargo, el software asociado al proceso de ejecución de llamadas de cabina debe adaptarse a los requerimientos funcionales del ascensor, adoptando así el estándar con el cual el ascensor fue programado.

La interfaz de control constará básicamente de un sistema compuesto por un teclado que concederá ajustar fecha y hora, un teclado que permitirá introducir código para dar acceso a una serie de pisos gestionados para tal fin y adicionalmente una pantalla alfanumérica donde se mostrarán la hora y fecha actuales o bien una serie de menús para el ajuste de parámetros o ejecución de datos concernientes a la aplicación que en el momento se encuentre seleccionada.

En primera instancia, la hora y fecha actuales deben ser colocadas por medio del teclado destinado para ello, dado que la interfaz se encuentra diseñada para operar con tales datos, de lo contrario su operación no será ejecutada de manera idónea a su elaboración. Una vez calibrado, la interfaz permitirá dar ingreso a códigos y horas de acceso para algún piso seleccionado, lo cual agregará seguridad a dichos niveles. Una vez realizado esto, la interfaz en conjunto con el ascensor se encuentran disponibles para operar.

Las llamadas de cabina pueden efectuarse mediante el marcado de un botón de la botonera, los cuales corresponden a la serie de pisos que presenta la edificación en la cual el ascensor se encuentra operando. Para realizarse una llamada de cabina, el usuario debe pulsar el botón correspondiente al piso al cual desea dirigirse. Posteriormente, la interfaz procesará dicha petición permitiendo convenientemente dar acceso a dicho piso de acuerdo a una lógica pre-programada.

Para activarse la llamada, la interfaz a de capturar el botón marcado a fin de verificar si el piso al cual se desea ingresar posee hora y/o código de acceso. En

primera instancia el sistema comprobará si el piso al cual se desea ingresar posee hora de acceso programada, y de ser así la interfaz se encargará de verificar y comparar los datos de hora actual con los de horas programadas de forma tal que se pueda o no acceder en ese momento. Por el contrario, de no poder ingresar debido a no cumplir con las horas de acceso permitidas, será mostrado un mensaje en la pantalla LCD indicando que no es concedido el acceso hacia el piso deseado.

Seguidamente, de ser aprobado el acceso de acuerdo a la hora, el sistema se encargará de comprobar si el piso presenta código para su ingreso, y de tenerlo, la pantalla alfanumérica mostrará un mensaje de petición de código con lo que la interfaz esperará el marcado del mismo. Si el código de seguridad ingresado por el usuario es incorrecto se mostrará en la pantalla un mensaje de acceso denegado al usuario, rechazando de esta manera dicha petición. Si por el contrario el código ingresado por el usuario es correcto, la pantalla mostrará un mensaje de acceso concedido, el cual permitirá realizar dicha llamada. Posteriormente, concedido o no el acceso hacia el piso seleccionado, la pantalla volverá a mostrar de manera actualizada los datos de hora y fecha permitiendo dar opción a realizar una nueva llamada.

Una vez aprobado el acceso hacia el piso marcado, la interfaz autorizará el envío de una señal a la tarjeta principal de control la cual se encargará de ejecutar la solicitud de llamada de cabina introducida, pudiendo optar posteriormente por realizar alguna otra llamada. Si la llamada solicitada posee clave de acceso programada o bien si por error fue presionado un piso que si la posee, dicha llamada puede ser cancelada pulsando la tecla programada para tal fin, permitiendo realizar una nueva llamada desde la cabina del ascensor.

Por otra parte, la interfaz permitirá realizar cambios de código para el acceso hacia los pisos programados para tal fin. Al acceder a este modo se mostrará en pantalla una petición del piso al cual se desea cambiar la clave de acceso, posteriormente la pantalla mostrará una petición de código de seguridad (de tenerla

previamente programada), de ser confirmada la clave conferida por el usuario se mostrará en la LCD una petición de ingreso de nuevo código y una vez ingresado dicho código se pedirá de nuevo el ingreso del mismo para verificación y aceptación de la petición. Una vez realizada esta operación, la interfaz volverá al menú principal para seguir ejecutando las operaciones para la cual fue diseñada.

A su vez la interfaz diseñada posee la opción de ingreso de horas de acceso a piso, permitiendo de ésta manera dar ingreso a un determinado piso sólo en horarios pre-configurados. Al acceder a este modo se mostrará por medio de la pantalla alfanumérica una petición de código de seguridad (de tenerla previamente programada), de ser confirmada la clave conferida por el usuario se mostrará en la LCD una petición de ingreso para la hora de inicio y confirmada dicha hora la petición siguiente será el ingreso de una hora final, a fin de que el acceso hacia dicho piso sea concedido sólo en el horario comprendido entre tales horas programadas. De existir algún error por parte del usuario al introducir las horas solicitadas, las mismas pueden ser canceladas al presionar un botón configurado para ejecutar tal acción. Una vez realizada esta operación, la interfaz volverá al menú principal para seguir ejecutando las operaciones para la cual fue diseñada.

La interfaz además de poseer opción a ingreso de horas y códigos de acceso, permite el borrado de cualquiera de los datos que confieren seguridad al piso. El ingreso a estos modos se admite por separado, pero ambos permiten la supresión de tales datos de forma similar. Al acceder a cualquiera de estos modos se mostrará por medio de la pantalla alfanumérica una petición de código de seguridad (de tenerla previamente programada), de ser confirmada la clave conferida por el usuario se mostrará en la LCD un mensaje indicando que la clave o las horas que el piso tenía programadas fueron eliminadas. Una vez realizada esta operación, la interfaz volverá al menú principal para seguir ejecutando las operaciones para la cual fue diseñada.

Además, el sistema posee una funcionalidad de identificación personal por piso (servicio privado). Tras ser activado el sistema en esta modalidad, continuamente la interfaz se encontrará mostrando mediante la pantalla de caracteres los datos asociados a fecha y hora de forma actualizada, y de existir alguna llamada, se mostrará por medio de dicha pantalla una petición de código. Una vez ingresado éste, el sistema verificará el código y de ser aceptado, procederá a verificar la existencia o no de alguna hora de ingreso programada, de no existir o bien de ser permitido el acceso en ese instante, la interfaz realizará la llamada hacia el piso asociado a dicho código suministrado. Por el contrario, de poseer una hora de restricción el sistema no efectuará dicha solicitud de acceso, caso análogo al de haber introducido una clave no existente, salvo que para este caso será mostrado en el display un mensaje de clave no existente. En cualquiera de los modos al que se haya ingresado es posible su cancelación por medio de una tecla configurada para esa finalidad.

La interfaz de control podrá ser alimentada por defecto a través de una batería cuya tensión se encuentre entre los 9 y 12 V DC o en su defecto por la línea eléctrica convencional (110V @ 60Hz). Dicha interfaz deberá tener todas las protecciones pertinentes contra rigidez mecánica, rigidez eléctrica y de interferencia electromagnética de manera tal que no genere daños a la tarjeta principal de control ni demás componentes asociados al ascensor.

A diferencia con la tarjeta de circuito impreso que fue diseñada y ensamblada en la empresa, los dispositivos y componentes utilizados para la implementación de la interfaz de control tales como teclados, pantalla LCD y demás han sido adquiridos mediante proveedores externos, cuyas casas poseen experiencia en la manufactura de los mismos.

Cada uno de los ajustes previamente mencionados son realizados únicamente por un técnico encargado de la instalación de la correspondiente interfaz en un determinado ascensor.

3.1.3 Módulos de Aplicación

La botonera de cabina y la tarjeta de aplicación o tarjeta maestra se componen de una serie de elementos y dispositivos que los caracterizan de acuerdo a su aplicación, diferenciándolos de cualquier otro producto que se desee implementar o desarrollar. La botonera de cabina es el dispositivo primordial para establecer una comunicación directa entre el usuario y el ascensor.

Este módulo está diseñado mediante sistemas electrónicos, encargados para hacer funcionar, mediante el uso de una serie de botones, la dirección de movimiento de la cabina, la selección de pisos en los que ésta deba detenerse y a su vez permitir la ejecución de una serie de maniobras de control como lo son la apertura y cierre de puertas de cabina, seguridad, ventilación interna, luz interna de cabina, alarma y parada de emergencia. Por su parte, la tarjeta maestra es la encargada de ejecutar todas las operaciones correspondientes a los eventos asociados al ascensor, y ésta dispone de una serie de conexiones de entrada correspondientes con la botonera de cabina (o botonera de pisos) cuya utilidad es establecer una conexión entre la interfaz a diseñar y la propia tarjeta maestra.

Los puertos de entrada en la tarjeta principal del ascensor son activos con un cero lógico “0” lo que implica realizar una conexión entre tierra (“0” Volt) y el pin correspondiente al piso que se desea ingresar. La Figura 11 muestra la botonera de cabina y seguidamente en la Figura 12 se muestra el esquema de entradas/salidas correspondiente a la tarjeta maestra.

Figura 11: Botonera de Cabina del Ascensor.

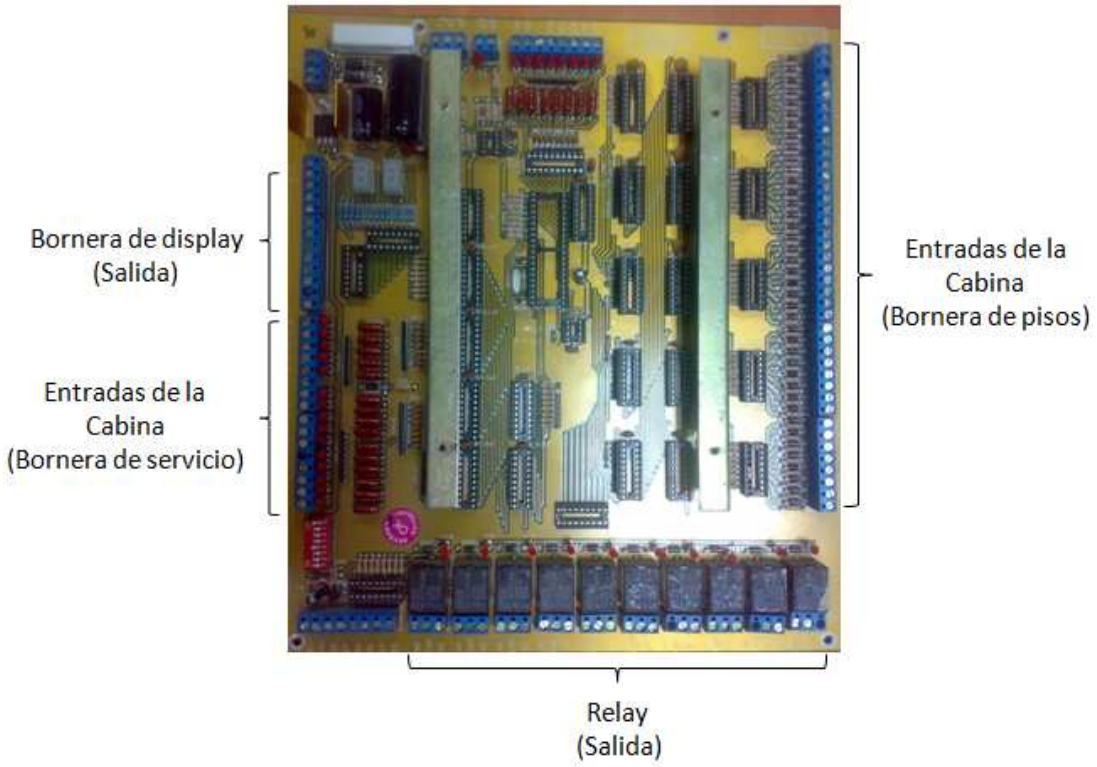

Figura 12: Esquema de Entradas/Salidas de la Tarjeta Maestra

- Botonera de display: Este se encuentra compuesto por ocho (8) terminales de salida los cuales permiten la transmisión de señales relacionadas a la posición o piso actual en el que se encuentra ubicado el ascensor, para ser mostrado por medio de un display siete segmentos (generalmente).

- Botonera de servicio: Ésta comprende los siguientes bornes de entrada (Se activan con un “1” lógico):

  - MAN: Manual/Automático.
  - BOM: Bomberos.
  - IND: Servicio independiente.
  - CER: Pulsador cerrar puerta.
  - ABR: Pulsador abrir puerta.

- ALT: Alarma por temperatura PTC Motor (no considerar).
- CMP: Completo.
- SBR: Sobrecargado.

- Relay: De izquierda a derecha representan las siguientes señales de salida:

  - CSU: Maneja contactora de subir.
  - CSD: Maneja contactora de bajar.
  - RAV: Maneja contactora de alta velocidad.
  - RBV: Maneja contactora de baja velocidad.
  - POTR1: Relay de potencia.
  - PSU: Flecha subir.
  - PDE: Flecha bajar.
  - V2: Velocidad Intermedia.
  - RAP: Relay de abrir puerta DIP 78=1,2,3,4 tiempo de puerta abierta (se selecciona únicamente uno).
  - RCP: Relay de cerrar puerta.

- Botonera de pisos: Éstas son las señales provenientes del interior de la cabina, las cuales son activadas con un “0” lógico, o bien una tensión de cero volt. Las señales se encuentran dispuestas desde arriba hacia abajo en la placa F40.

  - LC1: PB.
  - LC2: Piso 1.
  - LC3: Piso 2.
  - LC4: Piso 3.
  - LC5: Piso 4.
  - LC6: Piso 5.
  - LC7: Piso 6.
  - LC8: Piso 7.

- LC9: Piso 8.
- LC10: Piso 9.

CAPÍTULO IV

4.1 METODOLOGIA DE PROGRAMACIÓN

4.1.1 Descripción General

La función principal del programa es capturar los datos provenientes de la botonera ubicada en la cabina del ascensor y procesarlos de forma conveniente para su posterior envío a la tarjeta maestra.

De forma general la interfaz diseñada después de activada mostrará, por medio del display LCD, un menú teniendo como primera opción el ajustar hora y fecha a fin de obtenerlas de manera actualizadas, seguidamente se presenta una segunda opción la cual permitirá el ingreso de código de acceso hacia algún piso determinado para tal modalidad, posteriormente se mostrará en el display una tercera opción la cual presentará la función para ingreso de horas de acceso, luego como cuarta opción se presenta el modo que permitirá el borrado del código de acceso de algún piso, consecutivamente será mostrada una quinta opción cuya función radica en el borrado de las horas de acceso programadas en un piso determinado, y finalmente una sexta opción la cual permitirá iniciar el funcionamiento de la interfaz diseñada.

Una vez aceptada la opción de inicio del programa principal, la interfaz estará ejecutando paralelamente la muestra de hora y fecha de forma actualizada por medio de una LCD, escaneo de petición de ajuste de datos para acceso hacia algún piso, y realizando un escaneo constante en los pines asociados a la botonera de cabina a fin de verificar la existencia o no de alguna llamada, y de existir alguna, la ejecutará de acuerdo a la comparación con los parámetros de permiso para acceso a ese piso según horario y por medio de clave de acceso al piso seleccionado. Ejecutadas estas

premisas la interfaz enviará convenientemente una señal a la tarjeta maestra en función de la data procesada permitiendo convenientemente la ejecución de la llamada en cuestión. Una vez realizada tal operación, la interfaz estará lista para procesar una nueva llamada. La Figura 13 muestra un diagrama de flujo correspondiente con el programa general.

4.1.2 Estructura de Tareas del Sistema

Para el desarrollo de ésta interfaz se definieron ocho (8) etapas, las cuales son:

- Manejo de Hora y Fecha (DS1307).
- Ingreso de Clave de Acceso.
- Ingreso de Horas de Acceso.
- Borrado de Clave y Horas de Acceso.
- Entrada de Datos de Botonera (Llamadas).
- Salida de Datos Hacia Tarjeta Maestra.
- Manejo de Pantalla.
- Servicio Privado.

4.1.2.1 Manejo de Hora y Fecha (DS1307)

Para el manejo o ajuste de hora y fecha asociados al reloj de tiempo real implementado en el sistema, debe ser previamente seleccionado el modo “Ajustar Hora y Fecha” del menú principal mostrado en pantalla, a fin de solicitar su ingreso.

Una vez ingresado a éste modo la pantalla mostrará los datos de hora y fecha almacenados por defecto, y de forma intermitente será mostrado en el display el dato correspondiente a los segundos de modo que pueda ser modificado con el uso de la tecla “EDITOR” del teclado de tres pulsadores. Así mismo, al presionar “MENU” del mismo teclado se accederá al modo de modificación del dato correspondiente a los

minutos, mostrándose de forma similar al dato anterior, es decir, de manera intermitente.

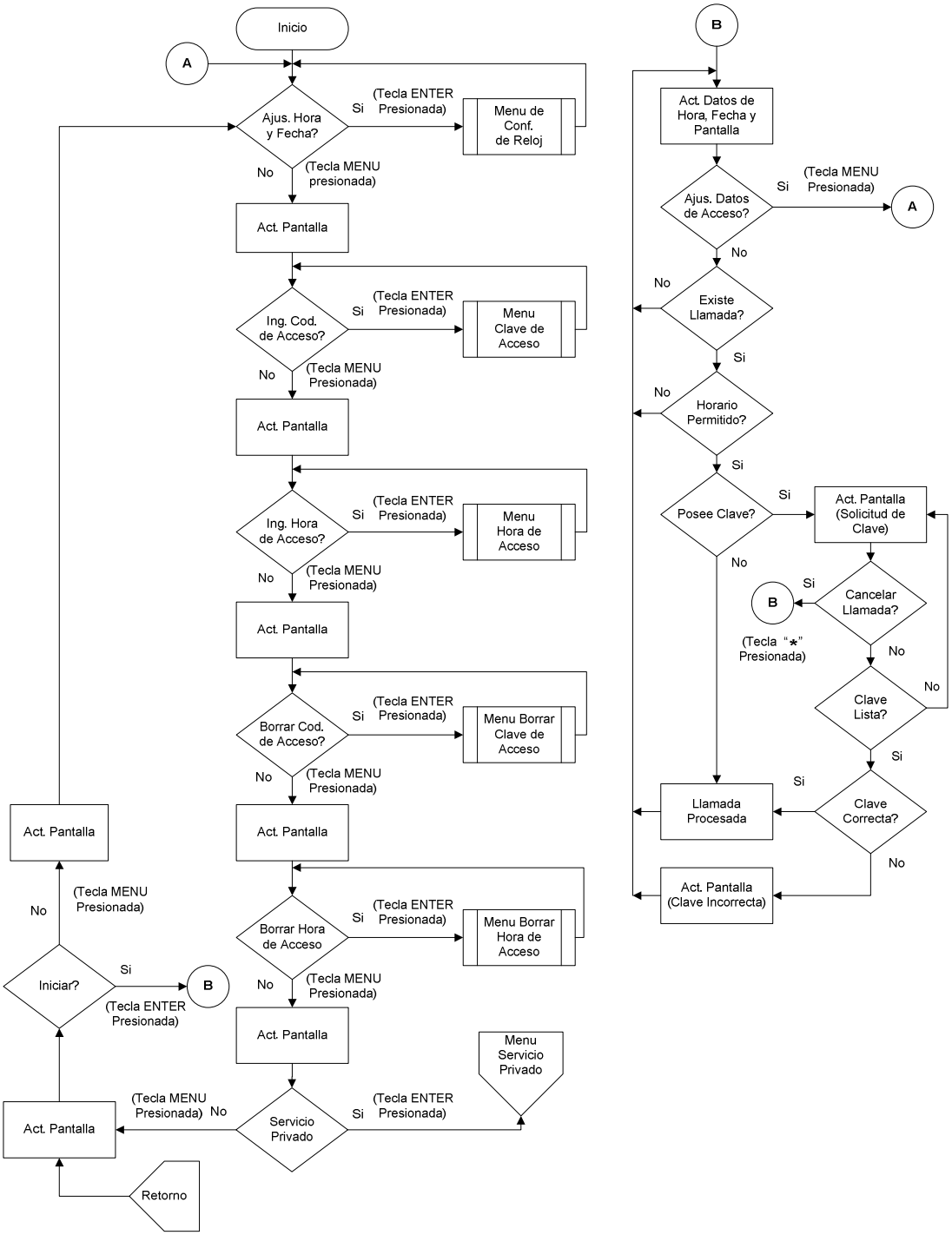

Figura 13: Diagrama de Flujo del Programa Generalizado.

En ésta modalidad de configuración de fecha y hora, cada uno de los datos asociados pueden ser configurados de la misma forma haciendo uso de los teclas “MENU” y ”EDITOR” para cambiar al siguiente dato y modificar el dato actual, respectivamente. El sistema se encuentra configurado para formato de 24 horas (0-23 horas) y diseñado para compensación de años bisiestos.

Una vez finalizada la configuración, los datos son escritos en la memoria interna del RTC para su posterior actualización, generándose posteriormente un retorno hacia el menú de selección de configuración. En la Figura 14 se muestra el diagrama de flujo correspondiente al modo de ajustes de hora y fecha del sistema.

4.1.2.2 Ingreso de Clave de Acceso

Para conferir seguridad a algún piso mediante ingreso de clave de acceso es necesario seleccionar el modo “Ingresar Código de Acceso” del menú principal. Al ingresar en este modo se mostrará en pantalla una petición de ingreso del piso al cual se le desea asignar o modificar el código para su acceso. Si el piso ingresado es incorrecto, se mostrará un mensaje indicando que el piso es erróneo, y posteriormente será solicitado nuevamente el ingreso de un piso para su configuración o ingreso de código de acceso. De ser válido el piso ingresado se visualizará en la LCD un mensaje de petición de código (si previamente poseía alguno) y de ser ingresado el código correcto se procederá a su configuración, de lo contrario se mostrará en el display un mensaje indicando que la clave es incorrecta.

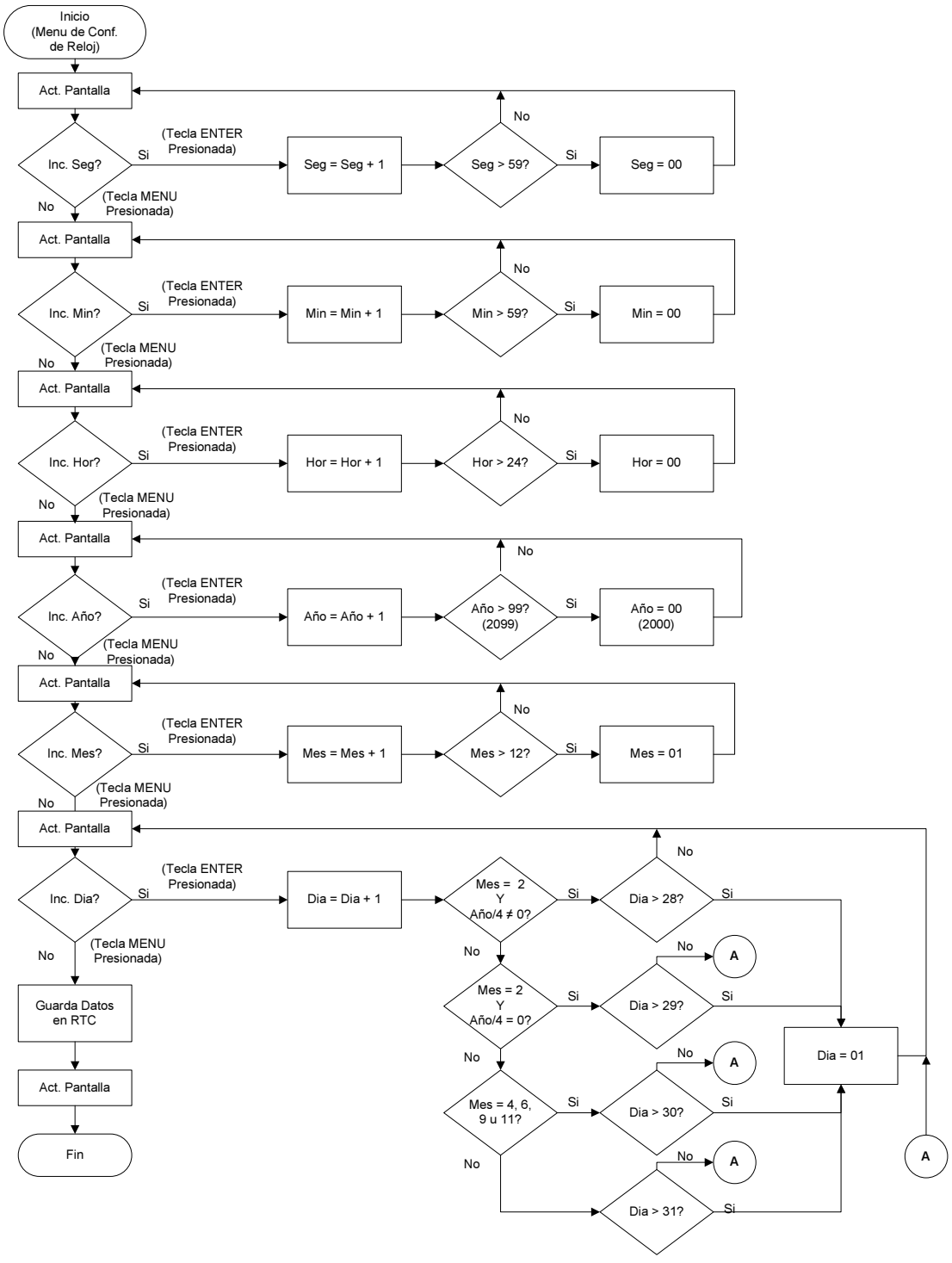

Figura 14: Diagrama de Flujo para Ajuste de Hora y Fecha del Sistema.

