Universidad
Carlos III de Madrid
www.uc3m.es

Trabajo Fin de Grado:

SISTEMA DE CONTROL DE ACCESOS BASADO EN MICROPROCESADORES ARM32 CORTEX

GRADO EN INGENIERÍA ELECTRÓNICA Y AUTOMÁTICA

AUTOR: *Mario Acevedo Aguilar*

TUTOR Y DIRECTOR: *Raúl Sánchez Reíllo*

Leganés, 1 de Septiembre de 2012

Gracias a mi familia y a mi novia
por soportarme y apoyarme cuando más
lo necesitaba. Os quiero.

RESUMEN

La constante evolución de la microelectrónica ha conseguido alcanzar un nivel que hace años parecía imposible. No sólo ha conseguido reducir el tamaño de los chips sino también aumentar exponencialmente su capacidad. Gracias a estos continuos avances hoy en día somos capaces de desarrollar sistemas como el que se desarrolla en este proyecto, un sistema de control de accesos basado en microprocesadores. Este tipo de sistemas que antes necesitaban una gran cantidad de dispositivos, ahora se pueden diseñar a partir de unas pocas conexiones focalizadas en un minúsculo chip. Nuestro sistema de control de accesos se desarrollará en la plataforma de desarrollo STM3210C-EVAL y se centrará en permitir o denegar el acceso a usuarios, identificándolos por una tarjeta inteligente o por una contraseña introducida por pantalla táctil y verificando sus permisos.

ABSTRACT

The continuous evolution of microelectronics has reached a level that years ago seemed to be impossible. It has not only reduced the semiconductor size, but also has increased their capacity exponentially. Thanks to these ongoing developments, nowadays we are able to develop systems like the one developed in this project, an access control system based on a single microprocessor. This type of systems, which used to need many devices, today can be designed with few connections focused on a tiny chip. Our access control system will be developed in the STM3210C-EVAL evaluation board allowing or denying access to users, identifying them by a smartcard or a password introduced using a touch screen, checking the user permissions to enter the facility controlled by the device.

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE FOTOGRAFÍAS

ÍNDICE DE ACRÓNIMOS

ARM	Advanced RISC Machine
RISC	Reduced Instruction Set Computer
NVIC	Nested Vectored Interrupt Controller
RTC	Real-Time Clock
EXTI	External Interrupt Controller
RCC	Reset and Clock Control
DMA	Direct Memory Access
SRAM	Static Random Access Memory
AHB	Advanced High-performance Bus
APB	Advanced Peripheral Bus
HSE	High Speed External
HSI	High Speed Internal
LSE	Low Speed External
LSI	Low Speed Internal
PLLCLK	Microprocessor Clock
SYSCLK	System Clock
ISR	Interrupt Service Routine
WKUP	WakeUp
NRST	External Reset
IWDG	Independent WatchDog
GPIO	General Purpose Input/Output
USART	Universal Synchronous/Asynchronous Receiver Transmitter
AFIO	Alternate Function Input/Output
ATR	Answer To Reset
SW1 y SW2	Bytes de Estado

1 INTRODUCCIÓN

El presente Proyecto Fin de Carrera titulado "Sistema de control de accesos basado en microprocesadores" tiene como objetivo la elaboración de un sistema autónomo que controle todos los permisos para acceder a cualquier zona de una empresa o institución y archivar las fechas y horas exactas de cuándo se hace.

La motivación que mueve al autor a realizar este proyecto es tan simple como la motivación que mueve a un ingeniero por aprender e intentar dominar cosas que le gustan.

Desde un principio, se sintió atraído por los microprocesadores, y no sólo por su gran capacidad de procesamiento de datos y comunicación con el exterior, sino también por la extraordinaria variedad de posibilidades que ofrece saber manejar un sistema como el que se utiliza a continuación. Esa gran variedad que permite manejar desde lo más básico que puede ser un electrodoméstico pequeño, hasta toda la domótica de una casa.

La posibilidad que brinda la Universidad de conocer este tipo de tecnología con un Trabajo Fin de Grado, no es una posibilidad como tal, sino más bien un privilegio al que todo el mundo debería acceder.

Los objetivos de este proyecto no son muy distintos a los requisitos mínimos que se le piden a este tipo de servicio, los sistemas de control de accesos. Dichos objetivos se resumen brevemente a continuación:

- ✓ Controlar de forma exhaustiva los horarios de entrada y salida de los empleados y/o clientes.
- ✓ Permitir/no permitir a los empleados y/o clientes acceso a las diferentes partes de la empresa o institución.
- ✓ Capacidad para permanecer como sistema de bajo consumo en periodos largos de inactividad.