De haberse confirmado el acceso por medio del ingreso de la clave o bien si el piso no poseía una previamente programada, se mostrará en pantalla un mensaje de petición de ingreso de nueva clave y al ingresarse la misma (4 dígitos) se indicará una nueva petición para verificación de la clave previamente ingresada. Si ambos códigos ingresados coinciden, dicha clave es evaluada a fin de verificar que no exista otro piso con el mismo código de seguridad, y de no existir, la misma será guardada en la posición de memoria (EEPROM Externa) correspondiente al piso seleccionado, mostrando posteriormente un mensaje de que el código ingresado ha sido guardado, retornando seguidamente al menú principal. De existir la clave previamente en algún piso será mostrado en la LCD un mensaje de error indicando que la clave no puede ser guardada dado que se evita esta duplicación para el caso de ser activada la función "Servicio Privado".

Es de interés recordar que este modo puede ser cancelado en cualquier momento al ser presionada la tecla "MENU" del teclado de tres pulsadores utilizado en el diseño en cuestión. En la Figura 15 se muestra el diagrama de flujo correspondiente al modo de ingreso de código de acceso del sistema.

4.1.2.3 Ingreso de Horas de Acceso

Por otra parte, si el acceso hacia algún piso desea ser limitado mediante el establecimiento de un horario, debe ser seleccionado el modo "Ingresar Horas de Acceso" del menú principal. Al ingresar en este modo se mostrará en pantalla una petición de ingreso del piso al cual se le desea asignar o modificar el horario para su ingreso. Si el piso ingresado es incorrecto, se mostrará un mensaje indicando que el piso es erróneo, y posteriormente será solicitado nuevamente el marcado de un piso para su configuración o ingreso de horario de admisión. De ser válido el piso ingresado se visualizará en la LCD un mensaje de petición de código (si previamente poseía alguno) y de ser ingresado el código correcto se procederá a su configuración,

de lo contrario se mostrará en el display un mensaje indicando que la clave es incorrecta.

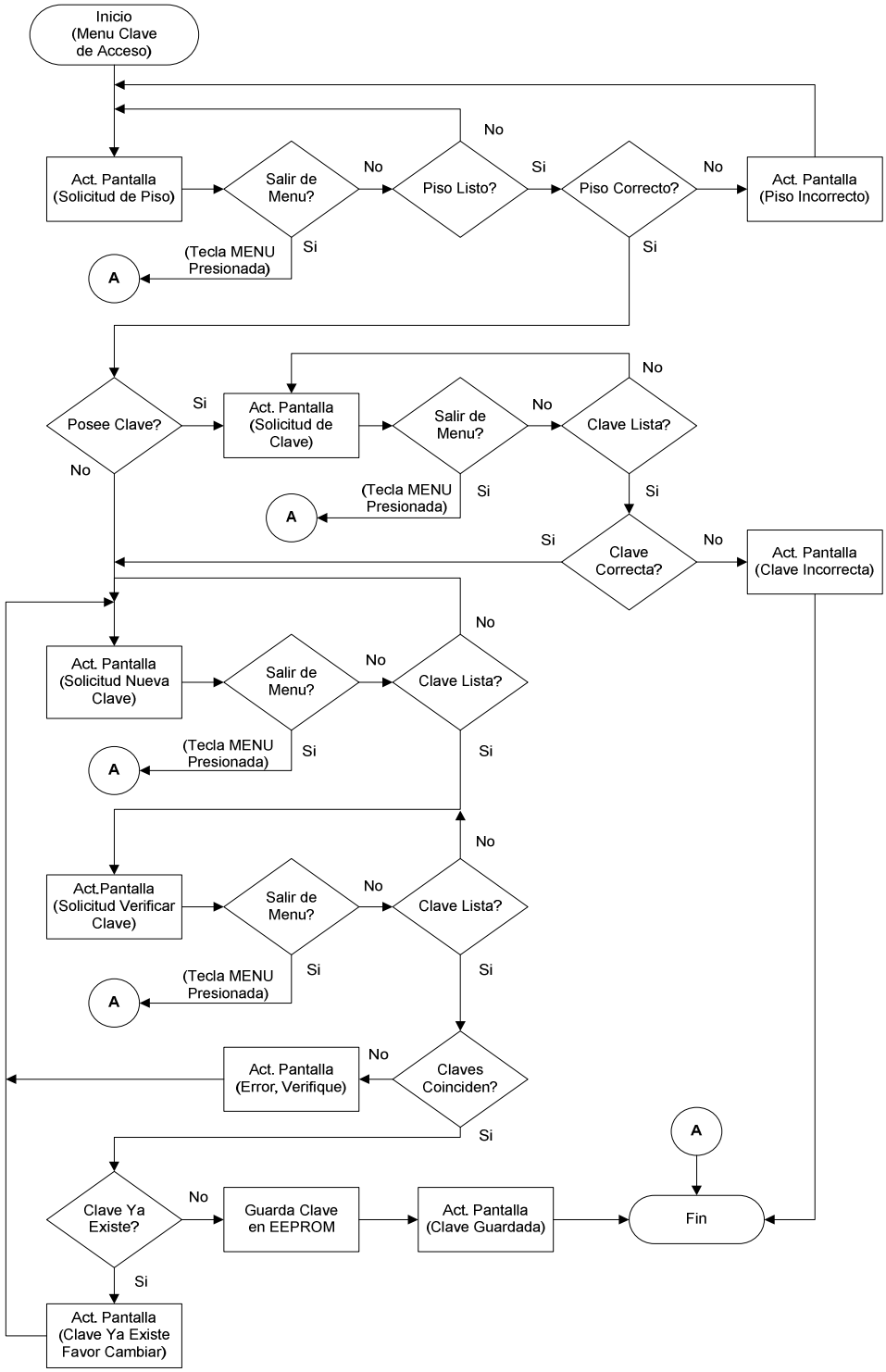

Figura 15: Diagrama de Flujo para Ingreso de Código de Acceso.

De haberse confirmado el acceso por medio del ingreso de la clave o bien si el piso no poseía una previamente programada, se mostrará en pantalla un mensaje de petición de ingreso de hora inicial de acceso al piso indicado, y al ingresarse la misma (hora y minutos) se indicará una nueva petición para el ingreso de la hora final para el acceso, a fin de que el ingreso hacia dicho piso sea concedido sólo en el horario comprendido entre tales horas programadas. Una vez finalizada la configuración asignada, estas horas son guardadas en la posición de memoria (EEPROM Externa) correspondiente al piso seleccionado, mostrando posteriormente un mensaje de que las horas ingresadas han sido guardadas, retornando seguidamente al menú principal.

Es de interés recordar que este modo puede ser cancelado en cualquier momento al ser presionada la tecla “MENU” del teclado de tres pulsadores utilizado en el diseño en cuestión. La Figura 16 muestra el diagrama de flujo correspondiente al modo de ingreso de horas de acceso del sistema.

4.1.2.4 Borrado de Clave y Horas de Acceso

El control de acceso mediante códigos o por medio de restricción de algún horario programado resulta de gran utilidad, sin embargo existen ocasiones en donde el o los usuarios optan por eliminar dicho nivel de seguridad, por lo que se decidió integrar las opciones “Borrar Código de Acceso” y “Borrar Horas de Acceso”, a fin de que el ingreso hacia el piso seleccionado sea concedido de manera convencional.

Si alguna de estas opciones es seleccionada, se mostrará en pantalla una petición de ingreso del piso al cual se le desea eliminar la clave o el horario para su ingreso. Si el piso ingresado es incorrecto, se mostrará un mensaje indicando que el piso es erróneo, y posteriormente será solicitado nuevamente el discado de un piso para el ingreso hacia la operación seleccionada. De ser válido el piso ingresado se visualizará en la LCD un mensaje de petición de código (si previamente poseía

alguno) y de ser ingresado el código correcto se procederá a su configuración, de lo contrario se mostrará en el display un mensaje indicando que la clave es incorrecta.

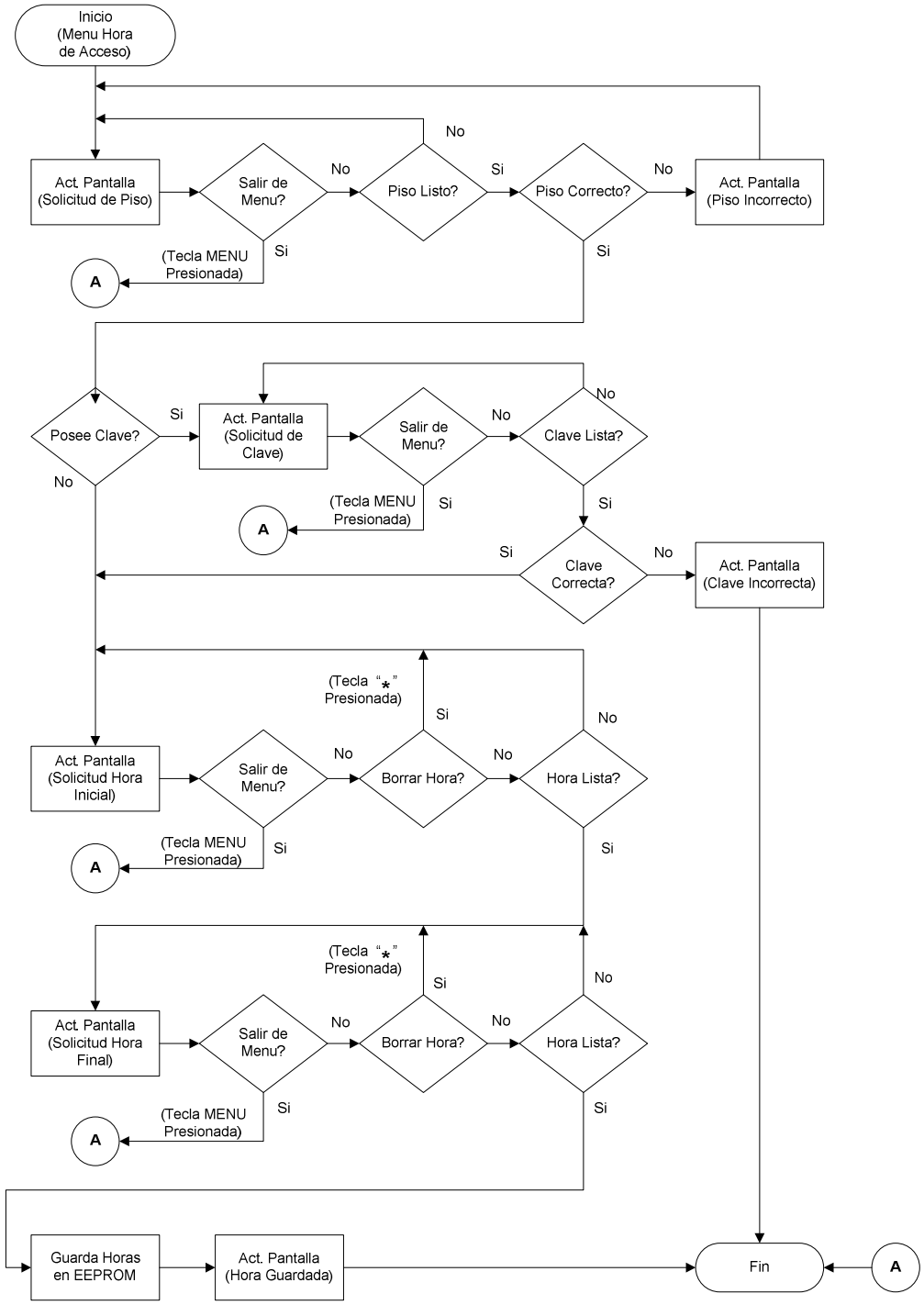

Figura 16: Diagrama de Flujo para Ingreso de Horas de Acceso.

De haberse confirmado el acceso por medio del ingreso de la clave (borrar código) el sistema procederá al borrado de esta clave, almacenando tal información en la EEPROM Externa y dejando al piso libre de esta modalidad de ingreso. Por otra parte, si el piso carecía de alguna clave previamente programada, el sistema mostrará un mensaje indicando que el piso seleccionado no tenia clave alguna programada. Ahora bien, de confirmarse el acceso por medio del ingreso de una clave correcta o bien por la ausencia de la misma hacia el modo de borrado de horas de acceso, se mostrará en pantalla un mensaje indicando que las horas de acceso programadas (de haberlas tenido) han sido borradas, almacenando tal información en la EEPROM Externa y dejando al piso libre de esta modalidad de ingreso.

Una vez finalizada la configuración asignada, el sistema procederá a retornar al menú principal.

Es de interés recordar que cualquiera de estos modos puede ser cancelado en cualquier momento al ser presionada la tecla “MENU” del teclado de tres pulsadores utilizado en el diseño en cuestión.

En las Figuras 17 y 18 se muestran los diagramas de flujo correspondientes al modo de borrado de código de acceso y el de borrado de horas de acceso del sistema, respectivamente.

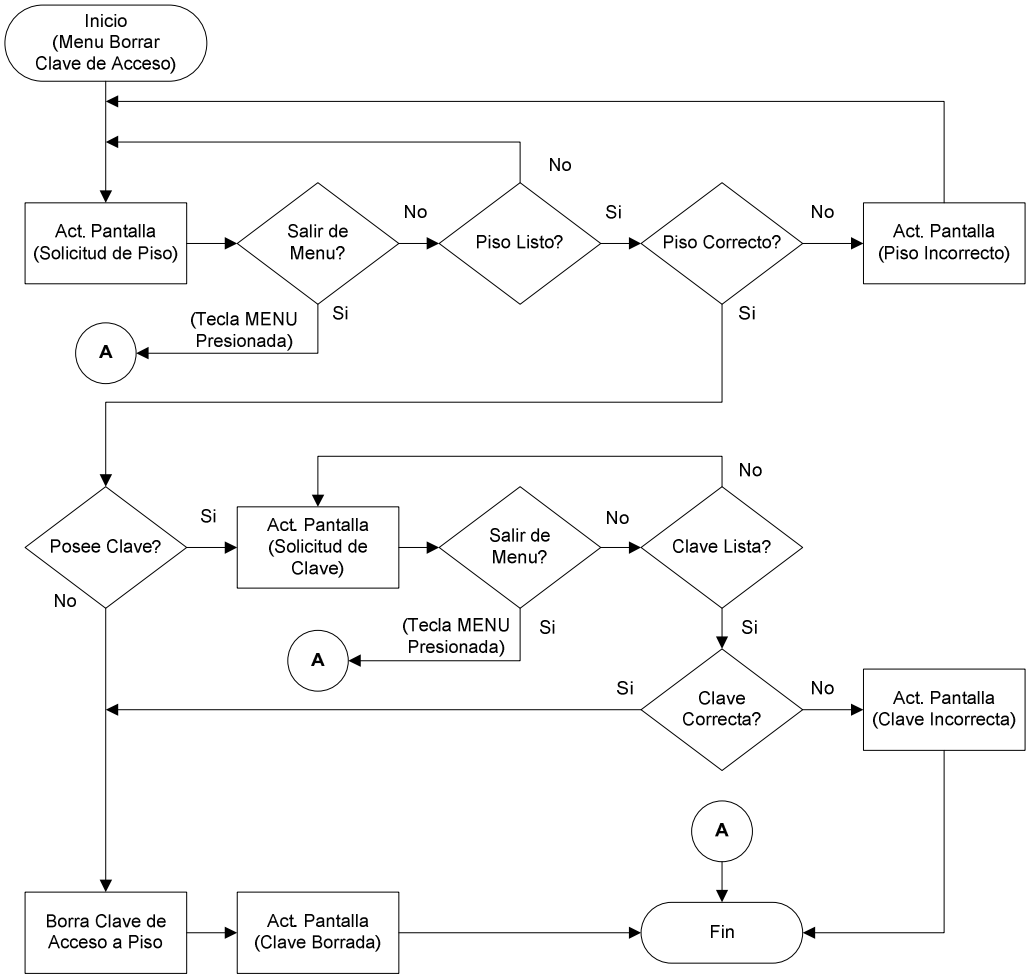

Figura 17: Diagrama de Flujo para Borrado de Código de Acceso.

4.1.2.5 Entrada de Datos de Botonera (Llamadas)

Si del menú principal fue seleccionada la opción que lleva por nombre "Inicializar" se procede a dar inicio al programa para el cual se confiere la interfaz diseñada, permitiendo de ésta manera mostrar los datos de hora y fecha de forma actualizada mediante el display utilizado, dar opción a una nueva configuración de los datos para el ingreso a pisos por medio de la tecla "MENU", y realizar barrido continuo de la botonera de cabina del ascensor para verificar la existencia o no de una llamada realizada.

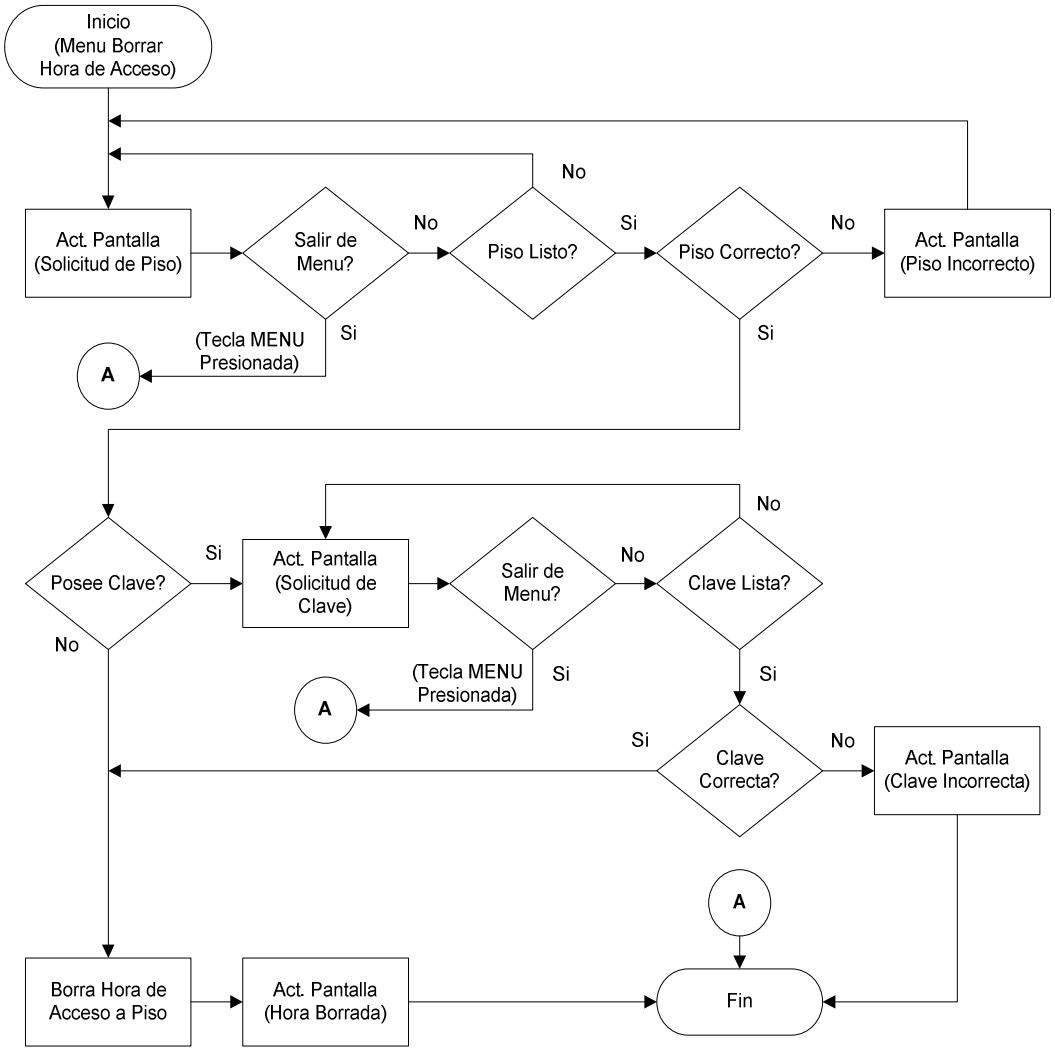

Figura 18: Diagrama de Flujo para Borrado de Horas de Acceso.

La verificación de petición de alguna llamada se realiza al producirse la interrupción externa programada en el microcontrolador y de ésta manera el microcontrolador toma la señal proveniente de la botonera de cabina, la transforma de modo que pueda ser procesada de forma adecuada por medio de comparaciones con diversas variables, y una vez obtenido el valor del piso que fue solicitado se procede a la lectura de los datos almacenados en la EEPROM externa, a fin de verificar si el piso presenta algún modo programado para otorgar su acceso.

Posteriormente se procede en primera instancia a verificar si el piso discado posee intervalos de hora de acceso programadas, de ser así se realiza una comparación entre las horas configuradas con la hora actual para ceder el permiso o no al ingreso del mismo.

De ser denegado el acceso de acuerdo a una hora preconfigurada en el sistema, la interfaz anula de forma instantánea la petición de ingreso, y procede a seguir con sus funciones asignadas. De lo contrario, si el ingreso es aceptado la interfaz verificará si el piso posee alguna clave de acceso asignada, de no poseer alguna el sistema realizará la petición de la llamada deseada, por el contrario de tener configurado algún código de seguridad, se mostrará por medio de la LCD un mensaje de petición de clave.

Una vez ingresado y confirmado por el sistema se ejecutará de manera correcta la llamada indicada, de ser incorrecto el código discado el sistema mostrará un mensaje de acceso denegado, anulando la llamada solicitada, y seguidamente procesando sus funciones programadas. Si la llamada solicitada posee clave de acceso programada o bien si por error fue presionado un piso que si la posee, dicha llamada puede ser cancelada pulsando la tecla “*” del teclado matricial, permitiendo realizar una nueva llamada desde la cabina del ascensor.

En la Figura 19 se muestra el diagrama de flujo correspondiente a la entrada de datos proveniente de la botonera de cabina (llamadas).

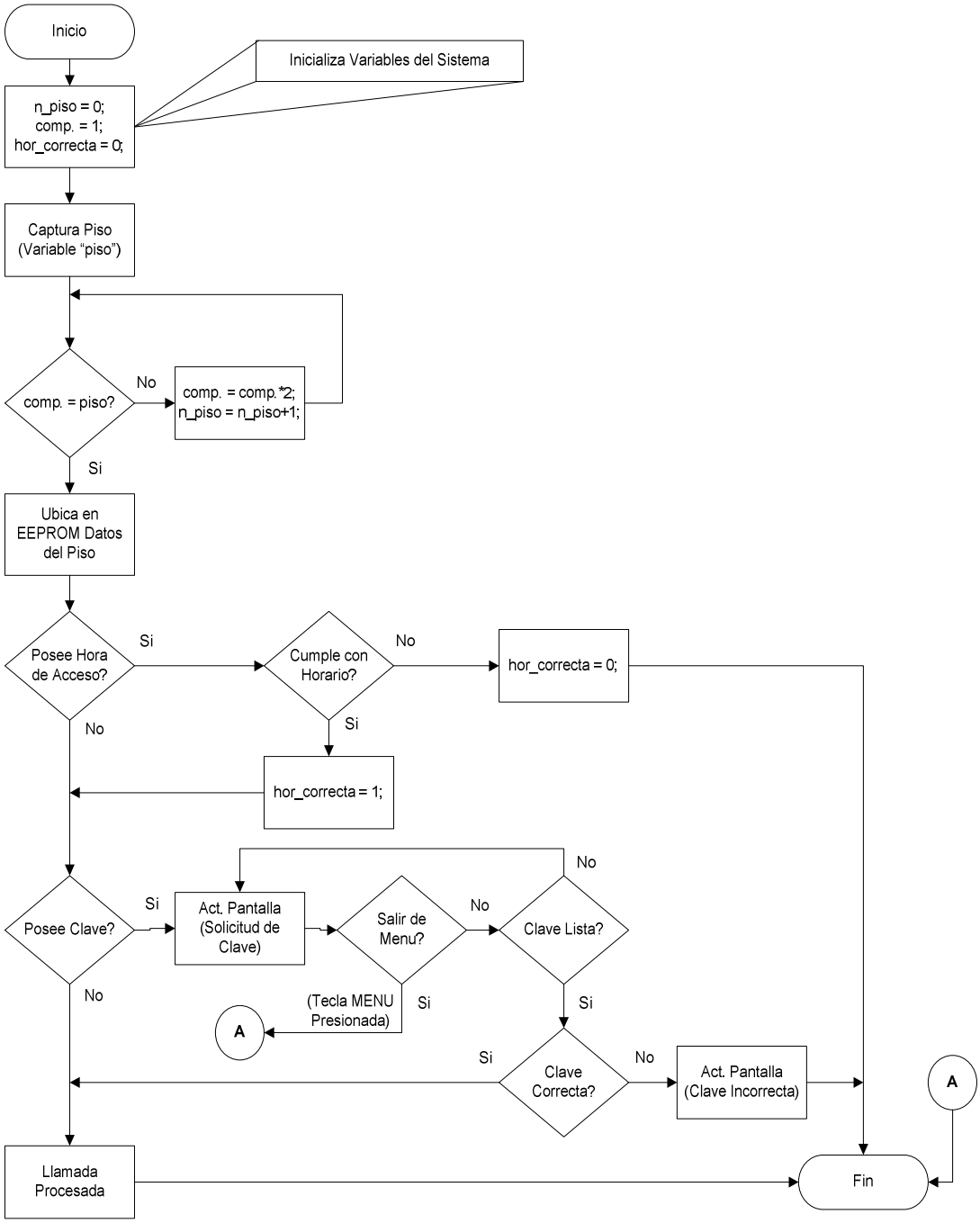

Figura 19: Diagrama de Flujo de Llamada de Cabina.

4.1.2.6 Salida de Datos hacia Tarjeta Maestra

Además de tener en cuenta el proceso de una llamada entrante, es conveniente saber de qué manera esta es enviada hacia la tarjeta principal del ascensor. Una vez procesada y confirmada la llamada, el dato obtenido originalmente de la cabina es enviado continuamente durante un tiempo determinado hacia los shift register ubicados en la salida de la interfaz, posteriormente el dato a enviar durante el mismo tiempo hacia estos dispositivos será un "0" para que de ésta manera todas las señales de salida de la interfaz sean apagadas, así pues la llamada en cuestión ya ha sido ejecutada. A su vez, terminado este proceso, estos dispositivos (shift register 74LS595) son inhabilitados a fin de evitar que los mismos realicen "falsas llamadas" debido a ruido producto de interferencias cercanas. En la Figura 20 se muestra el diagrama de flujo correspondiente a la salida de datos hacia la tarjeta principal (llamada ejecutada).

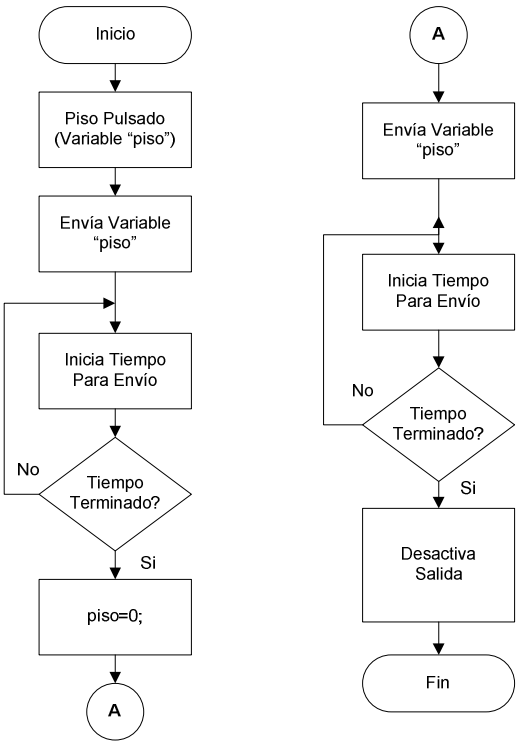

Figura 20: Diagrama de Flujo de Llamada Ejecutada.

4.1.2.7 Servicio Privado

Existen lugares cuyo acceso mediante ascensores se otorga de manera diferente a los dispuestos por los ascensores convencionales. Por tal motivo, el sistema posee una funcionalidad de identificación personal por piso (servicio privado). Tras ser activado el sistema en esta modalidad, continuamente la interfaz se encontrará mostrando mediante la pantalla de caracteres los datos asociados a fecha y hora de forma actualizada, y ser existir alguna llamada, se mostrará por medio de dicha pantalla una petición de código.

Una vez ingresado este código, el sistema lo verificará y de ser aceptado, procederá a inspeccionar si existe alguna hora de ingreso programada. De no existir o bien de ser permitido el acceso en ese instante, la interfaz realizará la llamada hacia el piso asociado a dicho código suministrado. Por el contrario, de poseer una hora de restricción el sistema no efectuará dicha solicitud de acceso. En caso de que no exista tal clave será mostrado en el display un mensaje de clave no existente. Así mismo la llamada puede ser interrumpida en todo momento al presionar la tecla “*”, actualizando posteriormente la pantalla LCD de forma tal que la hora y la fecha se muestren adecuadamente.

Cabe destacar que el nivel “Planta Baja” no debe poseer clave alguna, por lo que le fue asignado el botón “#” a dicha planta, permitiendo realizar tal solicitud de llamada al presionar sólo por primera vez esta tecla, ya que de lo contrario esta tecla no tendrá efecto alguno si previamente se estaba ingresando alguna clave.

En la Figura 21 se muestra el diagrama de flujo correspondiente al sistema de servicio privado.

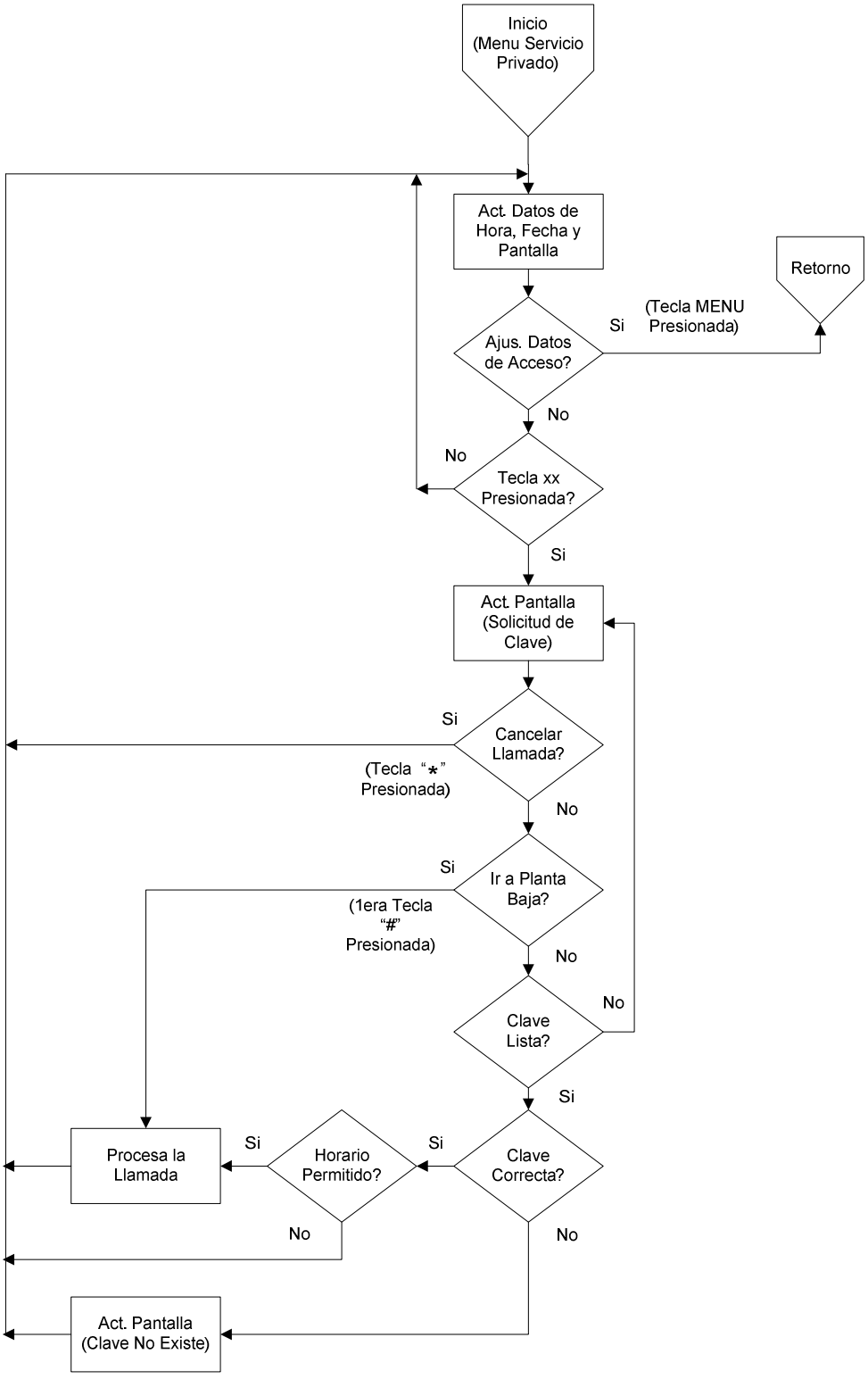

Figura 21: Diagrama de Flujo del Sistema de Servicio Privado.

4.1.3 Manejo de Pantalla

Por medio de la pantalla alfanumérica de 2x16 caracteres se da la presentación de mensajes de interés y/o información proveniente del microcontrolador o de cualquiera de los periféricos utilizados en el presente proyecto. La muestra de caracteres en el display comienza con la inicialización de la pantalla LCD propiamente dicha, ya que allí se configura el modo de trabajo de 4 bits, la selección de número de líneas y fuente, la configuración del cursor y finalmente el borrado del contenido que esta pueda tener.

Una vez configurado el display se verifica en que fila y posición de la misma se desea mostrar el mensaje, teniendo en cuenta que al ser seleccionada la primera fila el cursor no debe estar en la última posición, ya que de ser así el cursor sería ubicado en la segunda fila de este display. Por el contrario, si el cursor se encontraba en la última posición correspondiente a la segunda fila, el cursor sería inmediatamente ubicado en la primera posición de la fila número uno del mismo.

Realizado esto, se procede a la lectura de la DDRAM del LCD, memoria volátil en la cual se almacenan los caracteres que se van a mostrar en la pantalla, y una vez leída se muestra el mensaje en la posición actual del cursor. Pasado el tiempo de escritura de los caracteres en la pantalla, se incrementa en uno la posición del cursor, y culminado esto se abandona esta modalidad hasta la muestra de un nuevo mensaje.

A continuación en la Figura 22 y 23 se muestran los diagramas de flujo correspondientes a la inicialización de la pantalla LCD y la escritura de caracteres en el mismo, respectivamente.

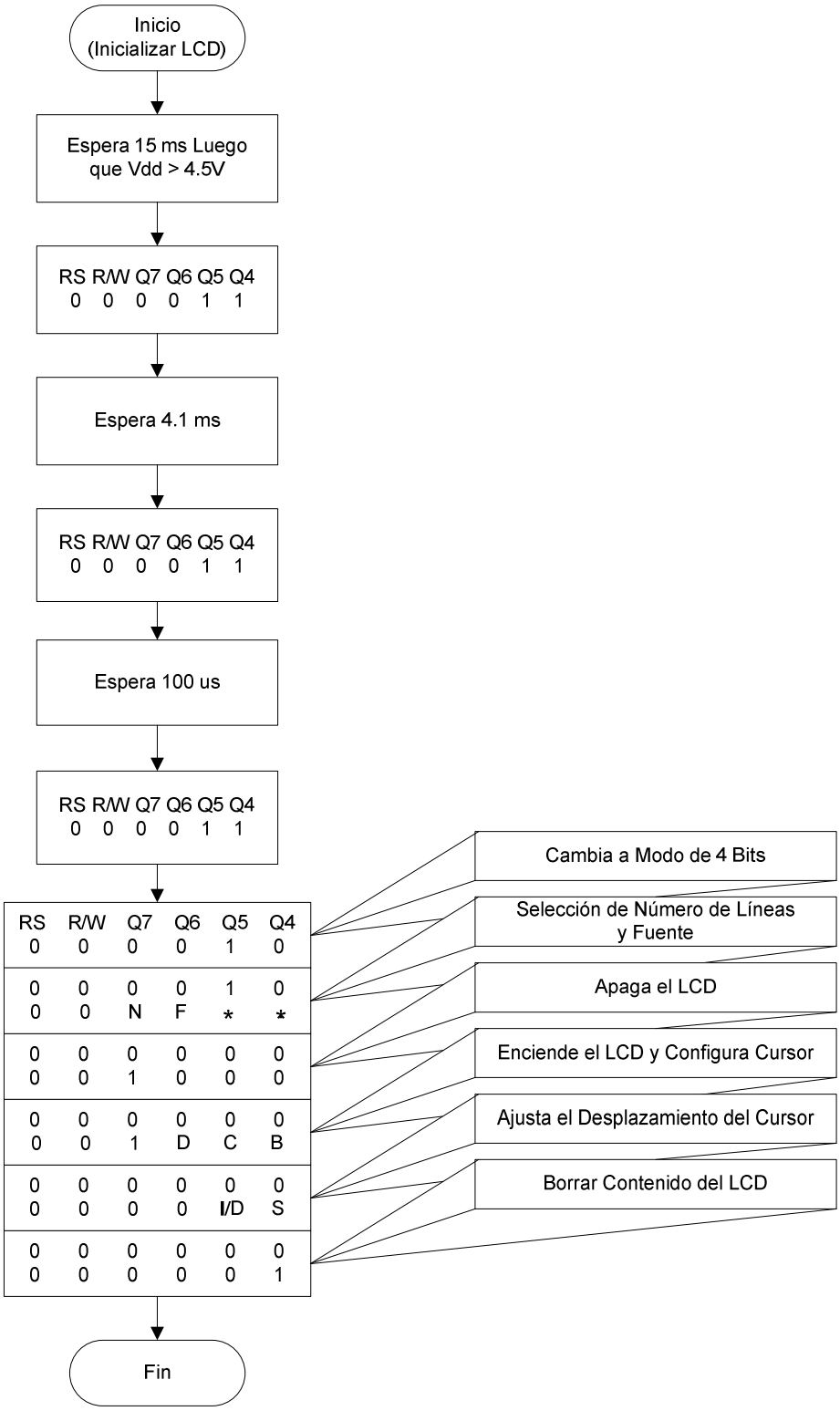

Figura 22: Diagrama de Flujo de Inicialización de Pantalla LCD.

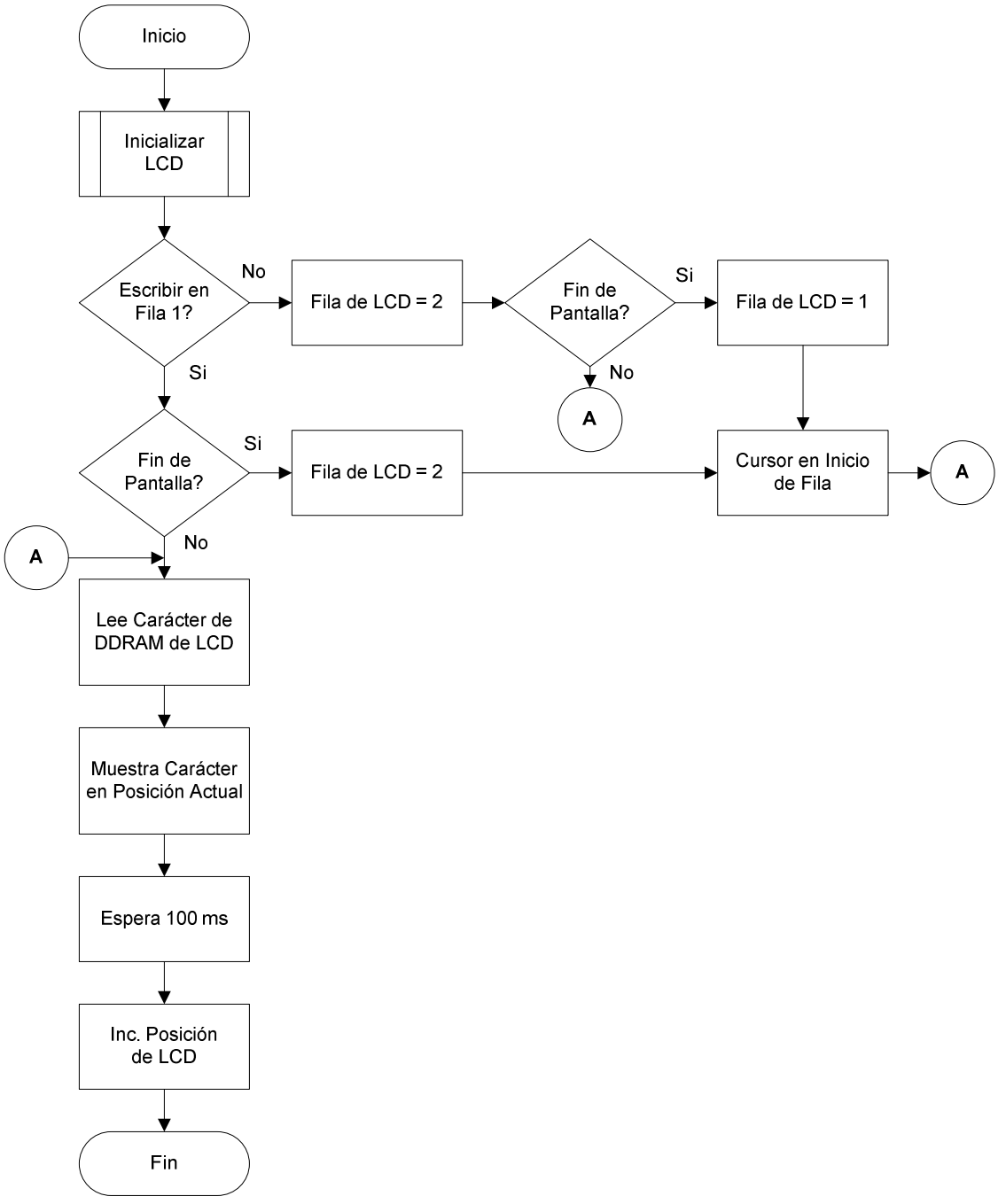

Figura 23: Diagrama de Flujo de Modo de Escritura en LCD.

CAPÍTULO V

5.1 PRUEBAS Y RESULTADOS

Una vez culminada la implementación del sistema diseñado se procedió a la realización de una serie de pruebas a fin de verificar su correcta funcionalidad. Las mismas se muestran a continuación:

5.1.1 Comportamiento Aislado del Prototipo Implementado

El sistema diseñado fue energizado a fin de verificar cada una de las etapas que lo componen, es decir, las diversas operaciones de ajuste y modificación para el cual fue elaborado. Cada una de estas pruebas fue realizada de forma manual haciendo uso de cables de hilo de cobre como sustitución de la botonera de cabina.

En los diagramas de flujo mostrados en los anexos 2 se observan las diversas pantallas mostradas según selección de configuración por parte del técnico encargado.

5.1.2 Acoplamiento Botonera-Interfaz

Habiendo verificado el comportamiento aislado del prototipo realizado, y asegurando su funcionamiento se procedió al acople entre la botonera del ascensor con la interfaz elaborada logrando obtener el vinculo buscado a fin de que el sistema diseñado captara convenientemente las señales provenientes de la cabina.

5.1.3 Acoplamiento Interfaz-Tarjeta Principal

Una vez realizada las pruebas convenientes con la botonera de cabina se procedió al acople entre la interfaz y la tarjeta principal de control del ascensor. Una vez conectadas se realizaron una serie de pruebas a fin de verificar su correcto funcionamiento logrando observar que las señales eran enviadas correctamente según conveniencia desde la interfaz hacia la tarjeta de control principal.

5.1.4 Funcionamiento Botonera-Interfaz-Tarjeta Principal

En base a los resultados previamente contemplados se decidió realizar las pruebas generales con todos los dispositivos dispuestos tanto para la implementación como para la adaptación de tal interfaz sobre el sistema disponible por parte de la compañía, logrando obtener resultados deseados anteriormente mencionados.

CONCLUSIONES

En función de los resultados obtenidos en el desarrollo del presente trabajo, se enumeran a continuación los puntos concluyentes a los que se pudo llegar:

- Se logró acoplar correctamente la interfaz diseñada con los elementos de interés asociados al ascensor, siendo estos la botonera de cabina y la tarjeta maestra.

- Se obtuvo un sistema de control de acceso con las funcionalidades mínimas exigidas para su correcto funcionamiento.

- El comportamiento asociado al software diseñado corresponde a los requerimientos mínimos exigidos por la empresa, permitiendo convenientemente el uso o no de los sistemas de seguridad para el ingreso de un piso determinado.

- Se logró establecer una correcta comunicación entre los elementos del sistema desarrollado, teniendo como resultado un hardware adecuado a las exigencias de la empresa.

- La interfaz elaborada mantuvo los esquemas y bases tecnológicas implementadas en los equipos utilizados por la compañía.

- La adaptación de la pantalla LCD ayuda a que el usuario visualice la hora de forma que este pueda saber si es posible su ingreso hacia un piso de interés, de acuerdo a un horario pre-programado.

- El uso de una memoria EEPROM externa fortalece el almacenamiento de la data suministrada por los usuarios, ya que la misma mantiene estos datos incluso si el microcontrolador de la interfaz es reprogramado.

- La incorporación del sistema desarrollado no interfiere con las funciones asociadas al ascensor debido a que la programación de la interfaz implementada no afecta el programa pre-instalado del microcontrolador de la tarjeta principal ni a sus dispositivos.

- En todo momento fue aprovechada la infraestructura existente para el acople de la interfaz implementada, resguardando así su espacio físico de aplicación.

- Los teclados numéricos como sistemas de control de acceso, así como los diseñados para el ingreso hacia un piso mediante horarios pre-configurados, resultan un medio de seguridad práctico, fácil de emplear y con un bajo costo atribuido, considerando que se encuentran programados bajo el uso de software externo.

- La interfaz diseñada posee actualmente un hardware configurado para la aceptación de una estructura de 10 plantas (pisos) mas posee un software diseñado y configurado para admitir su instalación en una edificación con un máximo de cien (100) pisos.

- El sistema trabaja de forma correcta mediante la utilización de baterías como fuente de alimentación (fuente DC), o bien sustentada por medio del sistema eléctrico (fuente AC).

RECOMENDACIONES

En base a los resultados obtenidos se proponen las siguientes recomendaciones a fin de mejorar y reforzar las prestaciones del producto final desarrollado en este trabajo:

- Hacer uso de placas de circuito impreso (PCB) doble cara, para así evitar el uso de cableado (puentes) en la tarjeta final.

- Es conveniente agregar módulos para la expansión de entradas y salidas, permitiendo de esta manera que sean aprovechadas las capacidades de la interfaz diseñada.

- Hacer uso de un teclado numérico de mayor robustez a fin de asegurar una larga durabilidad.
- Se recomienda el uso de nuevas tecnologías como lo son los microcontroladores de la familia PIC24 dada su amplia gama de aplicaciones, pudiendo desarrollar sistemas del tipo SCADA en sus ascensores evolucionando a sistemas inteligentes.