Aunque estos son los servicios mínimos que en un principio deberá cumplir nuestro sistema, se incluirán más funciones que colaboren positivamente.

Para ello se dispondrá del sistema de desarrollo *STM3210C Eval* basado en el microcontrolador STM21F107VCT. Este sistema dispone de los medios suficientes para, con la programación adecuada, cumplir los objetivos que se han descrito.

El principal método de identificación en este sistema serán las tarjetas inteligentes, las cuales contienen un microprocesador integrado, de ahí el título del presente proyecto: "*Sistema de Control de Accesos basado en Microprocesadores*".

Es importante destacar que la elaboración de este proyecto no es conseguir un sistema comercial que sirva como control de accesos totalmente definido y de forma inmediata, más bien se trata de ejemplarizar las numerosas posibilidades que presenta el uso de microprocesadores en este tipo de sistemas.

También es importante dejar claro que el objetivo de este proyecto se centra en la construcción de un prototipo de dispositivo de control de acceso, dejando para futuros trabajos, aquellas herramientas de gestión que habría que implantar para labores como la explotación de los datos de acceso, la emisión y anulación de tarjetas inteligentes, la gestión de permisos, etc.

La **redacción** de este proyecto comenzará con unos pequeños fundamentos teóricos que se necesitan para comprender correctamente el funcionamiento del sistema. Esta breve descripción se realizará en el siguiente capítulo 2 "*ESTADO DE LA TÉCNICA".*

Posteriormente se iniciará el capítulo 3 "*PLATAFORMA DE DESARROLLO*" donde se describirá de forma generalizada el sistema de desarrollo STM3210C-EVAL, centrándose en aquellos aspectos de mayor importancia.

En el siguiente capítulo nos dedicaremos a explicar en detalle cuáles son los requisitos de este tipo de sistemas, así como la metodología de trabajo que se ha seguido, capítulo 4 "*METODOLOGÍA Y REQUISITOS".*

Seguidamente se explicará pormenorizadamente cómo ha sido el diseño de nuestro sistema (capítulo 5 "*DISEÑO DEL SISTEMA"),* cómo se ha desarrollado (capítulo *6*

"DESARROLLO DEL SOFTWARE") y cuáles han sido las pruebas realizadas para lograr el resultado final", que se verán en el capítulo 7 "*PRUEBAS Y RESULTADOS*".

En el último capítulo de este proyecto, capítulo 8 "*CONCULSIONES Y LÍNEAS FUTURAS",* hablaremos sobre la consecución o no de los objetivos y qué deducimos de todo lo anterior. Además de comentar posibles ampliaciones que ayudarían a este prototipo a ser más competente.

Por último, mencionar que todo capítulo empezará con una breve descripción del mismo, indicando qué aspectos trata y cómo se organizará.

2 ESTADO DE LA TÉCNICA

Dentro de este capítulo nos centraremos en explicar de forma breve y concisa los fundamentos teóricos necesarios para entender el funcionamiento del sistema.

El proyecto, como ya se ha mencionado, es un sistema de control de accesos basado en microprocesadores. Por lo tanto nuestro primer objetivo será explicar qué entendemos por "Sistema de control de accesos", cuál es su objetivo y qué funciones debe cumplir cualquiera sea su propósito.

Seguidamente hablaremos sobre la tecnología de microprocesadores enfocándola lo más posible a su utilización como sistema de control de accesos. Explicaremos cuál es su situación actual y qué innovaciones introduce esta tecnología, si es que introduce alguna.

Y por último, nos centraremos en un método de control de accesos, la tarjeta inteligente. Ésta es una tecnología que proporciona un gran número de posibilidades y que utilizaremos como uno de los modos para controlar el acceso. Aunque los modos de control de accesos no se explican en esta parte del proyecto, es necesario e imprescindible explicar aquellos términos de tarjetas inteligentes, así como sus condiciones de uso, para entender el funcionamiento de las mismas.

2.1 Sistema de Control de Accesos

Un sistema de control de accesos se define como un método para gestionar y administrar el flujo de personas, vehículos o materiales a través de las entradas y salidas de un edificio o área protegida [1].

Para controlar y gestionar dichos movimientos se utilizan dispositivos electrónicos o electromecánicos que permiten controlar los privilegios de cada usuario y cada puerta

con sencillez. Aunque en nuestro caso se utilizará un microcontrolador para gestionar dicho sistema, en la realidad existen numerosas formas y métodos para controlar el acceso a las áreas requeridas.

La idea básica es identificar mediante algún parámetro al usuario, transmitirle dicha información a la unidad de control para su evaluación y una vez evaluada, permitir o denegar el acceso.

Los parámetros de identificación constituyen la clave del método que se va a utilizar para el sistema de control de accesos. Según se quiera identificar al usuario de una manera u otra, el sistema se deberá diseñar con una tecnología u otra. Las formas de identificación más características y conocidas son:

- ✓ Número/Contraseña introducido por un teclado
- ✓ Lectura de información codificada en tarjeta (por ejemplo, tarjetas de proximidad, de banda magnética, tarjetas inteligentes, etc.)
- ✓ Verificación por atributo biométrico (por ejemplo, reconocimiento por voz, por patrón del iris, por huella dactilar, etc.)
- ✓ Combinación de dos o más de los anteriores

En nuestro caso utilizaremos las dos primeras: contraseña por teclado y tarjetas inteligentes. La identificación predeterminada será con tarjetas inteligentes, donde se entregará una para cada usuario, programada según sean sus privilegios de acceso. El segundo método es por la posibilidad de que en un determinado momento la tarjeta inteligente falle, o el usuario no disponga de ella.

Una vez que hemos definido la forma de identificar a las personas/objetos, dicha información se transmite a una unidad de control para evaluar los datos. Lo más común es utilizar un dispositivo que simplemente verifique la entrada de datos, y dé permiso al usuario.

2.2 Microcontroladores en sistemas de control de accesos

En la actualidad, el diseño de sistemas, sea cual sea su objetivo, se está enfocando en la programación de dispositivos programables, tales como los microcontroladores. La razón radica en que este tipo de dispositivos permiten reducir notablemente la cantidad de componentes electrónicos necesarios para un determinado sistema [2].

Los microcontroladores se definen como circuitos integrados que incorporan un microprocesador con una serie de módulos de memoria y una gran variedad de periféricos. De esta manera el mismo chip es capaz de gestionar y manipular todos los periféricos y memorias, reduciendo el espacio necesario en placa de circuito impreso, los posibles errores debidos a conexiones y soldaduras, y minimizando las pérdidas de rendimiento por retardos en las interconexiones.

Además, esta gran ventaja de manejar los periféricos desde un mismo chip, se traduce en otras ventajas como: mayor facilidad y rapidez en las comunicaciones, más posibilidades de actuación y procesamiento de información con el exterior y, más versatilidad y variedad de funciones.

Para nuestro caso particular, disponer de un microcontrolador para gestionar el control de accesos permite aprovecharse de las ventajas mencionadas anteriormente. Pero además, al disponer de placas de desarrollo diseñadas para ese microcontrolador, tales como la STM3210C-EVAL, permite poder centrarse en el desarrollo de la programación, sin tener que preocuparse del interconexionado de los distintos componentes y conectores que se necesitan como acompañamiento al microcontrolador.

La placa STM3210C-EVAL dispone, entre otros elementos, de un lector de tarjetas inteligentes, una pantalla táctil de tipo resistivo y un puerto para la comunicación serie. Por lo tanto, no será necesario incluir nuevos periféricos y bastará con aprender a utilizar los que ya están incluidos.

2.3 Tarjetas Inteligentes

Como ya habíamos comentado, nuestro principal sistema de identificación del usuario será una tarjeta inteligente personalizada. Es imprescindible saber cómo funciona este tipo de tecnología para entender el desarrollo del diseño.

Inicialmente nos vamos a ocupar de definir qué es una tarjeta inteligente y cuál es su aspecto. La tarjeta inteligente se define como una tarjeta de plástico de dimensiones normalizadas basada en un microprocesador y una memoria (en su conjunto, microcontrolador) [3].

En la Figura 1, se observa un diagrama de bloques explicativo sobre las tarjetas inteligentes. En él se observa que dispone de todas las celdas características de un ordenador a excepción de los periféricos que se comunican con el exterior (ratón, teclado). Dispone también de un Sistema Operativo (SOTI) que permite procesar información a altas velocidades y guardar en memoria sus resultados.