REFERENCIAS BIBLIOGRAFICAS

[1] Novenca Security Systems [en línea].
<http://www.novenca.com/site/index.php?option=86&Itemid=74> [Consulta: 2010]

[2] RFID [en línea]. <http://es.wikipedia.org/wiki/RFID> [Consulta 2010]

[3] Introducción a la tarjeta con banda magnética [en línea].
<http://www.monografias.com/trabajos43/banda-magnetica/banda-magnetica.shtml>
[Consulta 2011]

[4] Código de barras [en línea].
<http://es.wikipedia.org/wiki/C%C3%B3digo_de_barras> [Consulta 2011]

[5] Biometría [en línea]. <http://es.wikipedia.org/wiki/Biometr%C3%ADa>
[Consulta 2010]

BIBLIOGRAFÍA

Tesis.

Gordillo Pérez, Victor M. Automatización de tres ascensores./ Gordillo Pérez Victor M. Barrado José Antonio (Tesis).-- Tarragona: Universitat Rovira i Virgili, 2002.

Nunes F, Joao Luis G. Diseño e implementación de un modelo de teléfono público inalámbrico basado en la tecnología gsm./ Nunes F Joao Luis G (Tesis).-- Caracas: Universidad Central de Venezuela, 2008.

Manuales.

Manual de Operación: Control Electrónico de Ascensores. 10 de Febrero de 2011.

Manual de Programación: Manual de Programación Placa F40. 16 de Febrero de 2011.

Manual de Referencia: Manual de Usuario del Compilador PCW de CCS. 22 de Septiembre de 2010.

Manual de Referencia: Pantalla de Caracteres LCD. 30 de Abril de 2011.

Microchip Reference Manual: Mid-Range MCU Family. 23 de Enero de 2011.

Manual de Reparación: Reparación de Fallas en Placas JYE. 2 de Diciembre de 2010.

Libros.

Gardner, Nigel. An introduction to programming the microchip pic in ccs c, USA: Editorial Bluebird Electronics, 2002.

Breijo, Eduardo García. Compilador c ccs y simulador proteus para microcontroladores pic, México: Editorial Alfaomega, 2008.

Internet.

Villela Tejeda Héctor, Manual de C [en línea].
<http://www.fismat.umich.mx/mn1/manual/> [Consulta: 2010]

Urrutia Gorka, El rincón de C [en línea].
<http://www.elrincondelc.com/cursoc/cursoc.html> [Consulta: 2010]

Johnson, Shawn, Curso de Microcontroladores PIC [en línea].
<http://www.cursomicros.com> [Consulta: 2011]

ARQHYS, Ascensores [en línea].
< http://www.arqhys.com/arquitectura/ascensor-historia.html> [Consulta: 2010]

Afinidad Eléctrica, La Historia del Ascensor [en línea].
<http://www.afinidadelectrica.com.ar/articulo.php?IdArticulo=125> [Consulta: 2010]

Historia del Ascensor [en línea].
< http://www.saber.golwen.com.ar/hascensor.htm> [Consulta: 2010]

ANEXOS

[ANEXO Nº 1]

[Código De Programas Asociados Al Diseño Elaborado]