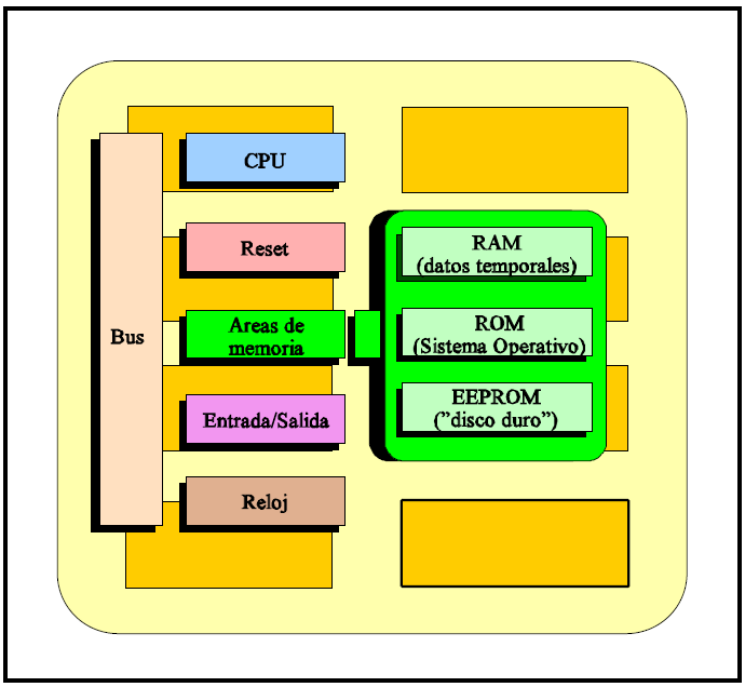


Figura 1: Diagrama de Bloques en una tarjeta inteligente (tomado de [3]).

Gracias a esta arquitectura, la tarjeta inteligente presume de grandes ventajas de fiabilidad, capacidad y sobre todo seguridad. Sin embargo, el precio por tarjeta es relativamente elevado si lo comparamos con otras tarjetas de identificación, como las de banda magnética. Es por esto, que la empresa o institución que vaya a utilizar esta solución debería realizar un estudio comparativo entre precio y seguridad para ver qué opción del mercado le compensa más.

A continuación vamos a definir muy brevemente los tipos de tarjetas que hay, ya que algunas de las características que se explican posteriormente dependen del tipo de tarjeta.

Básicamente existen dos tipos de tarjetas inteligentes: las tarjetas con contactos y las tarjetas sin contactos. En nuestro caso será con contactos donde la comunicación se hace mediante la conexión física de los contactos metálicos entre la tarjeta y la unidad lectora. En el caso de las tarjetas sin contactos, la comunicación se realiza de forma aérea mediante radiofrecuencia.

Cabe destacar que, dependiendo de la tensión de alimentación, se definen dos tipos de tarjetas con contactos: tipo A y tipo B [3].

- ✓ Las tarjetas tipo A alimentadas por 5V con un consumo máximo de de corriente de 60mA con picos máximos de 100mA durante 400ns[1].
- ✓ Las tarjetas tipo B alimentadas por 3V con un consumo máximo de corriente de 50mA con picos máximos de 50mA durante 400ns.

Otro aspecto importante respecto a las tarjetas inteligentes es la existencia de un sistema operativo. El SOTI (Sistema Operativo de la Tarjeta Inteligente) del que dispone la tarjeta, proporciona un interfaz de comandos que facilitan el uso de las tarjetas. Sin embargo, no se queda ahí, ya que contiene una serie de restricciones de comunicación que aseguran su funcionamiento y evitan una posible falsificación o un mal uso.

Dichas restricciones se resumen como un protocolo de comunicación que debe seguirse siempre que se quiera transmitir/recibir información de la tarjeta. Éste se puede resumir en lo siguiente:

- ✓ Impide la ejecución de cualquier tipo de instrucción hasta que no se haya realizado un reset.
- ✓ Al recibir un reset, inicializa los parámetros de la tarjeta, da una respuesta denominada ATR (Answer to Reset) y espera la recepción de una instrucción. Dicho ATR es una serie de bytes que cumplen la relación impuesta por la

[1] *Nota: La tarjeta utilizada en este proyecto será de este tipo.*

norma ISO/IEC 7816-3 [4]. Esta cadena de bytes aseguran que la tarjeta está funcionando de forma correcta, ya que si se detecta un error en esa trama, significa que la tarjeta está mal configurada y debe reconfigurarse o repararse. Además sirve para indicarle al terminal los límites del protocolo de comunicación que acepta/exige la tarjeta (por ejemplo, velocidad máxima de transmisión)

- ✓ Recepción de la instrucción. Cuando recibe la instrucción, el SOTI hace una serie de comprobaciones como: comprobar que la instrucción sea válida, verificar las condiciones de seguridad, verificar parámetros, etc. Si todas las comprobaciones son positivas, lleva a cabo la instrucción. Si por el contrario alguna ha dado negativa, declara un error al exterior.