ProyectoFinal.c:

```
#include "18f4520.h"                                                    //Microcontrolador Utilizado
#include "74165.c"                                                      //Libreria de Shift Register 74LS165
#include "74595.c"                                                      //Libreria de Shift Register 74LS595
#fuses XT,NOWDT,NOPROTECT,NOLVP
#use delay(clock=4000000)                                               //Uso de Reloj de 4MHz
#use i2c(Master, sda=PIN_C4, scl=PIN_C3)                                //Habilitación del Puerto I2C
#include "24128.c"                                                     //Libreria de Memoria EEPROM Externa 24LC128
#include "RTC_DS1307.c"                                                 //Libreria de RTC DS1307 Externo
#include "lcd_2.c"                                                      //Libreria de LCD
#include "kbd_2.c"                                                      //Libreria de Teclado Matricial 4x3
#use fast_io(A)
#use standard_io(C)
#use standard_io(D)
//************************************************************************//
//************************************************************************//
#define MENU     PIN_A0                                                 //   Asociación de Pines Correspondientes a Teclado de 3
#define ENTER    PIN_A1                                                 // Pulsadores.
#define EDITAR   PIN_A2

#define numero_pulsos 4                                                 //   Valor de Comparación Para Tecla Pulsada en la
Cabina.
#define TT1 65535-65535/100                                             //   Valor de Cuenta del Timer 1.
//************************************************************************//
//************************************************************************//
int8 veces_pulsado, salir;                                              //   Variable que Permite Comparar con la Variable
"cuentas".
                                                                        // Variable que Permite Salir de Este Menú Cuando Está
en 1
                                                                        // y Seguir en él Cuando Está en 0.
long p_pulsado;                                                         //   Variable Asociada a Piso Pulsado en la Cabina.
int hor=2, min=30, seg=0, date, dia=1, mes=1, a_o=0, menu_rtc, conf_datos=0;     //   Variables Globales del Programa.
byte inicia_programa=0, PB;                                             //   Variable que Permite o no el Inicio de la Función
Principal.
char clave1[4], clave2[4], clave3[4], clave4[4], clave5[4], clave6[4];
//************************************************************************//
//************************************************************************//
#INT_EXT                                                                //   Interrupción Externa
void IntExt(void){                                                      //
  disable_interrupts(GLOBAL);                                           //
  disable_interrupts(int_ext);                                          //   Al Producirse una Interrupción Externa (Llamada) se
  enable_interrupts(int_timer1);                                        // Inhabilita Tal Interrupción a Fin de Que no Ocurra
  veces_pulsado=0;                                                      // Algun Otra Llamada Mientras Ésta es Atendida; a su
vez es
  setup_timer_1(T1_INTERNAL|T1_DIV_BY_8);                               // Habilitada la Interrupción de Timer 1 a un Valor
Predeterminado
  set_timer1(TT1);                                                      // y se Inicializa un Contador Para Realizar una Serie
de
  enable_interrupts(GLOBAL);                                            // Comparaciones en Dicha Interrupción Habilitada.
}                                                                       //
//************************************************************************//
//************************************************************************//
#INT_TIMER1                                                             //   Interrupción de Timer 1
void IntT1(void){

  read_expanded_inputs(&p_pulsado);                                     //   Es leído el Piso Pulsado y Configurado Para
  p_pulsado=~p_pulsado;                                                 // ser Evaluado.

  if (p_pulsado==0){                                                    //   Es Desactivada la Interrupción de Timer 1 en Caso
     disable_interrupts(GLOBAL);                                        // de una Falsa Llamada, el Contador es Puesto en "0"
     disable_interrupts(int_timer1);                                    // de Nuevo y Posteriormente es Habilitada Nuevamente
     setup_timer_1(T1_DISABLED);                                        // la Interrupción Externa Para Esperar una Nueva
Llamada.
     veces_pulsado=0;
     ext_int_edge(L_TO_H);
     enable_interrupts(int_ext);
     enable_interrupts(GLOBAL);
    }

  else{veces_pulsado++;                                                 //   Es Incrementado el Contador en 1 Cada vez que el
Timer 1
       setup_timer_1(T1_INTERNAL|T1_DIV_BY_8);                          // Culmina su Cuenta, con lo que Seguidamente es Cargado
       set_timer1(TT1);                                                 // una Vez más Para una Nueva Puesta en Marcha.
      }

  if (veces_pulsado==numero_pulsos){                                    //   Cuando el Valor del Contador es Igual a un Número
de
     disable_interrupts(int_timer1);                                    // Preestablecido, Implica que Hubo una Llamada,
Descartando
     setup_timer_1(T1_DISABLED);                                        // de Esta Manera un Posible Error de Pulsación en la
Cabina;
     ext_int_edge(L_TO_H);                                              // la Interrupción del Timer 1 es Desactivada Mientras
     enable_interrupts(int_ext);                                        // que la Interrupción Externa es Habilitada Nuevamente.
     enable_interrupts(GLOBAL);
  }
}
//************************************************************************//
//************************************************************************//
typedef struct                                                          //   Definición Para Tipo de Datos Estructurados.
 {char Hi,Mi,Hf,Mf;                                                     //   Variables Correspondientes a Hora y Minuto Inicial
y Final.
  char code[4];                                                         //   Vector Para Guardado de Clave en EEPROM Externa.
```

```
 } RegPisoType;                                                    //   Nombre de Código Estructurado.

#define RegPisoSize 8                                             //   Tamaño de Bytes Utilizados de la EEPROM Externa por
Piso.

RegPisoType RegPiso;                                              //   Puntero en Código Estructurado.

void ReadRegPiso(unsigned char Piso)                              //   Lectura de Datos de EEPROM Externa por Piso.
 {Piso--;                                                         //   Es Restado 1 el Piso de LLamada Debido a que
Ingresa
                                                                  // con Valor Excedido en 1.
  init_ext_eeprom();                                              //   Inicialización de EEPROM Externa.
  RegPiso.Hi     =read_ext_eeprom(Piso*RegPisoSize+0);            //   Lectura de Hora Inicial del Piso Seleccionado.
  RegPiso.Mi     =read_ext_eeprom(Piso*RegPisoSize+1);            //   Lectura de Minuto Inicial del Piso Seleccionado.
  RegPiso.Hf     =read_ext_eeprom(Piso*RegPisoSize+2);            //   Lectura de Hora Final del Piso Seleccionado.
  RegPiso.Mf     =read_ext_eeprom(Piso*RegPisoSize+3);            //   Lectura de Minuto Final del Piso Seleccionado.
  RegPiso.code[0]=read_ext_eeprom(Piso*RegPisoSize+4);            //   Lectura de Primer Digito de Clave del Piso
Seleccionado.
  RegPiso.code[1]=read_ext_eeprom(Piso*RegPisoSize+5);            //   Lectura de Segundo Digito de Clave del Piso
Seleccionado.
  RegPiso.code[2]=read_ext_eeprom(Piso*RegPisoSize+6);            //   Lectura de Tercer Digito de Clave del Piso
Seleccionado.
  RegPiso.code[3]=read_ext_eeprom(Piso*RegPisoSize+7);            //   Lectura de Cuarto Digito de Clave del Piso
Seleccionado.
 }

void WriteRegPisoHoras(unsigned char Piso)                        //   Escritura de Horas de Acceso en EEPROM Externa por
Piso.
 {Piso--;                                                         //   Es Restado 1 el Piso de LLamada Debido a que
Ingresa
                                                                  // con Valor Excedido en 1.
  init_ext_eeprom();                                              //   Inicialización de EEPROM Externa.
  write_ext_eeprom(Piso*RegPisoSize+0,RegPiso.Hi);                //   Escritura de Hora Inicial del Piso Seleccionado.
  write_ext_eeprom(Piso*RegPisoSize+1,RegPiso.Mi);                //   Escritura de Minuto Inicial del Piso Seleccionado.
  write_ext_eeprom(Piso*RegPisoSize+2,RegPiso.Hf);                //   Escritura de Hora Final del Piso Seleccionado.
  write_ext_eeprom(Piso*RegPisoSize+3,RegPiso.Mf);                //   Escritura de Minuto Final del Piso Seleccionado.
 }

void WriteRegPisoCode(unsigned char Piso)                         //   Escritura de Clave de Acceso en EEPROM Externa por
Piso.
 {Piso--;                                                         //   Es Restado 1 el Piso de LLamada Debido a que
Ingresa
                                                                  // con Valor Excedido en 1.
  init_ext_eeprom();                                              //   Inicialización de EEPROM Externa.
  write_ext_eeprom(Piso*RegPisoSize+4,RegPiso.code[0]);           //   Escritura de Primer Digito de Clave del Piso
Seleccionado.
  write_ext_eeprom(Piso*RegPisoSize+5,RegPiso.code[1]);           //   Escritura de Segundo Digito de Clave del Piso
Seleccionado.
  write_ext_eeprom(Piso*RegPisoSize+6,RegPiso.code[2]);           //   Escritura de Tercer Digito de Clave del Piso
Seleccionado.
  write_ext_eeprom(Piso*RegPisoSize+7,RegPiso.code[3]);           //   Escritura de Cuarto Digito de Clave del Piso
Seleccionado.
 }
//*****************************************************************************//
//*****************************************************************************//
char tecla_presionada(unsigned int key){                          //   Verificación de Tecla Pulsada en Teclado de 3
Pulsadores.
  unsigned int i;

  for (i=0; i<5; i++){ delay_ms(5); if (input(key)==0) return 0; }//   Espera 25 ms Para Verificar que Hubo una Tecla
Pulsada,
  return 1;                                                       // y de Ser Así Retorna dicha Verificación (Return 1).
}
//*****************************************************************************//
//*****************************************************************************//
char tecla_liberada(unsigned int key){                            //   Verificación de Tecla Liberada en Teclado de 3
Pulsadores.
  unsigned int i;

  for (i=0; i<5; i++){ delay_ms(5); if (input(key)!=0) return 0; }//   Espera 25 ms Para Verificar que la Tecla fue
Liberada,
  return 1;                                                       // y de Ser Así Retorna dicha Verificación (Return 1).
}
//*****************************************************************************//
//*****************************************************************************//
void calibrar(void)                                               //   Sub-Menú de Configuración de RTC Externo.
 {menu_rtc=1;                                                     //   Variable De Selección Entre Menú de Conf. de RTC.
                                                                  //   Nota: La Variable "conf_datos" Será Utilizada Para
                                                                  // Conf. de Datos.

  while (menu_rtc==1){                                            //   Sub-Menú de Configuración de Segundos.
    seg = ' ';
    lcd_gotoxy(1,1);
    printf(lcd_putc,"Fecha:%02d/%02d/20%02d" dia,mes,a_o );       //   Se Muestra la Hora y Fecha En la LCD Colocadas por
Defecto
    printf(lcd_putc,"\nHora: %02d:%02d:%02c "hor,min,seg);       // Pero el la Posición Correspondiente a los Segundos es
Puesto
    delay_ms(300);                                                // un Espacio en Blanco por un Tiempo, y Luego su
Respectivo
    seg = data[0] = conf_datos;                                   // Valor Actual Durante Otro Tiempo a Fin de Producir el
Efécto
    printf(lcd_putc,"\nHora: %02d:%02d:%02d "hor,min,seg);       // de Parpadeo en la Configuración Dada.
    delay_ms(300);

    if (tecla_presionada(EDITAR)){                                //   Si es Pulsada la Tecla EDITAR es Incrementado en 1
el
      conf_datos++;                                               // Valor Correspondiente al Segundo.
      if(conf_datos>59){conf_datos=0;}                            //   Si el Valor del Segundo es Mayor a 59, Este es
Llevado
    }                                                             // Nuevamente a 0.

    if (tecla_presionada(MENU)){                                  //   Si es Pulsada la Tecla MENU es Incrementada la
Variable
      delay_ms(300);                                              // menu_rtc a Fin de Saltar a la Conf. de Minutos.
      menu_rtc++; conf_datos=0;                                   //   La Variable "conf_datos" es Puesta Nuevamente en 0
Para
     }                                                            // los Minutos.
    }
```

```
  while (menu_rtc==2){                                    //   Sub-Menú de Configuración de Minutos.
    min = ' ';
    lcd_gotoxy(1,1);
    printf(lcd_putc,"Fecha:%02d/%02d/20%02d" dia,mes,a_o );   //   Se Muestra la Hora y Fecha En la LCD Colocadas por
Defecto
    printf(lcd_putc,"\nHora: %02d:%02c:%02d "hor,min,seg);   // Pero el la Posición Correspondiente a los Minutos es
Puesto
    delay_ms(300);                                        // un Espacio en Blanco por un Tiempo, y Luego su
Respectivo
    min = data[1] = conf_datos;                           // Valor Actual Durante Otro Tiempo a Fin de Producir el
Efécto
    printf(lcd_putc,"\nHora: %02d:%02d:%02d "hor,min,seg);   // de Parpadeo en la Configuración Dada.
    delay_ms(300);

    if (tecla_presionada(EDITAR)){                        //   Si es Pulsada la Tecla EDITAR es Incrementado en 1
el
      conf_datos++;                                       // Valor Correspondiente al Minuto.
      if(conf_datos>59){conf_datos=0;}                    //   Si el Valor del Minuto es Mayor a 59, Este es
Llevado
    }                                                     // Nuevamente a 0.

    if (tecla_presionada(MENU)){                          //   Si es Pulsada la Tecla MENU es Incrementada la
Variable
      delay_ms(300);                                      // menu_rtc a Fin de Saltar a la Conf. de Hora.
      menu_rtc++; conf_datos=0;                           //   La Variable "conf_datos" es Puesta Nuevamente en 0
Para
      }                                                   // la Hora.
    }

  while (menu_rtc==3){                                    //   Sub-Menú de Configuración de Hora.
    hor = ' ';
    lcd_gotoxy(1,1);
    printf(lcd_putc,"Fecha:%02d/%02d/20%02d" dia,mes,a_o);   //   Se Muestra la Hora y Fecha En la LCD Colocadas por
Defecto
    printf(lcd_putc,"\nHora: %02c:%02d:%02d "hor,min,seg);   // Pero el la Posición Correspondiente a la Hora es
Puesto
    delay_ms(300);                                        // un Espacio en Blanco por un Tiempo, y Luego su
Respectivo
    hor = data[2] = conf_datos;                           // Valor Actual Durante Otro Tiempo a Fin de Producir el
Efécto
    printf(lcd_putc,"\nHora: %02d:%02d:%02d" hor,min,seg);   // de Parpadeo en la Configuración Dada.
    delay_ms(300);

    if (tecla_presionada(EDITAR)){                        //   Si es Pulsada la Tecla EDITAR es Incrementado en 1
el
      conf_datos++;                                       // Valor Correspondiente a la Hora.
      if(conf_datos>23){conf_datos=0;}                    //   Si el Valor de la Hora es Mayor a 23, Este es
Llevado
    }                                                     // Nuevamente a 0.

    if (tecla_presionada(MENU)){                          //   Si es Pulsada la Tecla MENU es Incrementada la
Variable
      delay_ms(300);                                      // menu_rtc a Fin de Saltar a la Conf. de Año.
      menu_rtc++; conf_datos=0;                           //   La Variable "conf_datos" es Puesta Nuevamente en 0
Para
      }                                                   // el Año.
    }

  while (menu_rtc==4){                                    //   Sub-Menú de Configuración de Año.
    a_o = ' ';
    lcd_gotoxy(1,1);
    printf(lcd_putc,"Fecha:%02d/%02d/20%02c" dia,mes,a_o);   //   Se Muestra la Hora y Fecha En la LCD Colocadas por
Defecto
    printf(lcd_putc,"\nHora: %02d:%02d:%02d "hor,min,seg);   // Pero el la Posición Correspondiente al Año es Puesto
    delay_ms(300);                                        // un Espacio en Blanco por un Tiempo, y Luego su
Respectivo
    a_o = data[6] = conf_datos;                           // Valor Actual Durante Otro Tiempo a Fin de Producir el
Efécto
    lcd_gotoxy(1,1);                                      // de Parpadeo en la Configuración Dada.
    printf(lcd_putc,"Fecha:%02d/%02d/20%02d" dia,mes,a_o);
    delay_ms(300);

    if (tecla_presionada(EDITAR)){                        //   Si es Pulsada la Tecla EDITAR es Incrementado en 1
el
      conf_datos++;                                       // Valor Correspondiente al Año.
      if(conf_datos>99){conf_datos=0;}                    //   Si el Valor del Año es Mayor a 99 (2099), Este es
Llevado
    }                                                     // Nuevamente a 0.

    if (tecla_presionada(MENU)){                          //   Si es Pulsada la Tecla MENU es Incrementada la
Variable
      delay_ms(300);                                      // menu_rtc a Fin de Saltar a la Conf. de Mes.
      menu_rtc++; conf_datos=1;                           //   La Variable "conf_datos" es Puesta en 1 Para el
Mes.
      }
    }

  while (menu_rtc==5){                                    //   Sub-Menú de Configuración de Mes.
    mes = ' ';
    lcd_gotoxy(1,1);
    printf(lcd_putc,"Fecha:%02d/%02c/20%02d" dia,mes,a_o);   //   Se Muestra la Hora y Fecha En la LCD Colocadas por
Defecto
    printf(lcd_putc,"\nHora: %02d:%02d:%02d "hor,min,seg);   // Pero el la Posición Correspondiente al Mes es Puesto
    delay_ms(300);                                        // un Espacio en Blanco por un Tiempo, y Luego su
Respectivo
    mes = data[5] = conf_datos;                           // Valor Actual Durante Otro Tiempo a Fin de Producir el
Efécto
    lcd_gotoxy(1,1);                                      // de Parpadeo en la Configuración Dada.
    printf(lcd_putc,"Fecha:%02d/%02d/20%02d" dia,mes,a_o );
    delay_ms(300);

    if (tecla_presionada(EDITAR)){                        //   Si es Pulsada la Tecla EDITAR es Incrementado en 1
el
      conf_datos++;                                       // Valor Correspondiente al Mes.
      if(conf_datos>12){conf_datos=1;}                    //   Si el Valor del Mes es Mayor a 12, Este es Llevado
    }                                                     // Nuevamente a 1.

    if (tecla_presionada(MENU)){                          //   Si es Pulsada la Tecla MENU es Incrementada la
Variable
```

```
      delay_ms(300);                                                  // menu_rtc a Fin de Saltar a la Conf. de Dia.
      menu_rtc++; conf_datos=1;                                       //   La Variable "conf_datos" es Puesta en 1 Para el
Dia.
      }
    }

  while (menu_rtc==6){                                                //   Sub-Menú de Configuración del Dia.
    dia = ' ';
    lcd_gotoxy(1,1);
    printf(lcd_putc,"Fecha:%02c/%02d/20%02d" dia,mes,a_o);            //   Se Muestra la Hora y Fecha En la LCD Colocadas por
Defecto
    printf(lcd_putc,"\nHora: %02d:%02d:%02d" hor,min,seg);           // Pero el la Posición Correspondiente al Dia es Puesto
    delay_ms(300);                                                    // un Espacio en Blanco por un Tiempo, y Luego su
Respectivo
    dia = data[4] = conf_datos;                                       // Valor Actual Durante Otro Tiempo a Fin de Producir el
Efécto
    lcd_gotoxy(1,1);                                                  // de Parpadeo en la Configuración Dada.
    printf(lcd_putc,"Fecha:%02d/%02d/20%02d" dia,mes,a_o );
    delay_ms(300);

    if (tecla_presionada(EDITAR)){                                    //   Si es Pulsada la Tecla EDITAR es Incrementado en 1
el
     conf_datos++;                                                    // Valor Correspondiente al Dia.
     if ( ((mes == 2) && (conf_datos>28) && (a_o%4!=0))               //   Si es Febrero y Año Común, el Dia Llega a 28.
       || ((mes == 2) && (conf_datos>29) && (a_o%4==0))               //   Si es Febrero y Año Bisiesto, el Dia Llega a 29.
       || ((mes==4||mes==6||mes==9||mes==11) && (conf_datos>30))       //   Si es Abril, Junio, Septiembre o Noviembre, el Dia
Llega
       || (conf_datos>31) ) {conf_datos=1;}                           // a 30, Mientras que en los Meses Restantes Llega a 31.
    }                                                                 //   Si el Dia Llega al Tope en Cualquiera de los Casos,
                                                                      // Este es Llevado Nuevamente a 1.

    if (tecla_presionada(MENU))                                       //   Si es Pulsada la Tecla MENU es Incrementada la
Variable
      {delay_ms(300);                                                 // menu_rtc a Fin de Salir del Menú de Conf. del RTC.
       menu_rtc++;  conf_datos=0;                                     //   "conf_datos" es Puesta en 0 Para su Posterior
Reingreso.
       escribir_ds1307();                                             //   Son Escritos los Datos en el RTC Externo.
       lcd_putc('\f');                                                //   Es Borrada la Pantalla LCD.
      }
    }
 }
//************************************************************************//
//************************************************************************//
int ingrese_piso()                                                    //   Sub-Menú de Ingreso de Piso.
  {int posicion1=0, posicion2=0, posicion_total;                      //   Variables Correspondientes a Decena, Unidad y Piso
   int c=0, cuenta=0;                                                 //  Seleccionado, Para Impresión en Pantalla y Retorno
de Valor.
   char n_piso;
   salir=0;

   printf(lcd_putc,"\f  INGRESE PISO\n");                             //   Muestra en la LCD una Petición de Ingreso de Piso.
   printf(lcd_putc," Y PRESIONE '#'");
   delay_ms(2000);
   printf(lcd_putc,"\fPISO: ");                                       //   Queda en Espera del Ingreso del Piso.
   while (c==0)                                                       //   Mientras la Variable "c" es 0, Ejecuta Éste Menú.
     {n_piso=kbd_getc();                                              //   Guarda en n_piso el Piso Pulsado.
      if (tecla_presionada(MENU)){salir=1;}                           //   Si Tecla MENU es Presionada Sale del Menú de
Ingreso de Piso.
      if((n_piso!=0)&&(n_piso!='*'))                                  //   Da Paso Siempre que la Tecla Pulsada sea Diferente
de "*".
       {cuenta++;                                                     //   Permite Saber Cuantas Veces se Ha Pulsado el
Teclado.

       if ((((n_piso=='#')||(n_piso=='0'))&&(cuenta==1))               //   Si la 1era Tecla Pulsada fue "#" ó "0", o la 3era
Tecla
           ||(n_piso!='#')&&(cuenta==3))                              // no fue "#", Envia un Mensaje de Piso Incorrecto.
Luego
           {printf(lcd_putc,"\f PISO INCORRECTO");                    // Reinicia las Variables del Menú Para Solicitar
Nuevamente
            delay_ms(1000);                                           // el Ingreso de un Piso.
            posicion1=0;
            posicion2=0;
            cuenta=0;
            printf(lcd_putc,"\f  INGRESE PISO\n");
            printf(lcd_putc," Y PRESIONE '#'");
            delay_ms(2000);
            printf(lcd_putc,"\fPISO: ");
           }
       if ((cuenta==1)&&(n_piso!='#'))                                //   Si la 1era Tecla Pulsada es Diferente de "#" Dicho
Valor
           {posicion1=n_piso-48;}                                     // es Convertido a Decimal Para ser Guardado.
       if ((cuenta==2)&&(n_piso!='#'))                                //   Si la 2da Tecla Pulsada es Diferente de "#" Dicho
Valor
           {posicion2=n_piso-48;                                      // es Convertido a Decimal Para ser Guardado, y a su vez
el 1er
            posicion1=posicion1*10;                                   // Valor Almacenado es Multiplicado por 10.
           }
       if((n_piso=='#')&&(cuenta>1))                                  //   Si la 2da o 3era Tecla Pulsada es Igual a "#" se
Procede
           {posicion_total = posicion1 + posicion2;                   // a la Suma de los 2 Primeros Valores Guardados, Para
Poder
            c=1;                                                      // así Tener el Piso que fue Presionado. Luego la
Variable "c"
            cuenta=0;                                                 // es Puesta en 1 Para Indicar que se Debe Salir de Este
Menú
            posicion1=0;                                              // y las Variables Asociadas al Mismo son Puestas en 0.
            posicion2=0;
           }
       if((posicion1+posicion2)!=0)                                   //   Muestra en Pantalla el Piso Sólo Cuando es
Diferente
           {printf(lcd_putc,"\fPISO: %u",posicion1+posicion2);}       // de 0.
       }
      if(salir==1){return(0);}                                        //   Si fue Seleccionado Salir el Menú, no Será
Retornado
     }                                                                // Ningún Valor de Piso.
     return(posicion_total);                                          //   Retorna el Piso si fue Correcto.
  }
//************************************************************************//
//************************************************************************//
```

```
void ingrese_clave(byte x)                                          //   Sub-Menú de Ingreso de Código de Acceso.
 {int i=0;
  char k;
  salir=0;
  PB=0;

  while((i<4)&&(salir==0))                                          //   Se Ejecuta Ésta Acción Mientras que no Sean
Discados los
                                                                    // 4 Digitos de Clave y la Variable "salir" sea Igual a
0.
   {k=kbd_getc();                                                   //   Guarda en la Variable "k" la Tecla Pulsada.
    if (tecla_presionada(MENU)){salir=1;}                           //   Si Tecla MENU es Pulsada Sale del Menú de Ingreso
de Clave.
    if (((x==5)||(x==6))&&(k=='*')){salir=1;}                                //   Si Tecla "*" es Pulsada Mientras Está en
Menú de Acceso,
    if ((x==6)&&(i==0)&&(k=='#')){PB=1; i=5;}                                //   Sale del Menú de Ingreso de Clave.

    if ((k!=0)&&(k!='*')&&(k!='#'))                                 //   Ingresa si la Tecla Pulsada es Diferente de "*" y
"#".
       {if(x==1){clave1[i]=k;}                                      //   Vector Donde es Guardada Clave que Permite
Modificar Códigos.
        if(x==2){clave2[i]=k;}                                      //   Vector Donde es Guardada Nueva Clave.
        if(x==3){clave3[i]=k;}                                      //   Vector Donde es Guardada Confirmación de Nueva
Clave.
        if(x==4){clave4[i]=k;}                                      //   Vector Donde es Guardada Clave que Permite Guardar
y Borrar
                                                                    // Horas, así Como Borrar Clave de Acceso.
        if(x==5){clave5[i]=k;}                                      //   Vector Donde es Guardada Clave que Permite Ingreso
a Piso.
        if(x==6){clave6[i]=k;}                                      //   Vector Donde es Guardada Clave que Permite Ingreso
a Piso.
        i++;                                                        //   Incremento en 1 de Digito Pulsado.
        lcd_gotoxy(1,2);
    if (i==1){printf(lcd_putc,"    *");}
     if (i==2){printf(lcd_putc,"    * *");}                         //   Efécto de Ingreso de Clave sin que sea Vista la
Numeración.
      if (i==3){printf(lcd_putc,"    * * *");}
       if (i==4){printf(lcd_putc,"    * * * *"); delay_ms(100);}
       }
   }
 }
//*****************************************************************************//
//*****************************************************************************//
void guarda_codigo(void)                                            //   Sub-Menú de Guardar Código de Acceso.
  {int posicion_total;
   int q=0, repite=0, clave_numero=0;
   byte existe=0;
   lcd_init();
   kbd_init();

   posicion_total=ingrese_piso();                                   //   Llamada a Sub-Código Para Solicitud de Piso.
   ReadRegPiso(posicion_total);                                     //   Lee la Posición de Memoria Correspondiente al Piso
Pulsado.

   if(salir==0){

   if(RegPiso.code[0]!=0xff)                                        //   De Poseer Clave el Piso, se Muestra un Mensaje de
Solicitud
     {printf(lcd_putc,"\f INGRESE CLAVE");                          // de Ingreso de Código y se es Llamado el Sub-Código
Para
      ingrese_clave(1);                                             // el Ingreso del Mismo.
     }

   if(((clave1[0]==RegPiso.code[0])&&(clave1[1]==RegPiso.code[1])   //   Si la Clave Discada es Correcta y no fue Presionada
la Tecla
     &&(clave1[2]==RegPiso.code[2])&&(clave1[3]==RegPiso.code[3])&&(salir==0))   // de Cancelar, o Bien si el Piso no Poseía Clave de
Acceso
      ||(RegPiso.code[0]==0xff))                                    // se Muestra un Mensaje de Petición de Ingreso de Nueva
Clave,
     {while((repite==0)&&(salir==0))                                // Se Ejecuta la Función que Permite tal Discado y una
vez
       {printf(lcd_putc,"\f  NUEVA CLAVE");                         // Terminado es Mostrado en Pantalla un Mensaje de
Petición Para
        ingrese_clave(2);                                           // Confirmar la Nueva Clave Ingresada.
        printf(lcd_putc,"\f CONFIRME CLAVE");
        ingrese_clave(3);

        if ((clave2[0]==clave3[0])&&(clave2[1]==clave3[1])           //   Si las 2 Últimas Claves Discadas Coinciden, Permite
          &&(clave2[2]==clave3[2])&&(clave2[3]==clave3[3])&&(salir==0))      // el Ingreso a Ésta Operación.
          {existe=0;
           clave_numero=0;
           while((existe==0)&&(clave_numero<=10))                   //   Verifica si la Clave Discada ya Existe en ALgún
               {ReadRegPiso(++clave_numero);                        // Otro Piso,de Existir se Asigna a la Variable "existe"
                if( (clave2[0]==RegPiso.code[0])                    // un 1, de lo Contrario le es Asignado un 0.
                  &&(clave2[1]==RegPiso.code[1])
                  &&(clave2[2]==RegPiso.code[2])
                  &&(clave2[3]==RegPiso.code[3]) )
                  {existe=1;}
               }
           if(existe==0)
             {while(q<4)
               {RegPiso.code[q]=clave2[q];
                q++;
               }
              repite=1;                                             //   Sale del Ciclo de Repetición de Verificación.
              WriteRegPisoCode(posicion_total);                     //   Escribe en la Memoria EEPROM Externa la Clave
Discada
              printf(lcd_putc,"\f CLAVE GUARDADA\n");               // Sustituyendo (de Existir) la Clave Anterior.
              delay_ms(1000);
             }
          if(existe==1)
            {printf(lcd_putc,"\f CLAVE YA EXISTE\n");                //   Si Existe una Clave se Muestra Este Mensaje.
             printf(lcd_putc," FAVOR VERIFIQUE");
             delay_ms(2000);
            }
          }
         else if(salir==0)                                          //   Si no fue Pulsada la Tecla de Cancelar y las Claves
           {printf(lcd_putc,"\f     ERROR!\n");                     // no Coinciden, se Muestra un Mensaje de Error Para que
```

```
           printf(lcd_putc," FAVOR VERIFIQUE");                              // el Usuario Verifique el Discado de Tales Claves de
Forma
           delay_ms(1000);                                                   // Correcta.
          }
        }
     }
   else if(salir==0)                                                         //   Si no fue Pulsada la Tecla de Cancelar y la Clave
Para
         {printf(lcd_putc,"\f CLAVE INVALIDA");                              // el Ingreso a Modificación de la Misma fue Errada, se
          delay_ms(1000);                                                    // Muestra un Mensaje en Pantalla de Clave Invalida y se
Sale
         }                                                                   // del Presente Menú.
     }
   repite=0;
  }
//***************************************************************************//
//***************************************************************************//
void conf_hor_min(byte llevo)                                                //   Sub-Menú de Configuración de Horas de Acceso.
  {int d=0, sigue=0;
   int hidec=0, midec=0, hit=0, mit=0;
   char ingrese;

   while ((d==0)&&(salir==0))                                                //   Mientras la Variable "d" es 0 y no se Haya Pulsado
Cancelar
                                                                             // se Ejecuta Ésta Función.
    {ingrese=kbd_getc();                                                     //   Guarda en Ingrese la Tecla Pulsada.
     if (tecla_presionada(MENU)){salir=1;}                                   //   Si la Tecla Menú es Pulsada, se Cancela Ésta
Operación.
     if ((ingrese!=0)&&(ingrese!='#'))                                       //   Permite el Ingreso si la Tecla Pulsada es Diferente
de "#".
      {sigue++;                                                              //   Lleva la Cuenta de Cuantas Teclas Han Sido
Pulsadas.
       if(ingrese=='*')                                                      //   Si la Tecla Pulsada es "*" es Borrada la Segunda
Fila
        {hidec=0;                                                            // de la Pantalla LCD, y las Variables Asociadas son
Puestas
         midec=0;                                                            // Nuevamente en 0 Para Permitir Nuevamente el Ingreso
de Horas
         hit=0;                                                              // de Acceso Hacia el Solicitado.
         mit=0;
         sigue=0;
         lcd_gotoxy(1,2);
         printf(lcd_putc,"               ");
        }
       else if( (sigue==1)&&( (ingrese=='0')||(ingrese=='1')||(ingrese=='2') ) ) //   Delimita la 1era Tecla Pulsada Hasta 2 (Factor de
20 Horas).
        {hidec=ingrese-48;                                                   //   Convierte la Tecla Pulsada a su Valor Decimal.
         lcd_gotoxy(6,2);
         printf(lcd_putc,"%u :",hidec);                                      //   Muestra Dicho Valor en la Pantalla LCD.
        }
       else if( (sigue==2)&&(hidec+48=='0') )                                //   Si es Pulsada la 2da Tecla y la 1era Tecla
Presionada es 0,
        {hit=ingrese-48;                                                     // la Tecla Anterior no se Modifica.
         if(llevo==1){RegPiso.Hi=hit;}                                       //   Si "llevo=1" se Pide Ingreso de Hora Inicial, de lo
         else {RegPiso.Hf=hit;}                                              // Contrario se Pide Hora Final. Luego la Misma es
Guardada.
         lcd_gotoxy(6,2);
         printf(lcd_putc,"%02u:",hit);                                       //   Muestra Dicho Valor en la Pantalla LCD.
        }
       else if( (sigue==2)&&(hidec+48=='1') )                                //   Si es Pulsada la 2da Tecla y la 1era Tecla
Presionada es 1,
        {hit=hidec*10+ingrese-48;                                            // la Tecla Anterior se Multiplica por 10 y se Suma a la
2da.
         if(llevo==1){RegPiso.Hi=hit;}                                       //   Si "llevo=1" se Pide Ingreso de Hora Inicial, de lo
         else {RegPiso.Hf=hit;}                                              // Contrario se Pide Hora Final. Luego la Misma es
Guardada.
         lcd_gotoxy(6,2);
         printf(lcd_putc,"%02u:",hit);                                       //   Muestra Dicho Valor en la Pantalla LCD.
        }
       else if( (sigue==2)&&(hidec+48=='2')                                  //   Si es Pulsada la 2da Tecla y la 1era Tecla
Presionada es 2,
         &&( (ingrese=='0')||(ingrese=='1')||(ingrese=='2')||(ingrese=='3') ) )  // se Delimita esta 2da Hasta 3 (Sólo 23 Horas), y la
Tecla
        {hit=hidec*10+ingrese-48;                                            // Anterior (1era) se Multiplica por 10 y se Suma a la
2da.
         if(llevo==1){RegPiso.Hi=hit;}                                       //   Si "llevo=1" se Pide Ingreso de Hora Inicial, de lo
         else {RegPiso.Hf=hit;}                                              // Contrario se Pide Hora Final. Luego la Misma es
Guardada.
         lcd_gotoxy(6,2);
         printf(lcd_putc,"%02u:",hit);                                       //   Muestra Dicho Valor en la Pantalla LCD.
        }
       else if( (sigue==3)&&(ingrese!='6')&&(ingrese!='7')                   //   Delimita la 3era Tecla Pulsada a Menos de 6 (Sólo
59 Minutos).
         &&(ingrese!='8')&&(ingrese!='9') )
        {midec=ingrese-48;                                                   //   Convierte la Tecla Discada a su Valor Decimal.
         lcd_gotoxy(8,2);
         printf(lcd_putc,":%u",midec);                                       //   Muestra Dicho Valor en la Pantalla LCD.
        }
       else if( (sigue==4)&&(midec+48=='0')&&(ingrese!=0)&&(ingrese!='*') )   //   Si es Pulsada la 4ta Tecla y la 3era Tecla
Presionada es 0,
        {mit=ingrese-48;                                                     // Ésta 3era no se Modifica.
         if(llevo==1){RegPiso.Mi=mit;}                                       //   Si "llevo=1" se Pide Ingreso de Minuto Inicial, de
lo
         else {RegPiso.Mf=mit;}                                              // Contrario se Pide Minuto Final. Luego la Misma es
Guardada.
         lcd_gotoxy(8,2);
         printf(lcd_putc,":%02u",mit);                                       //   Muestra Dicho Valor en la Pantalla LCD.
         delay_ms(500);
         d=1;
        }
       else if( (sigue==4)&&(ingrese!=0)&&(ingrese!='*') )                   //   Si es Pulsada la 4ta Tecla y es Diferente de "*",
la 3era
        {mit=midec*10+ingrese-48;                                            // Tecla Discada se Multiplica por 10 y se Suma a la
4ta.
         if(llevo==1){RegPiso.Mi=mit;}                                       //   Si "llevo=1" se Pide Ingreso de Minuto Inicial, de
lo
         else {RegPiso.Mf=mit;}                                              // Contrario se Pide Minuto Final. Luego la Misma es
Guardada.
         lcd_gotoxy(8,2);
```

```
         printf(lcd_putc,":%02u",mit);                                  //   Muestra Dicho Valor en la Pantalla LCD.
         delay_ms(500);
         d=1;                                                           //   La Variable "d" es Puesta en 1 Para Indicar que se
Debe
        }                                                               // Salir de Ésta Función.
       else {sigue--;}
      }
    }
  }
//*************************************************************************//
//*************************************************************************//
void guarda_hora(void)                                                  //   Sub-Menú de Guardar Hora de Acceso.
  {int posicion_total;
   lcd_init();
   kbd_init();

   posicion_total=ingrese_piso();                                       //   Llamada a Sub-Código Para Solicitud de Piso.
   ReadRegPiso(posicion_total);                                         //   Lee la Posición de Memoria Correspondiente al Piso
Pulsado.

   if(salir==0){

   if(RegPiso.code[0]!=0xff)                                            //   De Poseer Clave el Piso, se Muestra un Mensaje de
Solicitud
     {printf(lcd_putc,"\f INGRESE CLAVE");                              // de Ingreso de Código y se es Llamado el Sub-Código
Para
      ingrese_clave(4);                                                 // el Ingreso del Mismo.
     }

   if(((clave4[0]==RegPiso.code[0])&&(clave4[1]==RegPiso.code[1])       //   Si la Clave Discada es Correcta y no fue Presionada
la Tecla
     &&(clave4[2]==RegPiso.code[2])&&(clave4[3]==RegPiso.code[3])&&(salir==0))   // de Cancelar, o Bien si el Piso no Poseía Clave de
Acceso
     ||(RegPiso.code[0]==0xff))                                         // se Muestra un Mensaje de Petición de Ingreso de Hora
de Acceso
     {printf(lcd_putc,"\fHORA INICIAL:");                               // Inicial, se Ejecuta la Función que Permite tal
Discado y una
      conf_hor_min(1);                                                  // vez Terminado es Mostrado en Pantalla un Mensaje de
Petición
      printf(lcd_putc,"\fHORA FINAL:");                                 // de Ingreso de Hora de Acceso Final, Ejecutando
Nuevamente
      conf_hor_min(2);                                                  // la Acción Anterior.
      if(salir==0)
        {WriteRegPisoHoras(posicion_total);                             //   Se Escribe en la Posición de la EEPROM Externa
Correspondiente
         printf(lcd_putc,"\f    GUARDADO!");                            // al Piso Seleccionado las Horas Ingresadas, y
Posteriormente
         delay_ms(1000);                                                // se Muestra en la LCD un Mensaje Indicando que ha Sido
Guardada.
        }
     }
   else if(salir==0)                                                    //   Si no fue Pulsada la Tecla de Cancelar y la Clave
Para
         {printf(lcd_putc,"\f CLAVE INVALIDA");                         // el Ingreso fue Errada, se Muestra un Mensaje en
Pantalla
          delay_ms(1000);                                               // de Clave Invalida y se Sale del Presente Menú.
         }
     }
  }
//*************************************************************************//
//*************************************************************************//
void borrar_codigo(void)                                                //   Sub-Menú de Borrar Código de Acceso.
  {int posicion_total, q=0;
   lcd_init();
   kbd_init();

   posicion_total=ingrese_piso();                                       //   Llamada a Sub-Código Para Solicitud de Piso.
   ReadRegPiso(posicion_total);                                         //   Lee la Posición de Memoria Correspondiente al Piso
Pulsado.

   if(salir==0){

   if(RegPiso.code[0]!=0xff)                                            //   De Poseer Clave el Piso, se Muestra un Mensaje de
Solicitud
     {printf(lcd_putc,"\f INGRESE CLAVE");                              // de Ingreso de Código y se es Llamado el Sub-Código
Para
      ingrese_clave(4);                                                 // el Ingreso del Mismo.

   if((clave4[0]==RegPiso.code[0])&&(clave4[1]==RegPiso.code[1])        //   Si la Clave Discada es Correcta y no fue Presionada
la Tecla
    &&(clave4[2]==RegPiso.code[2])&&(clave4[3]==RegPiso.code[3])&&(salir==0))    // de Cancelar,se Procede al Ingreso de Ésta Operación.
     {while(q<4)
        {RegPiso.code[q]=0xff;                                          //   Borra los Digitos Correspondientes a la Clave que
Poseía
         q++;                                                           // el Piso Seleccionado.
        }
      WriteRegPisoCode(posicion_total);                                 //   Borra de la Memoria EEPROM Externa la Clave que
Poseía
      printf(lcd_putc,"\fBORRANDO...");                                 // el Piso Seleccionado, Mostrando Posteriormente en
Pantalla
      delay_ms(1000);                                                   // un Mensaje Indicando que la Clave ha Sido Borrada.
      printf(lcd_putc,"\f CLAVE BORRADA!");
      delay_ms(1000);
     }
   else if(salir==0)                                                    //   Si no fue Pulsada la Tecla de Cancelar y la Clave
Para
         {printf(lcd_putc,"\f CLAVE INVALIDA");                         // el Ingreso fue Errada, se Muestra un Mensaje en
Pantalla
          delay_ms(1000);                                               // de Clave Invalida y se Sale del Presente Menú.
         }
     }
   else if(salir==0)                                                    //   Si no fue Pulsada la Tecla de Cancelar y el Piso
         {printf(lcd_putc,"\f   NO EXISTE\n");                          // Seleccionado no Poseía Clave para su Acceso, se
Muestra
          printf(lcd_putc,"CLAVE PROGRAMADA");                          // un Mensaje en Pantalla Indicando que el Piso no Tenía
          delay_ms(1500);                                               // Clave Programada.
         }
    }
}
```

```
//*****************************************************************************//
//*****************************************************************************//
void borrar_hora(void)                                                          //   Sub-Menú de Borrar Horas de Acceso.
  {int posicion_total;
   lcd_init();
   kbd_init();

   posicion_total=ingrese_piso();                                               //   Llamada a Sub-Código Para Solicitud de Piso.
   ReadRegPiso(posicion_total);                                                 //   Lee la Posición de Memoria Correspondiente al Piso
Pulsado.

   if(salir==0){

   if(RegPiso.code[0]!=0xff)                                                    //   De Poseer Clave el Piso, se Muestra un Mensaje de
Solicitud
     {printf(lcd_putc,"\f INGRESE CLAVE");                                      // de Ingreso de Código y se es Llamado el Sub-Código
Para
      ingrese_clave(4);                                                         // el Ingreso del Mismo.
     }

   if(((clave4[0]==RegPiso.code[0])&&(clave4[1]==RegPiso.code[1])               //   Si la Clave Discada es Correcta y no fue Presionada
la Tecla
     &&(clave4[2]==RegPiso.code[2])&&(clave4[3]==RegPiso.code[3])&&(salir==0))   // de Cancelar, o Bien si el Piso no Poseía Clave de
Acceso
     ||(RegPiso.code[0]==0xff))                                                 // se Procede al Ingreso de Ésta Operación.
     {if(RegPiso.Hi==0xff)                                                      //   Si el Piso no Poseía Horas Para su Acceso, se
Muestra
        {printf(lcd_putc,"\f   NO EXISTE\n");                                   // un Mensaje en Pantalla Indicando que el Piso no Tenía
         printf(lcd_putc,"HORA PROGRAMADA");                                    // Hora Programada.
         delay_ms(1500);
        }                                                                       //   Si el Piso Tenía Horas Programadas Para su Acceso,
se
      else{RegPiso.Hi=0xff;                                                     // Borra de la Memoria EEPROM Externa Tales Horas que
Poseía
           RegPiso.Mi=0xff;                                                     // el Piso Seleccionado, Mostrando Posteriormente en
Pantalla
           RegPiso.Hf=0xff;                                                     // un Mensaje Indicando que la Hora ha Sido Borrada.
           RegPiso.Mf=0xff;
           WriteRegPisoHoras(posicion_total);
           printf(lcd_putc,"\fBORRANDO...");
           delay_ms(1000);
           printf(lcd_putc,"\f  HORA BORRADA!");
           delay_ms(1000);
          }
     }
   else if(salir==0)                                                            //   Si no fue Pulsada la Tecla de Cancelar y la Clave
Para
        {printf(lcd_putc,"\f CLAVE INVALIDA");                                  // el Ingreso fue Errada, se Muestra un Mensaje en
Pantalla
         delay_ms(1000);                                                        // de Clave Invalida y se Sale del Presente Menú.
        }
    }
  }
//*****************************************************************************//
//*****************************************************************************//
 enum funciones{cal,cod,hora,b_cod,b_hora,priv,ini};                            //   Funciones Para los Multiples Menú.
//*****************************************************************************//
//*****************************************************************************//
void codigo(void)                                                               //   Sub-Menú de Ingreso de Código de Acceso.
   { printf(lcd_putc, "\fCLAVE...");
     delay_ms(1000);
     guarda_codigo();                                                           //   Llamada de Sub-Menú de Guardar Código de Acceso.
     lcd_putc("\f");
   }
//*****************************************************************************//
//*****************************************************************************//
void horas(void)                                                                //   Sub-Menú de Ingreso de Horas de Acceso.
   { printf(lcd_putc, "\fHORAS...");
     delay_ms(1000);
     guarda_hora();                                                             //   Llamada de Sub-Menú de Guardar Horas de Acceso.
     lcd_putc("\f");
   }
//*****************************************************************************//
//*****************************************************************************//
void run_func(int numfunc)                                                      //   Función que Permite Hacer Cambios Entre Diferentes
Sub-Menú.
 {switch(numfunc)
   {case cal:
       calibrar();                                                              //   Llamada a Sub-Menú de Configuración de RTC Externo.
       break;
    case cod:
       codigo();                                                                //   Llamada a Sub-Menú de Ingreso de Código de Acceso.
       break;
    case hora:
       horas();                                                                 //   Llamada a Sub-Menú de Ingreso de Horas de Acceso.
       break;
    case b_cod:
       borrar_codigo();                                                         //   Llamada a Sub-Menú de Borrar Código de Acceso.
       break;
    case b_hora:
       borrar_hora();                                                           //   Llamada a Sub-Menú de Borrar Horas de Acceso.
       break;
    case priv:
       printf(lcd_putc, "\fINICIANDO...");                                      //   Selección de Sub-Menú de Inicialización del
Programa
       delay_ms(1000);                                                          // Principal, Permitiendo Entrar en el Mismo al Momento
       inicia_programa=2;                                                       // de Asignarle a la Variable "inicia_programa" el Valor
1.
       lcd_putc("\f");
       leer_ds1307(hor,min,seg,date,dia,mes,a_o);                               //   Lee los Datos de Hora y Fecha del RTC Externo.
       break;
    case ini:
       printf(lcd_putc, "\fINICIANDO...");                                      //   Selección de Sub-Menú de Inicialización del
Programa
       delay_ms(1000);                                                          // Principal, Permitiendo Entrar en el Mismo al Momento
       inicia_programa=1;                                                       // de Asignarle a la Variable "inicia_programa" el Valor
1.
       lcd_putc("\f");
       leer_ds1307(hor,min,seg,date,dia,mes,a_o);                               //   Lee los Datos de Hora y Fecha del RTC Externo.
```

```
      break;
   }
 }
//***************************************************************************//
//***************************************************************************//
void acceso(void)                                                  //   Sub-Función de Acceso a Piso Seleccionado.
   {byte hor_correcta=2, clave_correcta=0;                          //   Indicadores de Hora y/o Clave Correcta.
    int  n_piso=0;                                                 //   Número de Piso Discado Luego de ser Llevado a
Decimal.
    long p_pulsado, compara=1;                                     //   Piso Discado Desde la Cabina y Comparador Para
Llevar
    lcd_init();                                                    // el Piso a su Valor en Decimal.
    kbd_init();

    read_expanded_inputs(&p_pulsado);                              //   Es leído el Piso Pulsado y Configurado Para
    p_pulsado=~p_pulsado;                                          // ser Evaluado.

    while(compara!=p_pulsado)                                      //   Conversión del Piso Pulsado en la Cabina Hacia su
Valor
        {compara=compara*2;                                        // en Decimal.
         n_piso++;
        }

   if(n_piso!=0)
     {printf(lcd_putc,"\fPiso Número: %u", n_piso);
      delay_ms(500);
     }
   if(n_piso==0)
     {printf(lcd_putc,"\f  Planta Baja");
      delay_ms(500);
     }

   if(n_piso!=0)
    {ReadRegPiso(n_piso);                                          //   Lee los Datos de la EEPROM Externa Correspondientes
al
                                                                   // Piso Seleccionado.
   if(RegPiso.Hi!=0xff)                                            //   Si el Piso Seleccionado Tiene Hora de Acceso se
Pregunta:
     {if( ((hor>RegPiso.Hi)&&(hor<RegPiso.Hf))                     // - Hora Actual Está Entre Horas de Acceso Inicial y
Final?
       || ((hor==RegPiso.Hi)&&(min>=RegPiso.Mi)&&(hor<RegPiso.Hf)) // - Minuto Actual Mayor a Minuto Inicial Programado?
       || ((hor==RegPiso.Hf)&&(min<=RegPiso.Mf)&&(hor>RegPiso.Hi)) // - Minuto Actual Menor a Minuto Final Programado?
       || ((hor==RegPiso.Hi)&&(hor==RegPiso.Hf)&&(min>=RegPiso.Mi) // - Si Hora Actual es Igual a Hora Inicial y Final
Programado,
        &&  (min<=RegPiso.Mf)) )                                   // - Minuto Actual Está Entre Minutos de Acceso Inicial
y Final?
       {hor_correcta=1;}                                           //   De Cumplirse las Premisas se Toma Como Hora
Correcta.
      else{hor_correcta=0;}                                        //   De no Cumplirse se Toma Como Hora Incorrecta.
     }

   if( ((RegPiso.code[0]!=0xff)&&(RegPiso.Hi==0xff))               //   Si el Piso Seleccionado Tiene Clave y no Posee Hora
de
    || ((RegPiso.code[0]!=0xff)&&(hor_correcta==1)) )               // Acceso o Bien Tiene Clave y fue Considerado Como Hora
Correcta
     {printf(lcd_putc,"\f INGRESE CLAVE\n");                       // se Muestra en la LCD una Solicitud de Ingreso de
Clave y es
      ingrese_clave(5);                                            // Llamado el Sub-Menú Para el Ingreso del Mismo.

      if( (clave5[0]==RegPiso.code[0])&&(clave5[1]==RegPiso.code[1])  //   Si la Clave Discada es Correcta y no fue Presionada
la
      &&(clave5[2]==RegPiso.code[2])&&(clave5[3]==RegPiso.code[3])&&(salir==0) ) // Tecla de Cancelar, se Asigna a clave_correcta el
Valor de 1.
        {clave_correcta=1;}
      else{clave_correcta=0;}                                       //   Si la Clave Discada es Incorrecta, se Asigna a
clave_correcta
     }                                                             // el Valor 0.
    }

   if( ((RegPiso.Hi==0xff)&&(RegPiso.code[0]==0xff))               //   Si el Piso Seleccionado no Posee Hora ni Clave de
Acceso, o
     |||((RegPiso.Hi==0xff)&&(clave_correcta==1))                  // si no Posee Hora y la Clave Discada fue Correcta, o
     |||((hor_correcta==1)&&(clave_correcta==1))                    // si la Hora y Clave de Acceso son Correctas, o
     |||((hor_correcta==1)&&(RegPiso.code[0]==0xff))|||(n_piso==0) )  // si la Hora de Acceso es Correcta y no Posee Clave, se
     {                                                             // Procede a Realizar la Llamada.
       if( clave_correcta==1 )                                     //   Si la Clave Discada fue Correcta (de Tener una) se
Muestra
        {printf(lcd_putc,"\f    ABIERTO!");                        // en la LCD un Mensaje de Abierto.
         delay_ms(1000);
        }
      write_expanded_outputs(&p_pulsado);                          //   Se Envia una Señal al Shift Register 74LS595
Indicando
      delay_ms(500);                                               // el Piso Para que el Mismo Emita una Señal Hacia la
Tarjeta
      p_pulsado=0;                                                 // Principal del Ascensor; luego se Borra la Variable
p_pulsado
      write_expanded_outputs(&p_pulsado);                          // Para Quitar el Pulso del Shift Register.
      output_low(PIN_B5);                                          // Evita que el Shift Register 74LS595 se Active con
Ruído.
     }

   else if((salir==0)&&(hor_correcta==0))
         {printf(lcd_putc,"\f     HORARIO    \n");                //   Si no fue Pulsada la Tecla de Cancelar y la Hora
Para   .
          printf(lcd_putc,  "  NO PERMITIDO  ");                  // el Ingreso fue errada, se Muestra un Mensaje en
Pantalla
          delay_ms(1000);                                          // de Horario no Permitido, Sale del Presente Menú y se
         }                                                         // Prepara Para Recibir una Nueva Llamada.

   else if(salir==0)                                               //   Si no fue Pulsada la Tecla de Cancelar y la Clave
Para
         {printf(lcd_putc,"\f ACCESO DENEGADO");                   // el Ingreso fue Errada, se Muestra un Mensaje en
Pantalla
          delay_ms(1000);                                          // de Acceso Denegado, se Sale del Presente Menú y se
Prepara
         }                                                         // Para Recibir una Nueva Llamada.
   }
//***************************************************************************//
//***************************************************************************//
```

```
void privado(void)                                                    //   Sub-Función de Servicio Privado.
   {byte hor_correcta=2, piso_correcto=0, incorrecto=0;                //   Indicadores de Hora y/o Clave Correcta.
    int  p_asignado=0;                                                //   Número de Piso Discado Luego de ser Configurado.
    long p_pulsado=1, compara=0;                                      //   Valor inicial del Piso y Variable a Comparar.
    lcd_init();
    kbd_init();

    printf(lcd_putc,"\f INGRESE CLAVE\n");                            // se Muestra en la LCD una Solicitud de Ingreso de
Clave y es
    ingrese_clave(6);                                                 // Llamado el Sub-Menú Para el Ingreso del Mismo.

    while((piso_correcto==0)&&(incorrecto==0)&&(p_asignado<=10)&&(PB==0))  //   Una vez Discada la Clave y Sin Ser Pulsada la Tecla
de
     {ReadRegPiso(++p_asignado);                                      // Cancelar, el Sistema Busca en la EEPROM el Piso
      if(p_asignado==10){incorrecto=1;}                               // Asociado a la Clave Suministrada, a No Ser que el
Piso
      if( (clave6[0]==RegPiso.code[0])                                // Haya Sido Planta Baja.
        &&(clave6[1]==RegPiso.code[1])
        &&(clave6[2]==RegPiso.code[2])
        &&(clave6[3]==RegPiso.code[3])&&(salir==0) )
        {piso_correcto=1;}
     }

   if(PB==0)
    {ReadRegPiso(p_asignado);                                         //   Lee los Datos de la EEPROM Externa Correspondientes
al
                                                                      // Piso Seleccionado.
   if(RegPiso.Hi!=0xff)                                               //   Si el Piso Seleccionado Tiene Hora de Acceso se
Pregunta:
     {if( ((hor>RegPiso.Hi)&&(hor<RegPiso.Hf))                        // - Hora Actual Está Entre Horas de Acceso Inicial y
Final?
       || ((hor==RegPiso.Hi)&&(min>=RegPiso.Mi)&&(hor<RegPiso.Hf))    // - Minuto Actual Mayor a Minuto Inicial Programado?
       || ((hor==RegPiso.Hf)&&(min<=RegPiso.Mf)&&(hor>RegPiso.Hi))    // - Minuto Actual Menor a Minuto Final Programado?
       || ((hor==RegPiso.Hi)&&(hor==RegPiso.Hf)&&(min>=RegPiso.Mi)     // - Si Hora Actual es Igual a Hora Inicial y Final
Programado,
        &&  (min<=RegPiso.Mf)) )                                      // - Minuto Actual Está Entre Minutos de Acceso Inicial
y Final?
       {hor_correcta=1;}                                              //   De Cumplirse las Premisas se Toma Como Hora
Correcta.
      else{hor_correcta=0;}                                           //   De no Cumplirse se Toma Como Hora Incorrecta.
     }

   while(compara!=p_asignado)                                         //   Conversión del Piso Asignado Según Clave a su Valor
        {p_pulsado=p_pulsado*2;                                       // Para Ser Enviado.
         compara++;
        }
     }
   if( ((RegPiso.Hi==0xff)&&(incorrecto==0))                          // Si el piso no Posee Hora y la Clave Discada fue
Correcta
     ||((hor_correcta==1)&&(incorrecto==0))                            // o si la Hora y Clave son Correctas, o si Bien fue
     ||( PB==1 ) )                                                    // Pulsado el Botón de Planta Baja, Procede a Realizar
     {                                                                // la Llamada.
      if(piso_correcto==1)
        {printf(lcd_putc,"\fACCESO A PISO %u",p_asignado);
         delay_ms(1000);
        }
      else if(PB==1)
        {printf(lcd_putc,"\f  Planta Baja");
         delay_ms(1000);
        }
      write_expanded_outputs(&p_pulsado);                             //   Se Envia una Señal al Shift Register 74LS595
Indicando
      delay_ms(500);                                                  // el Piso Para que el Mismo Emita una Señal Hacia la
Tarjeta
      p_pulsado=0;                                                    // Principal del Ascensor; luego se Borra la Variable
p_pulsado
      write_expanded_outputs(&p_pulsado);                             // Para Quitar el Pulso del Shift Register.
      output_low(PIN_B5);                                             // Evita que el Shift Register 74LS595 se Active con
Ruído.
     }

   else if((salir==0)&&(hor_correcta==0))
         {printf(lcd_putc,"\f     HORARIO    \n");                   //   Si no fue Pulsada la Tecla de Cancelar y la Hora
Para   .
          printf(lcd_putc,  "  NO PERMITIDO  ");                     // el Ingreso fue errada, se Muestra un Mensaje en
Pantalla
          delay_ms(1000);                                             // de Horario no Permitido, Sale del Presente Menú y se
         }                                                            // Prepara Para Recibir una Nueva Llamada.

   else if(salir==0)                                                  //   Si no fue Pulsada la Tecla de Cancelar y la Clave
Para
         {printf(lcd_putc,"\fCLAVE NO EXISTE");                       // el Ingreso fue Errada, se Muestra un Mensaje en
Pantalla
          delay_ms(1000);                                             // de Clave No Existente, Sale del Presente Menú y se
Prepara
         }                                                            // Para Recibir una Nueva Llamada.
   }
//*****************************************************************************//
//*****************************************************************************//
void main()                                                           //   Programa Principal.
 {char item=0, ir_piso;                                               //   Variable de ambio Entre Menús.
  char n_menus=7;                                                     //   7 Menús Principales Para Mostrar.

  set_tris_B(0x01);                                                   //   Pin B0 Como Entrada (Para Interrupción Externa)
  port_b_pullups(TRUE);                                               //   Resistencias de Pullups del Puerto B Activadas.
  lcd_init();
  kbd_init();

  // for (item=0;item<90;item++){ write_ext_eeprom(item,0xff); }
  item=0;
  while(true){                                                        //   Ciclo Infinito de Ejecución.
   while (inicia_programa==0)                                         //   Mientras no se Inicie la Función Principal, Realiza
Éste Ciclo.
    {if (tecla_presionada(MENU))                                      //   Si la Tecla MENU es Pulsada, se Produce un Salto
Hacia
        {item++;                                                      // el Siguiente Menú.
         delay_ms(300);
         lcd_putc('\f');
        }
```

```
     if (item > (n_menus-1))                                    //   Si se Llegó al Último Menú, se Coloca en 0 la
Variable
        {item=0;}                                               // "item" Para Empezar Nuevamente Desde el Menú Inicial.

     switch (item)
      {case 0:
       lcd_gotoxy(5,1);
       printf(lcd_putc, "AJUSTAR\n");                           //   Muestra en Pantalla la Opción de Ajuste de Hora y
Fecha
       printf(lcd_putc, "  HORA Y FECHA  ");                    // del RTC Externo.
       lcd_gotoxy(5,1);
       break;

       case 1:
       lcd_gotoxy(1,1);
       printf(lcd_putc, "INGRESAR CODIGO \n");                  //   Muestra en Pantalla la Opción de Ingreso de Código
       printf(lcd_putc, "   DE ACCESO    ");                   // de Acceso de un Determinado Piso.
       lcd_gotoxy(1,1);
       break;

       case 2:
       lcd_gotoxy(1,1);
       printf(lcd_putc, " INGRESAR HORAS \n");                  //   Muestra en Pantalla la Opción de Ingreso de Horas
       printf(lcd_putc, "   DE ACCESO    ");                   // de Acceso de un Determinado Piso.
       lcd_gotoxy(1,1);
       break;

       case 3:
       lcd_gotoxy(1,1);
       printf(lcd_putc, "  BORRAR CODIGO \n");                  //   Muestra en Pantalla la Opción de Borrar Código de
       printf(lcd_putc, "   DE ACCESO    ");                   // Acceso de un Determinado Piso.
       lcd_gotoxy(1,1);
       break;

       case 4:
       lcd_gotoxy(1,1);
       printf(lcd_putc, "  BORRAR HORAS  \n");                  //   Muestra en Pantalla la Opción de Borrar Horas de
       printf(lcd_putc, "   DE ACCESO    ");                   // Acceso de un Determinado Piso.
       lcd_gotoxy(1,1);
       break;

       case 5:
       lcd_gotoxy(1,1);
       printf(lcd_putc, "    SERVICIO    \n");                  //   Muestra en Pantalla la Opción de Iniciar la Función
       printf(lcd_putc, "    PRIVADO     ");                   // de Servicio Privado.
       lcd_gotoxy(1,1);
       break;

       case 6:
       lcd_gotoxy(1,1);
       printf(lcd_putc, "  INICIALIZAR   ");                    //   Muestra en Pantalla la Opción de Iniciar la Función
Principal.
       lcd_gotoxy(1,1);
       break;
      }

     if (tecla_presionada(ENTER))                               //   Si la Tecla ENTER es Pulsada, se Genera un Salto
Hacia
       {delay_ms(200);                                          // la Función "run_func" la Cual se Encarga de Realizar
Cambios
        run_func(item);                                         // Entre los Diferentes Sub-Menús de Configuración del
Sistema.
       }
    }

    if(inicia_programa==1)
      {setup_timer_1(T1_DISABLED);                              //   El Timer 1 es Desactivado.
       ext_int_edge(L_TO_H);                                    //   Se Consideran Pulsos en Alto.
       enable_interrupts(int_ext);                              //   La Interrupción Externa es Activada, ya que se
Ejecutará
       enable_interrupts(GLOBAL);                               // la Función Principal de la Interfaz.
      }

    while(inicia_programa==1)                                   //   Ciclo de Inicio de la Función Principal.
     {delay_ms(100);
      leer_ds1307(hor,min,seg,date,dia,mes,a_o);                //   Lee los Datos de Hora y Fecha del RTC Externo.
      lcd_gotoxy(1,1);
      printf(lcd_putc,"Fecha:%02d/%02d/20%02d" dia,mes,a_o );   //   Muestra en la LCD el Dia, Mes y Año Actualizados.
      lcd_gotoxy(1,2);
      printf(lcd_putc,"Hora: %02d:%02d:%02d "hor,min,seg);      //   Muestra en la LCD la Hora, Minutos y Segundos
Actualizados.
      if (tecla_presionada(MENU)){inicia_programa=0;}           //   Si la Tecla MENU es Presionada, se Salta Hacia
Configuración.
      if (veces_pulsado==numero_pulsos){ veces_pulsado=0; acceso(); }   //   Si se Verifica la Existencia de una Llamadade
Cabina, se
     }                                                          // Realiza un Salto Hacia la Sub-Función "acceso" Para
ser Ejecutada.

    while(inicia_programa==2)                                   //   Ciclo de Inicio de la Función Principal.
     {delay_ms(100);
      leer_ds1307(hor,min,seg,date,dia,mes,a_o);                //   Lee los Datos de Hora y Fecha del RTC Externo.
      lcd_gotoxy(1,1);
      printf(lcd_putc,"Fecha:%02d/%02d/20%02d" dia,mes,a_o );   //   Muestra en la LCD el Dia, Mes y Año Actualizados.
      lcd_gotoxy(1,2);
      printf(lcd_putc,"Hora: %02d:%02d:%02d "hor,min,seg);      //   Muestra en la LCD la Hora, Minutos y Segundos
Actualizados.
      if (tecla_presionada(MENU)){inicia_programa=0;}           //   Si la Tecla MENU es Presionada, se Salta Hacia
Configuración.
      ir_piso=kbd_getc();                                       //   Si Realizan una Petición de Llamada, se Realiza un
Salto
      if (tecla_presionada(ENTER)||(ir_piso=='#')){privado();}  //   Hacia la Función de Servicio Privado.
     }
   }
 }
```

24128.c

```
#define EEPROM_ADDRESS long int
#define EEPROM_SIZE    16384

#define CHIP_SELECT 0
#define WP PIN_E0

void init_ext_eeprom() {
   output_high(WP);
   output_float(PIN_C3);
   output_float(PIN_C4);
}

BOOLEAN ext_eeprom_ready() {
   int1 ack;
   output_high(WP);
   i2c_start();            // If the write command is acknowledged,
   ack = i2c_write(0xa0|((CHIP_SELECT&0x07)<<1));  // then the device is ready.
   i2c_stop();
   return !ack;
}

void write_ext_eeprom(long int address, BYTE data) {
   while(!ext_eeprom_ready());
   output_low(WP);
   i2c_start();
   i2c_write(0xa0|((CHIP_SELECT&0x07)<<1));
   i2c_write((BYTE)(address>>8));
   i2c_write((BYTE)address);
   i2c_write(data);
   i2c_stop();
   output_high(WP);
}

BYTE read_ext_eeprom(long int address) {
   BYTE data;

   output_high(WP);
   while(!ext_eeprom_ready());
   i2c_start();
   i2c_write(0xa0|((CHIP_SELECT&0x07)<<1));
   i2c_write((BYTE)(address>>8));
   i2c_write((BYTE)address);
   i2c_start();
   i2c_write((0xa0|((CHIP_SELECT&0x07)<<1))|1);
   data=i2c_read(0);
   i2c_stop();
   return(data);
}
```

74165.c

```
#IFNDEF EXP_IN_ENABLE

#define EXP_IN_ENABLE   PIN_A5//PIN_B3
#define EXP_IN_CLOCK    PIN_A3//PIN_B4
#define EXP_IN_DI       PIN_A4//PIN_B5
#define NUMBER_OF_74165 2

#ENDIF

void read_expanded_inputs(BYTE *ei) {
  BYTE i;

  output_high(EXP_IN_CLOCK);
  output_low(EXP_IN_ENABLE);      // Latch all inputs
  output_high(EXP_IN_ENABLE);

  for(i=1;i<=NUMBER_OF_74165*8;++i) {      // Clock in bits to the ei structure
    shift_left(ei,NUMBER_OF_74165,input(EXP_IN_DI));
    output_low(EXP_IN_CLOCK);
    output_high(EXP_IN_CLOCK);
  }
  output_low(EXP_IN_ENABLE);
}
```

74595.c:

```
#IFNDEF EXP_OUT_ENABLE

#define EXP_OUT_ENABLE  PIN_B2//PIN_B0
#define EXP_OUT_CLOCK   PIN_B3//PIN_B1
#define EXP_OUT_DO      PIN_B4//PIN_B2
#define NUMBER_OF_74595 2

#ENDIF

void write_expanded_outputs(BYTE* eo) {
  BYTE i;
  output_low(PIN_B5);
  output_high(PIN_B5);
  output_low(EXP_OUT_CLOCK);
  output_low(EXP_OUT_ENABLE);

  for(i=1;i<=NUMBER_OF_74595*8;++i) {  // Clock out bits from the eo array
    if((*(eo+(NUMBER_OF_74595-1))&0x80)==0)
      output_low(EXP_OUT_DO);
    else
      output_high(EXP_OUT_DO);
   shift_left(eo,NUMBER_OF_74595,0);
   output_high(EXP_OUT_CLOCK);
   output_low(EXP_OUT_CLOCK);
  }
  output_high(EXP_OUT_ENABLE);
  output_low(EXP_OUT_ENABLE);
}
```