En otras palabras, el SOTI no permite en ningún momento la ejecución de ninguna instrucción de bajo nivel, ni puede contener ninguna instrucción que pueda interferir en la seguridad del sistema o de la tarjeta.

Para finalizar con esta introducción a las tarjetas inteligentes es necesario profundizar ligeramente en la trama del ATR, el formato del par comando/respuesta que utiliza la tarjeta y en las instrucciones de comunicación entre ULE y tarjeta.

El ATR es el primer intercambio de información que se realiza en cualquier comunicación con tarjetas inteligentes. Como ya se ha comentado, en él vienen definidos los parámetros necesarios para que la comunicación se lleve a cabo sin errores. Dichos parámetros pueden ser, entre muchos otros: frecuencia de transmisión y tipo de protocolo de comunicación. Antes de la comunicación las dos partes implicadas mantienen un nivel eléctrico Z permitiendo detectar cambios y saber cuando comienza el ATR.

La estructura del ATR se define como una serie de bytes que dan información de la tarjeta:

- ✓ Carácter Inicial TS
- ✓ Carácter de Formato TO
- ✓ Carácter de Interfaz TAi, TBi, TCi, TDi

- ✓ Carácter Históricos T1, T2…. TK
- ✓ Carácter de Control TCK

Sin embargo, para los objetivos de este trabajo, no es necesario profundizar en la estructura interna del ATR, ya que el intercambio de información con la tarjeta no es masivo, y por lo tanto no merece la pena el esfuerzo de optimización del rendimiento mediante el análisis exhaustivo del ATR. Simplemente basta con explicar que el TS indica el convenio lógico de niveles y la velocidad de transmisión y que el TO indica si existen o no los caracteres de interfaz y el número de caracteres históricos. Los caracteres de interfaz, los cuales son opcionales, nos indican qué tipo de protocolo se va a usar en la comunicación (orientado a carácter, orientado a bloque, etc.). Por otro lado, los caracteres históricos especifican diversos aspectos de las tarjetas como: circuito integrado en la tarjeta, tamaño de memoria, versión de máscara, etc. Y por último el TCK, cuyo valor es el or-exclusivo de todos los caracteres del ATR que van desde el TO hasta el anterior al TCK.

El sistema operativo implementa un entorno multiaplicación con una estructura jerárquica de ficheros donde se almacena la información. Esto es algo análogo a lo que presenta el sistema operativo de un ordenador personal, donde se almacena la información en ficheros, y éstos en carpetas. Para acceder a la información no basta con pedirla, sino que hay que conocer y pedir el fichero exacto en el que se encuentra. Para poder acceder a esa información, la tarjeta puede tener implementado algún mecanismo de seguridad para el control de acceso a la información, ya sea un PIN o un algoritmo de cifrado. En ese caso, habrá que satisfacer esos mecanismos para poder acceder a la información almacenada o escribir una nueva información.

Las categorías de los ficheros antes mencionados pueden dividirse en tres tipos fundamentales:

- ✓ Fichero Maestro (MF): representa la raíz de la estructura de ficheros.
- ✓ Fichero Dedicado (DF): contiene datos relativos a la aplicación. También puede contener Ficheros Elementales (EF). Esto es lo que en el argot de los ordenadores personales se denomina Carpeta o Directorio.

- ✓ Fichero Elemental (EF): última categoría posible. Contiene unidades de datos del sistema o de las aplicaciones. Esto es lo que en el argot de los ordenadores personales se denomina, simplemente, Fichero.

Es importante destacar las restricciones implícitas en esta distribución. Por ejemplo, en la tarjeta que se va a utilizar (conocida como WG10), el sistema operativo no permite más de dos niveles de profundidad, lo que se traduce en que no puede existir un DF dentro de otro DF. El número de ficheros que se pueden crear está limitado por la memoria disponible dentro de la tarjeta y por la codificación que se utilice para direccionarlos. A continuación un ejemplo de jerarquía:

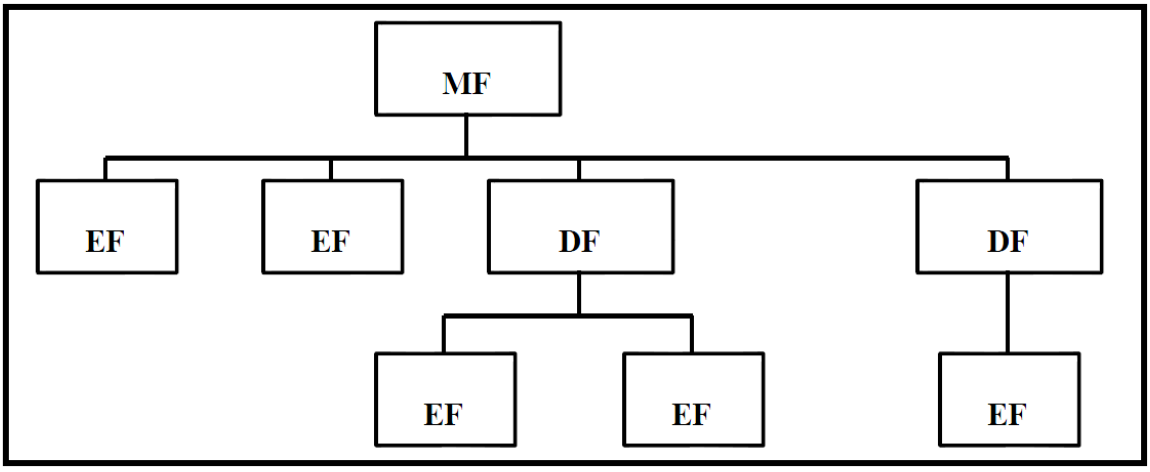


Figura 2: Organización de ficheros en tarjetas inteligentes (tomado de [4]).

Una vez definido ligeramente la organización interna de la tarjeta vamos a detenernos en explicar los comandos que se pueden llevar a cabo. Todos los comandos están compuestos por una cabecera de 5 bytes y opcionalmente por un campo de datos. Los 5 primeros bytes indican tanto el tipo de comando que se está ejecutando así como la longitud del campo de datos siempre que los haya. Cada uno de esos bytes se identifica con unas siglas y representa un parámetro concreto [4]:

- ✓ CLA: Clase de Instrucción.
- ✓ INS: Código de Instrucción
- ✓ P1 y P2: Referencia que complementa al código de instrucción, ya sea una dirección, un offset, etc.

- ✓ P3: Último byte para especificar el número de bytes que se transmiten con el comando.

Tras esos 5 bytes de comando, pueden seguirle más dependiendo del tipo de instrucción que se esté ejecutando. Por lo tanto una posible instrucción válida tendría el siguiente formato:

CLA	INS	P1	P2	P3	Datos

Existen múltiples instrucciones que puede procesar una tarjeta inteligente. A continuación se detallan los comandos más importantes para el objetivo del proyecto:

Comando	CLA	INS	P1	P2	Lc	Le
Select File por identificador por nombre	00h	A4h	 var 04h	00h	 02h var	var
Read Binary selección directa selección implícita	00h	B0h	 ofs sfi	ofs	00h	var
Verify secret code presentar número secreto leer número de intentos disponibles	00h	20h	00h	00h	 var 00h	00h
Get Response	00h	C0h	00h	00h	00h	var

Tabla 1: Comandos para interactuar con las tarjetas inteligentes (tomado de [4]).

Todos los comandos tienen como respuesta dos bytes (SW1 y SW2) que indican si se ha procesado bien la instrucción. Existen múltiples combinaciones para dichos bytes denominados bytes de estado, que explican si la ejecución fue correcta o el porqué ha fallado. Debido a la gran variedad de bytes de estado que existen, se decidió omitir la comprobación de éstos, considerando un SW1=0x90 y/o SW1=0x61 como ejecución correcta, y cualquier otra combinación, como ejecución incompleta[2].