Kbd_2.c:

```
#define COL0 PIN_C6
#define COL1 PIN_C5
#define COL2 PIN_D3

#define ROW0 PIN_D7
#define ROW1 PIN_D6
#define ROW2 PIN_D5
#define ROW3 PIN_D4

// Keypad layout:
char const KEYS[4][3] = {{'1','2','3'},
                         {'4','5','6'},
                         {'7','8','9'},
                         {'*','0','#'}};

#define KBD_DEBOUNCE_FACTOR 33    // Set this number to apx n/333 where
                                  // n is the number of times you expect
                                  // to call kbd_getc each second

void kbd_init() {
}

void kbd_input(){
  output_float(COL0);
  output_float(COL1);
  output_float(COL2);

  output_float(ROW0);
  output_float(ROW1);
  output_float(ROW2);
  output_float(ROW3);
}

char kbd_row(){
  char code;

  output_float(ROW0);
  output_float(ROW1);
  output_float(ROW2);
  output_float(ROW3);

  code=0;
  if (!input(ROW0)) code|=1;
  if (!input(ROW1)) code|=1<<1;
  if (!input(ROW2)) code|=1<<2;
  if (!input(ROW3)) code|=1<<3;

  return code;

}

char kbd_getc( ) {
   static BYTE kbd_call_count;
   static short int kbd_down;
   static char last_key;
   static BYTE col;

   BYTE kchar;
   BYTE row;
   BYTE kbd;

   kchar='\0';
   if(++kbd_call_count>KBD_DEBOUNCE_FACTOR) {
       kbd_input();
       switch (col) {
         case 0   :
                    output_drive(COL0);
                    output_low(COL0);
                    break;
         case 1   :
                    output_drive(COL1);
                    output_low(COL1);

                    break;
         case 2   :
                    output_drive(COL2);
                    output_low(COL2);
                    break;
       }

       delay_ms(10);

       kbd=kbd_row();
       if(kbd_down) {
         if(!kbd) {
           kbd_down=FALSE;
           kchar=last_key;
           last_key='\0';
         }
       } else {
          if(kbd) {
                 if(kbd & 1)      row=0;
             else if(kbd & (1<<1)) row=1;
             else if(kbd & (1<<2)) row=2;
             else if(kbd & (1<<3)) row=3;
             last_key =KEYS[row][col];
             kbd_down = TRUE;
          } else {
             ++col;
             if(col==3)
               col=0;
          }
       }
      kbd_call_count=0;
   }
  kbd_input();
  return(kchar);}
```

Lcd_2.c:

```
#define LCD_DATA3  PIN_D7
#define LCD_DATA2  PIN_D6
#define LCD_DATA1  PIN_D5
#define LCD_DATA0  PIN_D4

#define LCD_ENABLE PIN_D2
#define LCD_RS     PIN_D1//PIN_C5
#define LCD_RW     PIN_D0//PIN_C6

#define lcd_type 2           // 0=5x7, 1=5x10, 2=2 lines
#define lcd_line_two 0x40    // LCD RAM address for the second line

BYTE const LCD_INIT_STRING[4] = {0x20 | (lcd_type << 2), 0xc, 1, 6};
                             // These bytes need to be sent to the LCD
                             // to start it up.

void lcd_all_input(){
  output_float(LCD_DATA3);
  output_float(LCD_DATA2);
  output_float(LCD_DATA1);
  output_float(LCD_DATA0);

  output_low(LCD_ENABLE);
  output_float(LCD_RS);
  output_float(LCD_RW);
  delay_ms(1);
}

void lcd_input(){
  output_float(LCD_DATA3);
  output_float(LCD_DATA2);
  output_float(LCD_DATA1);
  output_float(LCD_DATA0);

  output_low(LCD_ENABLE);
  output_drive(LCD_RS);
  output_drive(LCD_RW);
  delay_ms(1);
}

void lcd_output(){
  output_drive(LCD_DATA3);
  output_drive(LCD_DATA2);
  output_drive(LCD_DATA1);
  output_drive(LCD_DATA0);

  output_low(LCD_ENABLE);
  output_drive(LCD_RS);
  output_drive(LCD_RW);
  delay_ms(1);
}

BYTE lcd_read_nibble(){
  BYTE data=0;

  if (input(LCD_DATA3)) data|=(1<<3);
  if (input(LCD_DATA2)) data|=(1<<2);
  if (input(LCD_DATA1)) data|=(1<<1);
  if (input(LCD_DATA0)) data|= 1;

  return data;
}

void lcd_write_nibble(BYTE data){
  output_bit(LCD_DATA3,(data>>3) & 1);
  output_bit(LCD_DATA2,(data>>2) & 1);
  output_bit(LCD_DATA1,(data>>1) & 1);
  output_bit(LCD_DATA0,(data   ) & 1);
}

BYTE lcd_read_byte() {
      BYTE low,high;

      lcd_input();
      output_high(LCD_RW);

      delay_cycles(1);

      output_high(LCD_ENABLE);
      delay_cycles(1);

      high = lcd_read_nibble();

      output_low(LCD_ENABLE);
      delay_cycles(1);

      output_high(LCD_ENABLE);
      delay_us(1);

      low = lcd_read_nibble();

      output_low(LCD_ENABLE);

      lcd_output();

      return( (high<<4) | low);
}

void lcd_send_nibble( BYTE n ) {
      lcd_write_nibble(n);
      delay_cycles(1);
      output_high(LCD_ENABLE);
      delay_us(2);
      output_low(LCD_ENABLE);
}

void lcd_send_byte( BYTE address, BYTE n ) {
```

```
      output_low(LCD_RS); delay_ms(1);
      while ( bit_test(lcd_read_byte(),7) ) ;
      output_bit(LCD_RS,address);
      output_low(LCD_RW);
      delay_ms(1);
      output_low(LCD_ENABLE);
      lcd_send_nibble(n >> 4);
      lcd_send_nibble(n & 0x0F);
}

void lcd_init() {
    BYTE i;

    lcd_output();
    output_low(LCD_RS);
    output_low(LCD_RW);
    output_low(LCD_ENABLE);

    delay_ms(15);
    for(i=1;i<=3;++i) {
       lcd_send_nibble(3);
       delay_ms(5);
    }
    lcd_send_nibble(2);
    for(i=0;i<=3;++i)
       lcd_send_byte(0,LCD_INIT_STRING[i]);
}

void lcd_gotoxy( BYTE x, BYTE y) {
   BYTE address;

   if(y!=1)
     address=lcd_line_two;
   else
     address=0;
   address+=x-1;
   lcd_send_byte(0,0x80|address);
}

void lcd_putc( char c) {
   switch (c) {
     case '\f'   : lcd_send_byte(0,1);
                   delay_ms(2);
                                         break;
     case '\n'   : lcd_gotoxy(1,2);        break;
     case '\b'   : lcd_send_byte(0,0x10);  break;
     default     : lcd_send_byte(1,c);     break;
   }
}

char lcd_getc( BYTE x, BYTE y) {
   char value;

    lcd_gotoxy(x,y);
    while ( bit_test(lcd_read_byte(),7) ); // wait until busy flag is low
    output_high(LCD_RS);
    delay_ms(1);
    value = lcd_read_byte();
    output_low(LCD_RS);
    delay_ms(1);
    return(value);
}
```

RTC_DS1307.c:

```
#define Configuracion_DS1307 0x83 // 32.768 KHz de salida
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
char data[7];            //Variable Global
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
int BCDaBIN(int bcd){    //Conversión de BCD a Binario
  return(((bcd >> 4)&0x0F)*10 + (bcd & 0x0F));
}
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
int BINaBCD(int bin){  //Conversión de Binario a BCD
  unsigned char Hi,Lo;

  Hi=bin/10;
  Lo=bin%10;
  return (Hi<<4)|Lo;
}
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
void escribir_ds1307(void){
char i;

  for(i = 0; i < 7; i++)           //Cargo la data de entrada
     {data[i] = BINaBCD(data[i]);}

  i2c_start();                     //Escritura
  i2c_write(0xD0);                 //Código de escritura
  i2c_write(0x00);                 //Puntero a la primera dirección

  for(i = 0; i < 7; i++)           //Escritura de data en el DS1307
     {i2c_write(data[i]);}

  i2c_write(Configuracion_DS1307); //Configuro el DS1307
  i2c_stop();
}
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
void leer_ds1307(byte &hor, byte &min, byte &seg, byte &day, byte &dia, byte &mes, byte &a_o){

  i2c_start();                     //Escritura
  i2c_write(0xD0);                 //Código de escritura
  i2c_write(0x00);                 //Puntero a la primera dirección
  i2c_start();                     //Lectura
  i2c_write(0xD1);                 //Código de lectura

  seg = BCDaBIN(i2c_read(1)&0x7f);  //Lectura de los 7 bit de los segundos
  min = BCDaBIN(i2c_read(1)&0x7f);  //Lectura de los 7 bit de los minutos
  hor = BCDaBIN(i2c_read(1)&0x3f);  //Lectura de los 6 bit de las horas
  day = BCDaBIN(i2c_read(1)&0x07);  //Lectura de los 6 bit de la fecha
  dia = BCDaBIN(i2c_read(1)&0x3f);  //Lectura de los 6 bit del dia
  mes = BCDaBIN(i2c_read(1)&0x1f);  //Lectura de los 6 bit del mes
  a_o = BCDaBIN(i2c_read(0));       //Lectura de los 8 bit del año

  i2c_stop();
}
```

[ANEXO Nº 2]

[Diagrama de Pantallas de Las Tareas del Sistema]

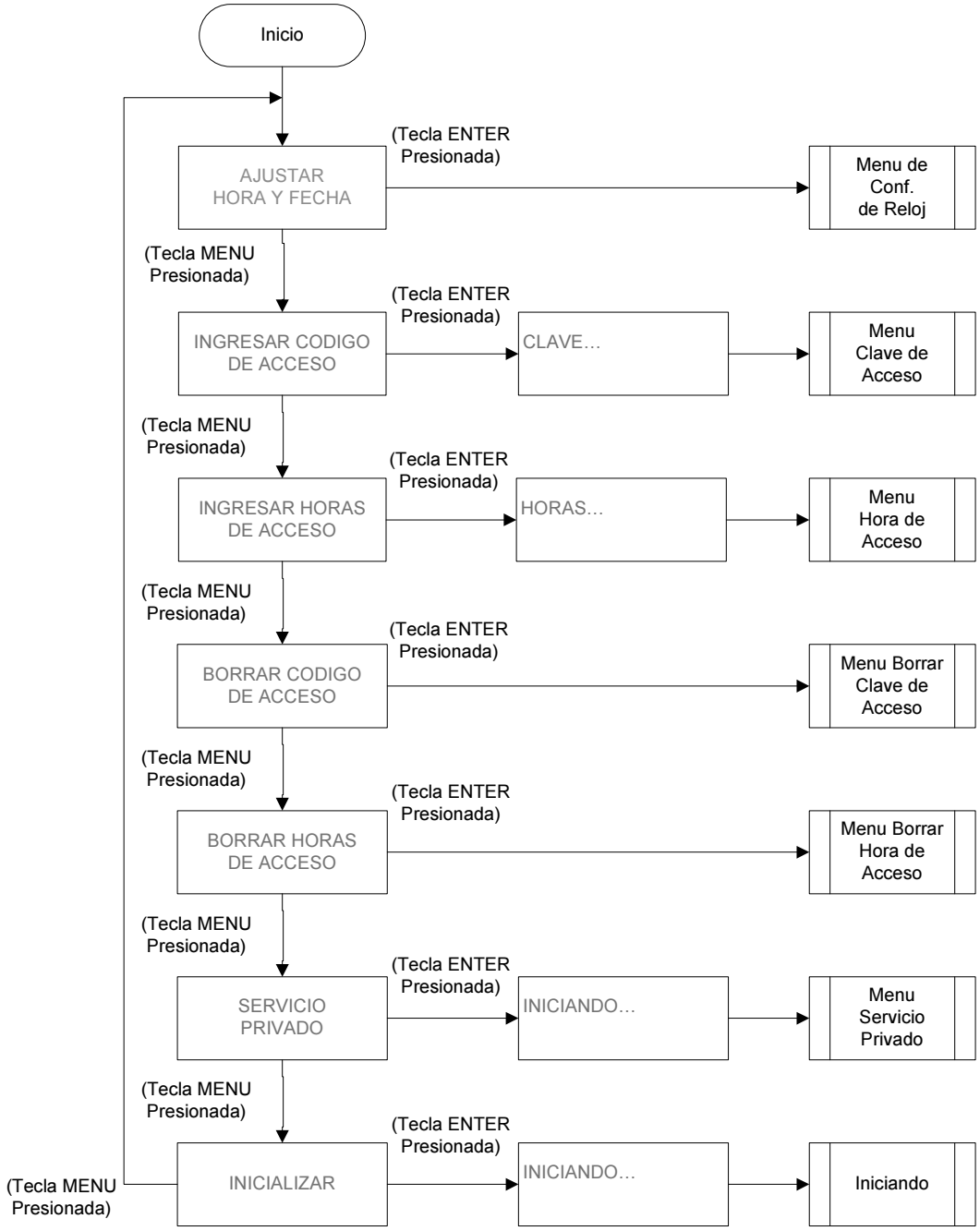

Figura 1: Diagrama de Pantallas del Menú Principal

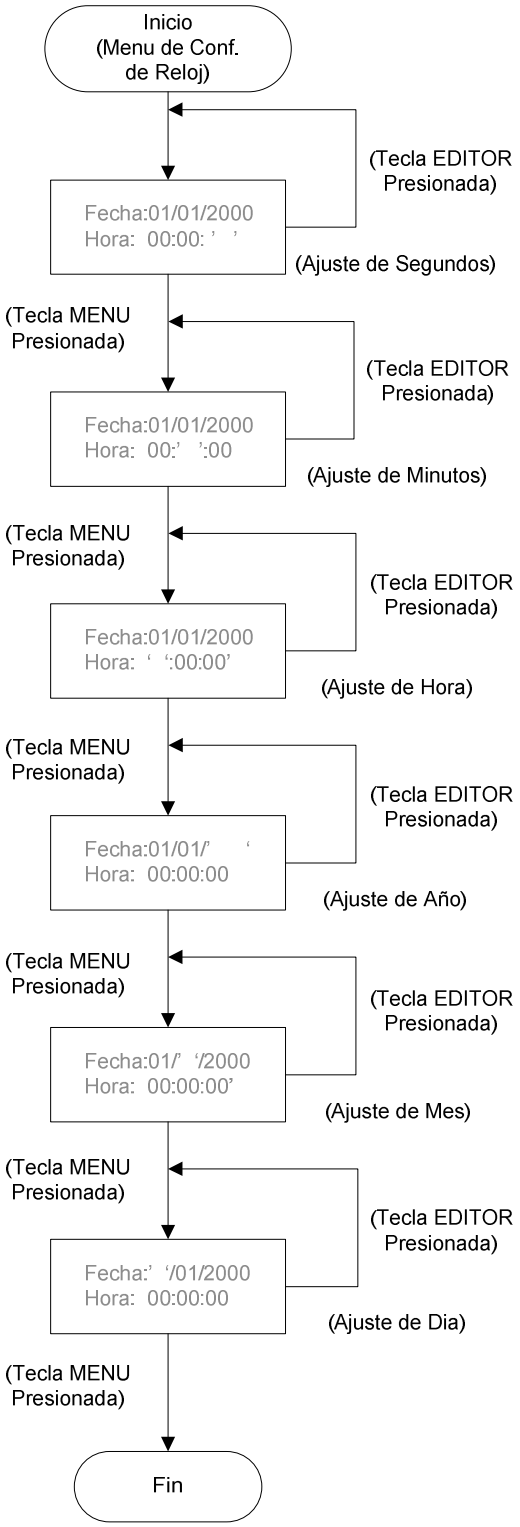

Figura 2: Diagrama de Pantallas del Menú de Ajuste del RTC Externo

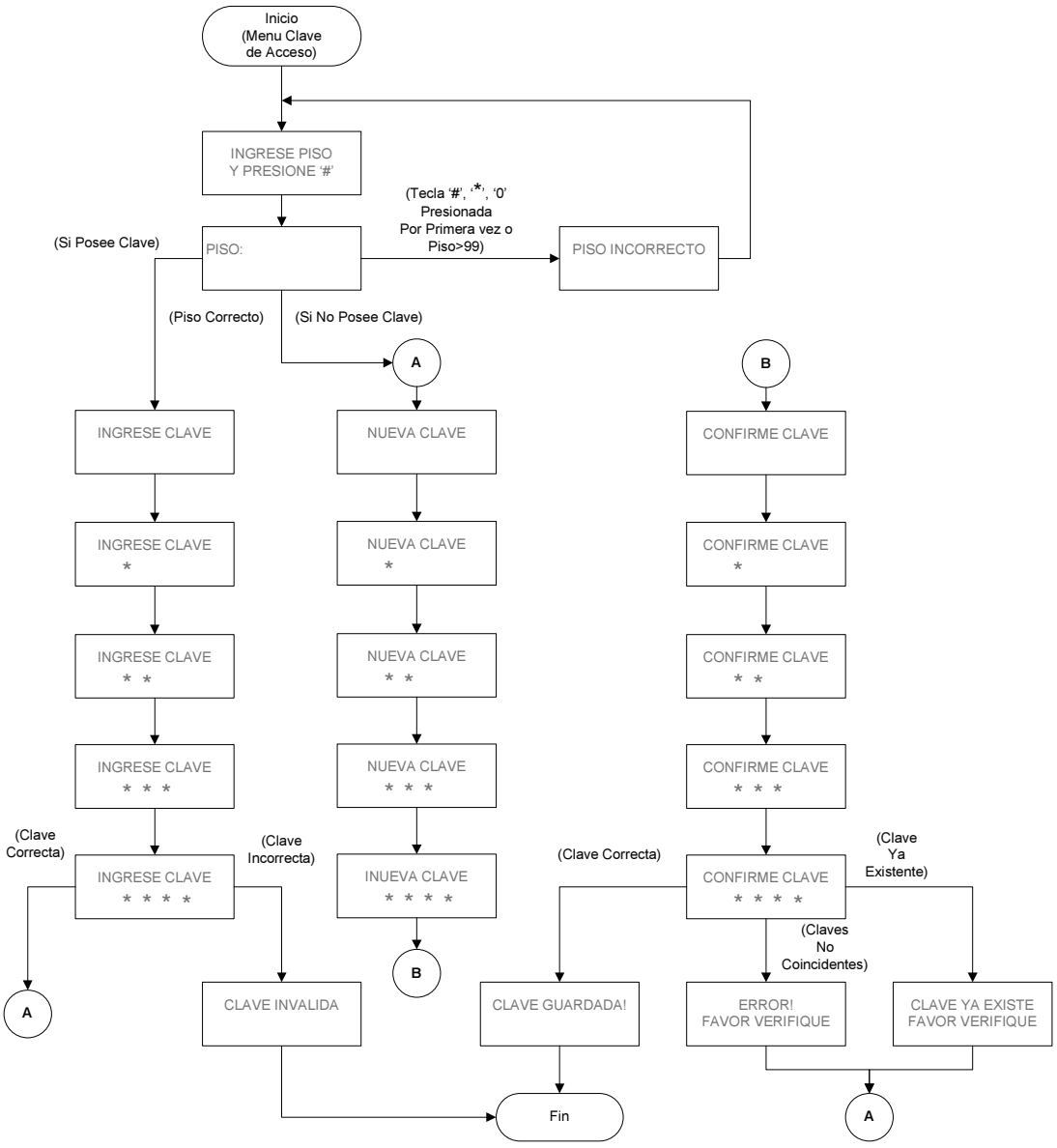

Figura 3: Diagrama de Pantallas del Menú de Ingreso de Clave de Acceso

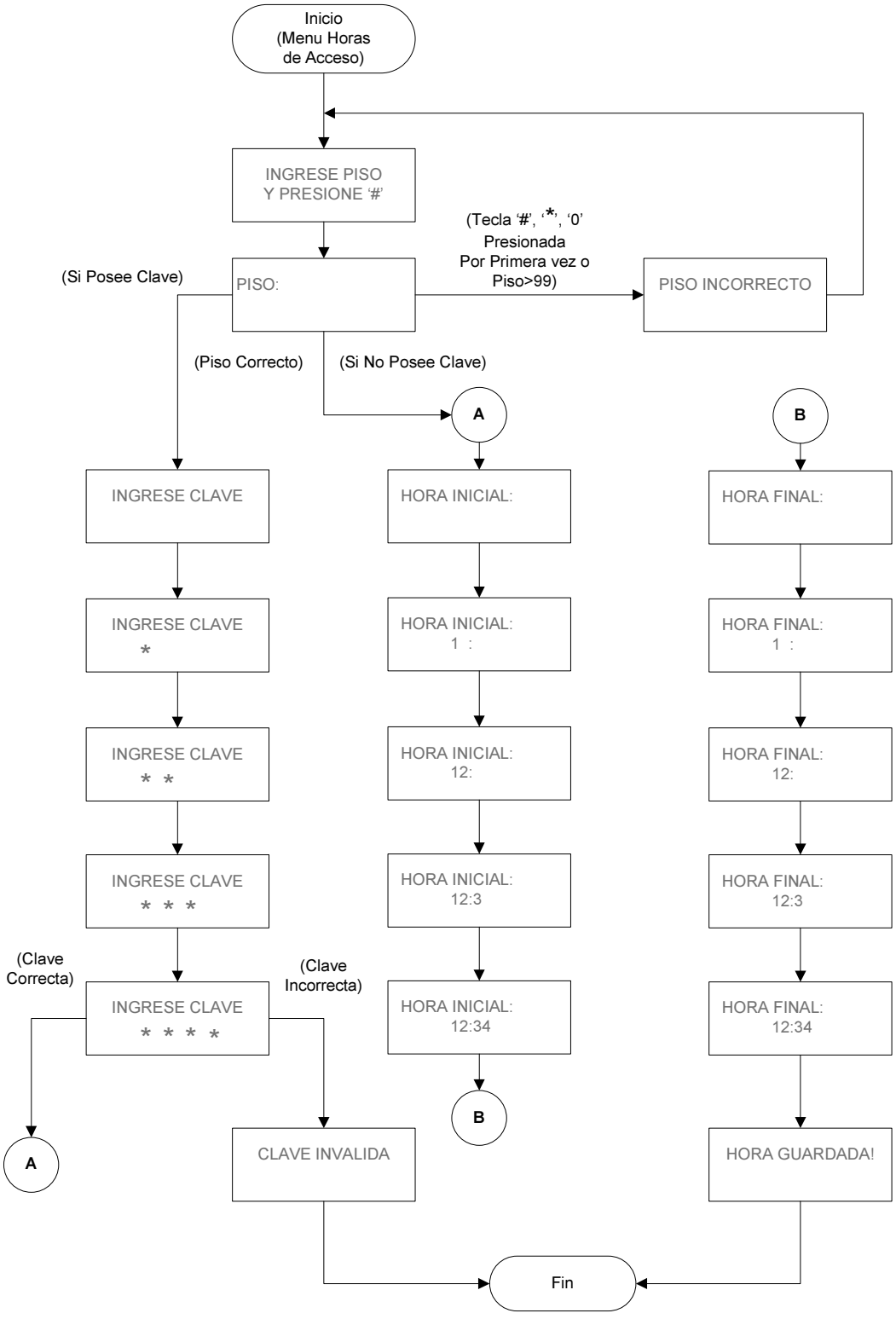

Figura 4: Diagrama de Pantallas del Menú de Ingreso de Horas de Acceso

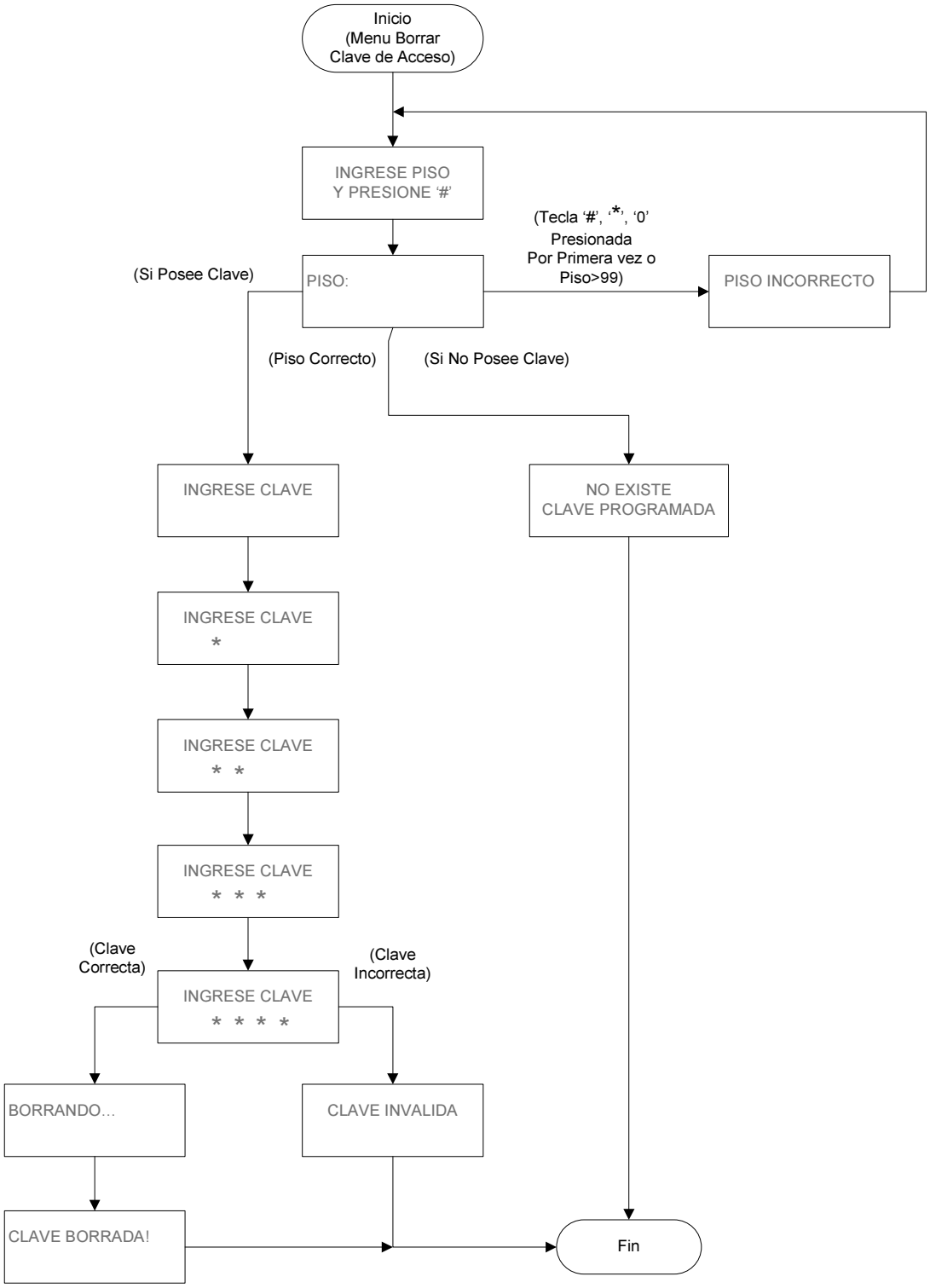

Figura 5: Diagrama de Pantallas del Menú de Borrado de Clave de Acceso

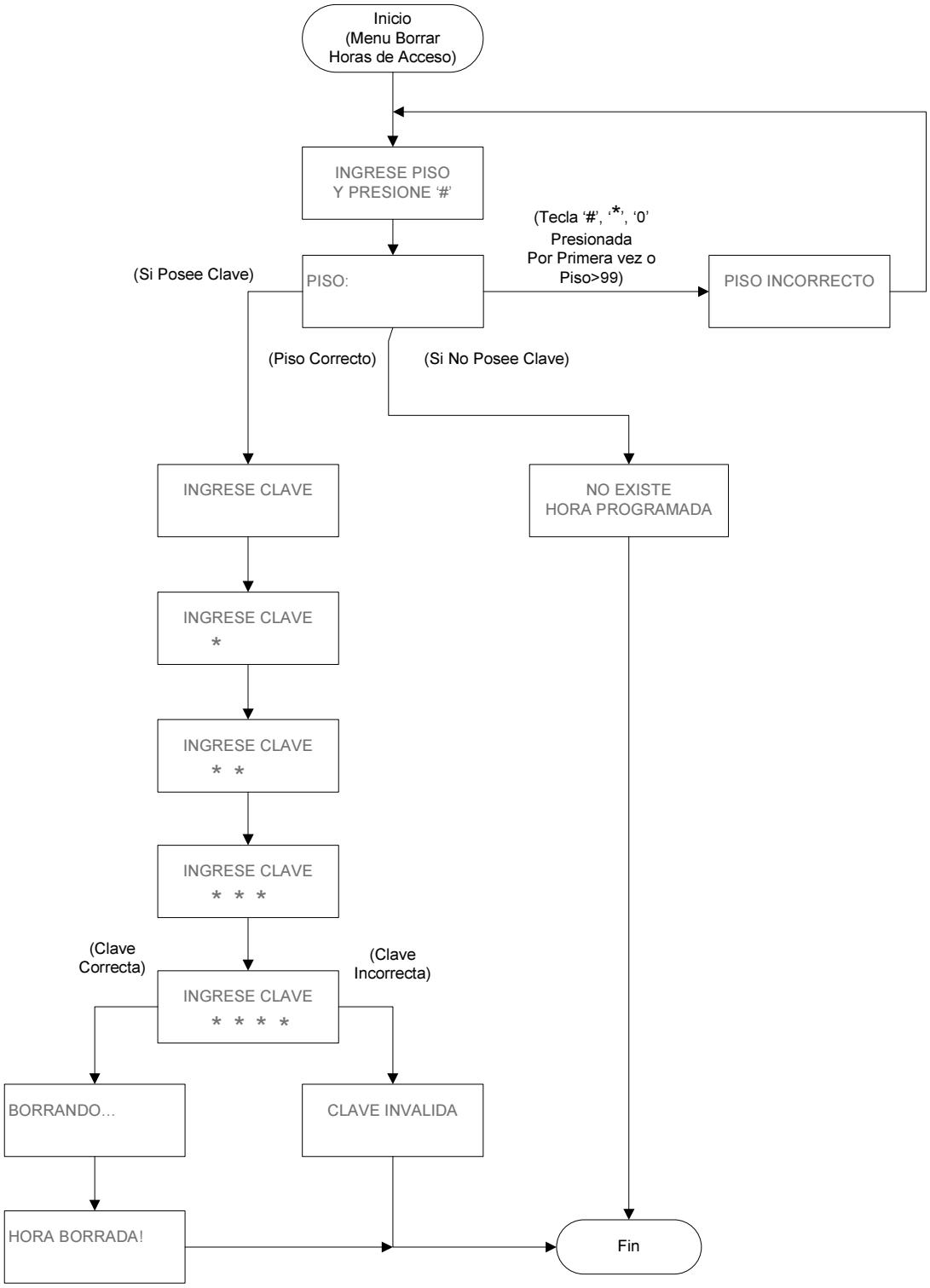

Figura 6: Diagrama de Pantallas del Menú de Borrado de Horas de Acceso

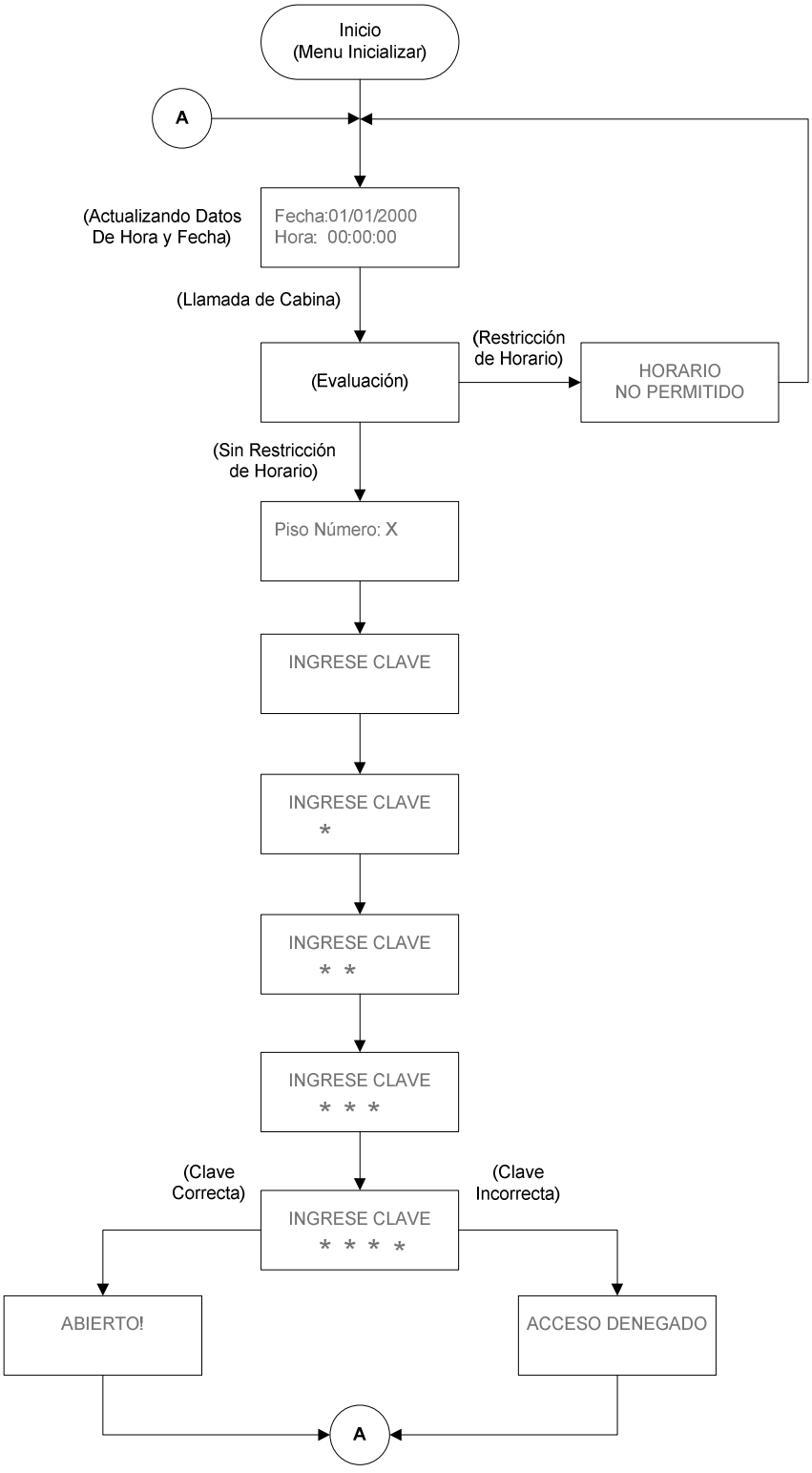

Figura 7: Diagrama de Pantallas del Menú Inicializar

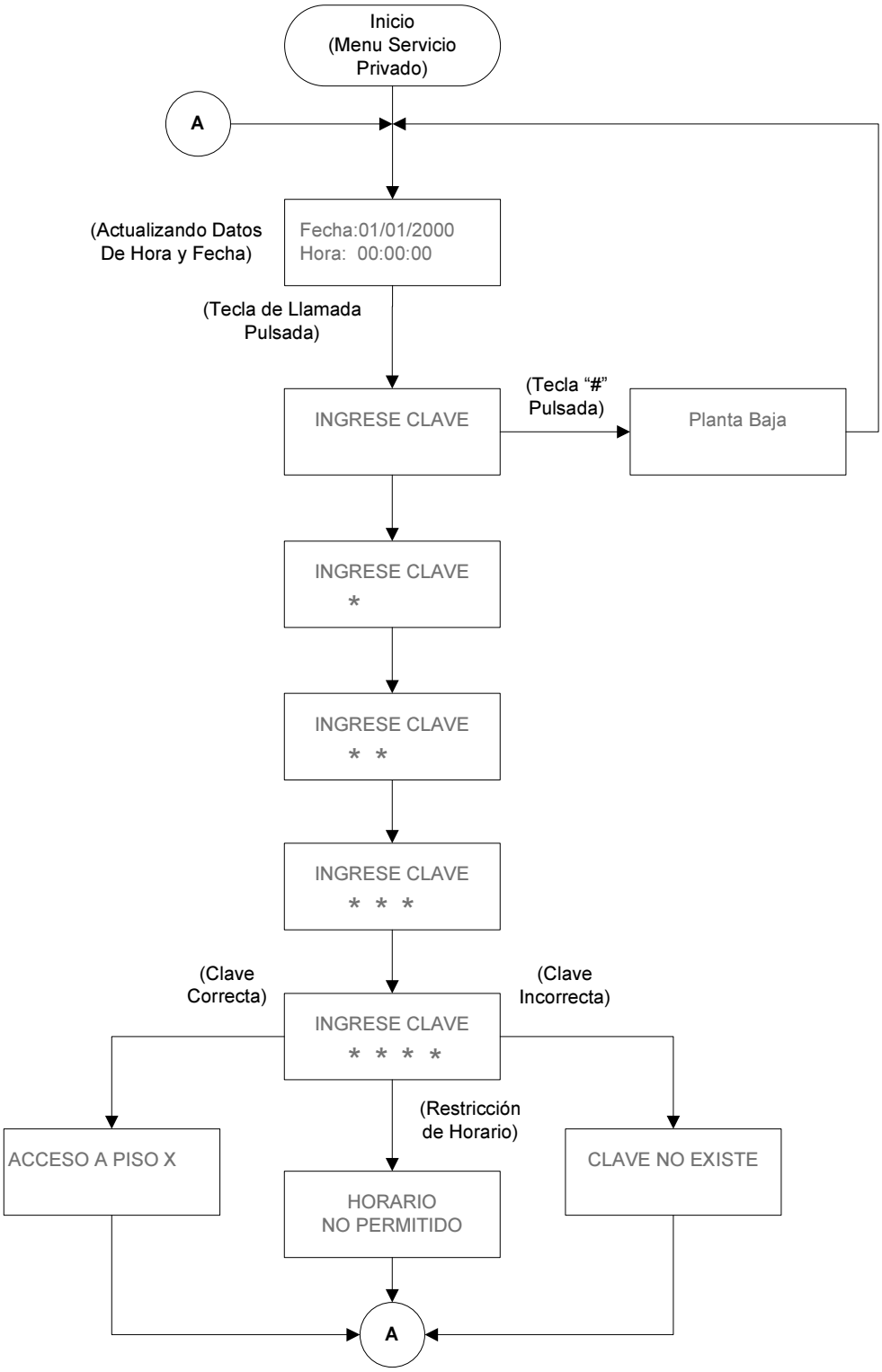

Figura 8: Diagrama de Pantallas del Menú Servicio Privado

[ANEXO Nº 3]

[Diagrama Esquemático, PCB e Implementación de La Interfaz Diseñada]

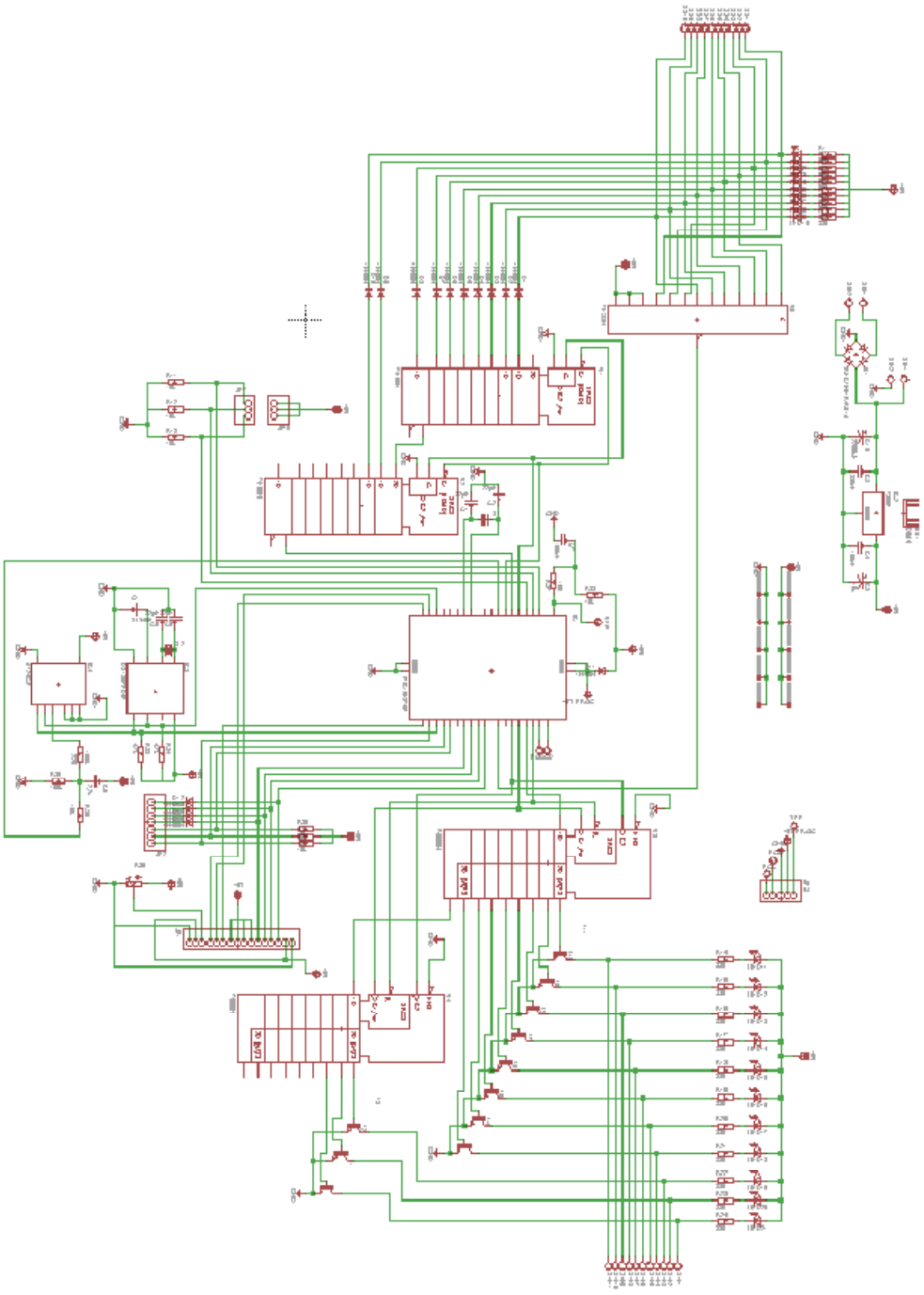

Figura 1: Diagrama Esquemático de La Interfaz Diseñada

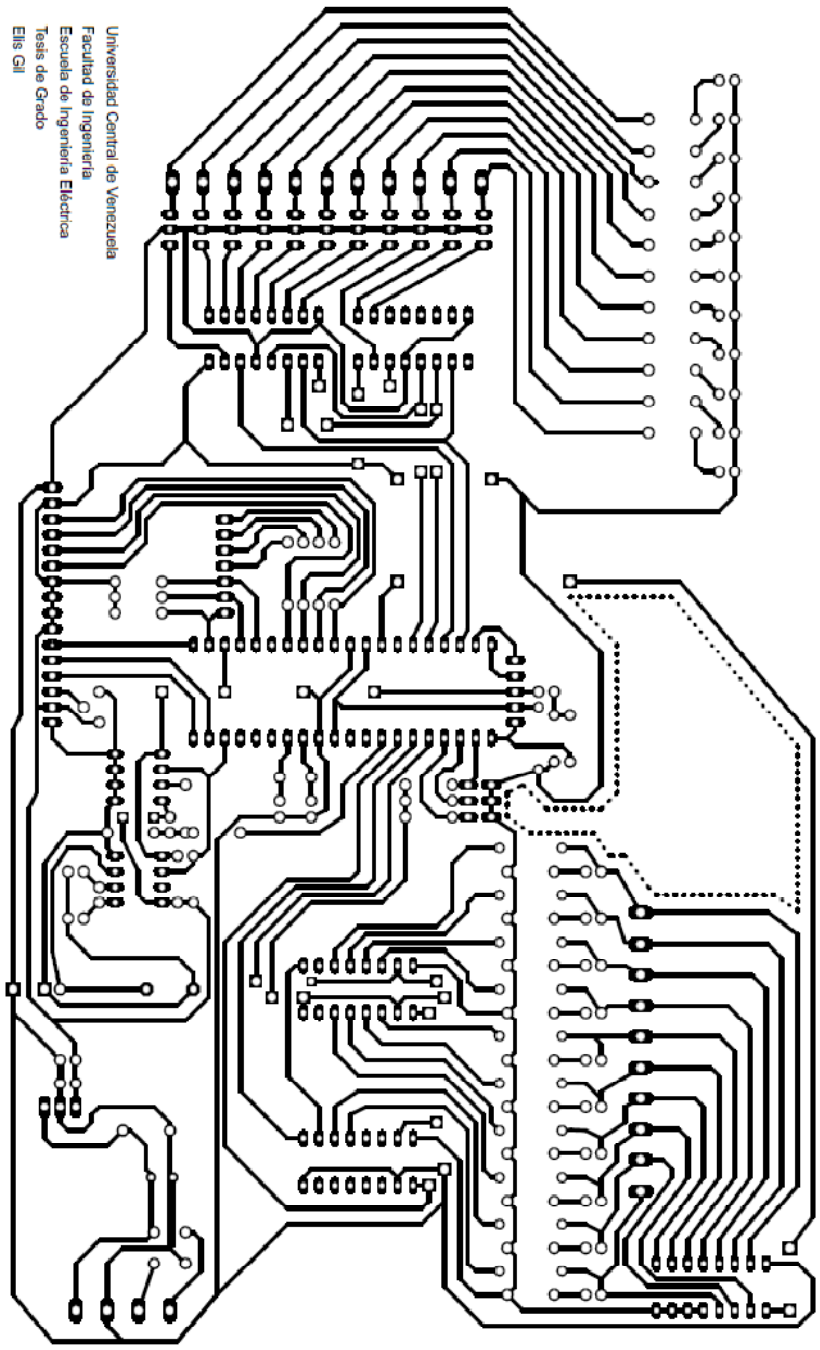

Figura 2: PCB de La Interfaz Diseñada.

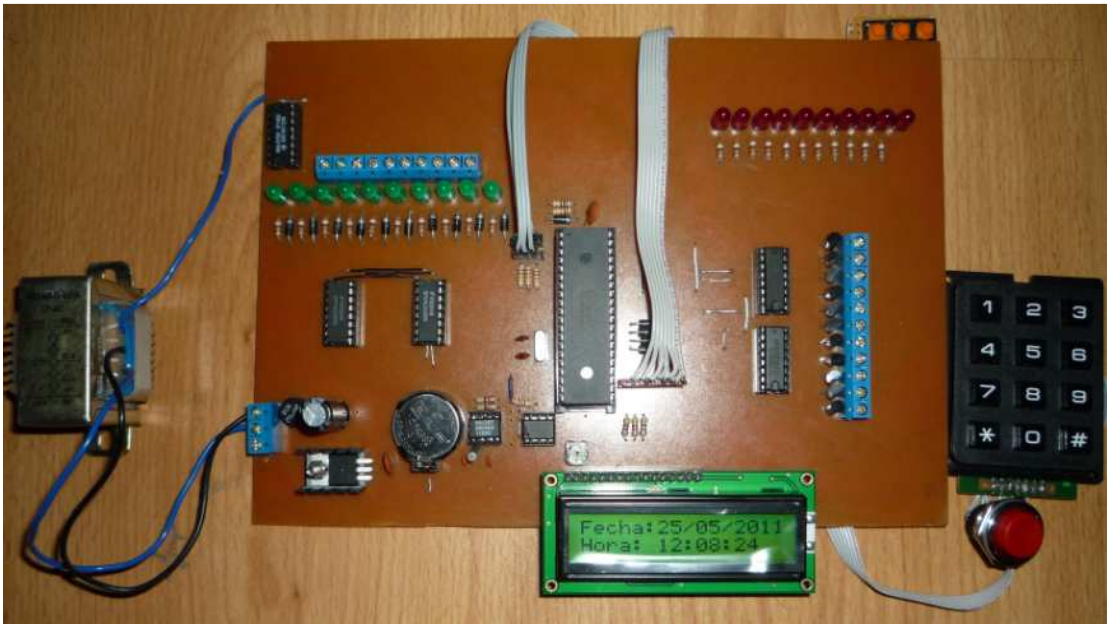

Figura 3: Implementación Final del Sistema Diseñado.