[2] *Nota: No se estudia el comportamiento de SW2.*

3 PLATAFORMA DE DESARROLLO

En este capítulo vamos a proceder a explicar la plataforma donde se va a desarrollar el proyecto. Como se había comentado anteriormente, la unidad que va a controlar los accesos y gestionar los datos de identificación va a ser un microcontrolador, exactamente el microcontrolador STM32F107VC. Dicho microcontrolador se encuentra como unidad central de la placa de desarrollo STM3210C-EVAL [5]. Para simplificar su entendimiento es conveniente separar la explicación en dos partes: La placa de desarrollo y sus periféricos.

Figura 3: Fotografía de la placa de desarrollo STM3210C-EVAL (tomado de [5]).

Se comenzará explicando las características más destacables de la placa de desarrollo STM3210C-EVAL. Y posteriormente los periféricos necesarios para el desarrollo del proyecto. Cabe destacar, que los periféricos que tiene esta placa son muy numerosos y variados, sin embargo, no es necesario detenerse en todos ellos ya que algunos son totalmente irrelevantes para la consecución de los objetivos.

Por último comentar, que este capítulo se centrará más en las características del hardware y no tanto en las de configuración por software. Éstas últimas se dejarán para el capítulo 6 (*"DESARROLLO DEL SOFTWARE")* ya que aprender a configurar todos los periféricos ha sido una de las labores más costosas del presente proyecto. No por ello esta parte deja de ser importante, ya que buscar, encontrar, sintetizar y entender todas estas características ha formado parte del diseño del proyecto y sin su comprensión, el desarrollo no hubiera sido posible.

3.1 Placa de desarrollo

La placa de evaluación STM3210C-EVAL es una plataforma diseñada por la compañía STMicroelectronics basada en el microcontrolador STM32F107VCT con tecnología ARM y núcleo Cortex-M3[6].

La arquitectura ARM hace referencia a "Advanced RISC Machine", entendiendo por RISC "Reduced Instruction Set Computer". Dicha tecnología se basa en simplificar lo máximo posible el conjunto de instrucciones que puede contemplar un microprocesador. Esto consigue quitar peso en el procesado de datos y hacer más eficiente su tratamiento, utilizando sólo instrucciones específicas o su combinación [7].

El núcleo Cortex-M3 es utilizado para aplicaciones en tiempo real debido a su gran capacidad de procesamiento. Utiliza una arquitectura Harvard con instrucciones tipo Thumb (reduciendo las instrucciones más utilizadas en la tecnología ARM de 32 bits a 16). Además, dispone de una segmentación de la unidad de control (pipeline) en 3 etapas, un controlador de interrupciones vectorizadas NVIC (Nested Vector Interrupt Controller) y varios temporizadores (Timers) tanto de uso general como en tiempo

real. Éstas son sólo unas características de este tipo de tecnología, posteriormente se irán añadiendo más según se vayan explicando las funcionalidades [8].

A continuación se enumeran las características más importantes de las que dispone la placa. Aunque no todas ellas serán utilizadas a lo largo del proyecto, su conocimiento es fundamental para entender el comportamiento de la placa y estudiar posibles ampliaciones en la funcionalidad del proyecto [5].

- ✓ Tres tipos de alimentación a 5V: Mediante un transformador conectado a la red eléctrica, Conector MicroUSB tipo B y a partir de otra placa más potente.
- ✓ El arranque del sistema puede ser programable de entre las siguientes opciones: Memoria Flash, Memoria interna del sistema y Memoria SRAM.
- ✓ Audio conectado mediante I2S a un Conversor Digital Analógico.
- ✓ Puerto para tarjetas microSD.
- ✓ Interfaz para tarjetas inteligentes con contactos, ya sean de tipo A o de tipo B.
- ✓ Comunicación serie I2C para memoria tipo EEPROM de 64Kbit, MEMS y entradas/salidas de la placa.
- ✓ Comunicación RS-232.
- ✓ Receptor de infrarrojos (IrDA).
- ✓ Conexión USB (On The Go) a plena velocidad y conector USB microAB.
- ✓ Conector Ethernet según IEEE-802.3-2002.
- ✓ Pantalla táctil TFT de 3,2” 240x320 color LCD.
- ✓ Joystick de 4 direcciones y botón central.
- ✓ Cuatro pulsadores: Reset, Wakeup, Tamper y User.
- ✓ 4 LEDs.
- ✓ RTC con batería y sistema de “backup”.

El esquema de la placa se muestra en la siguiente figura. En ella se presentan todos los periféricos de los que dispone así como su localización.

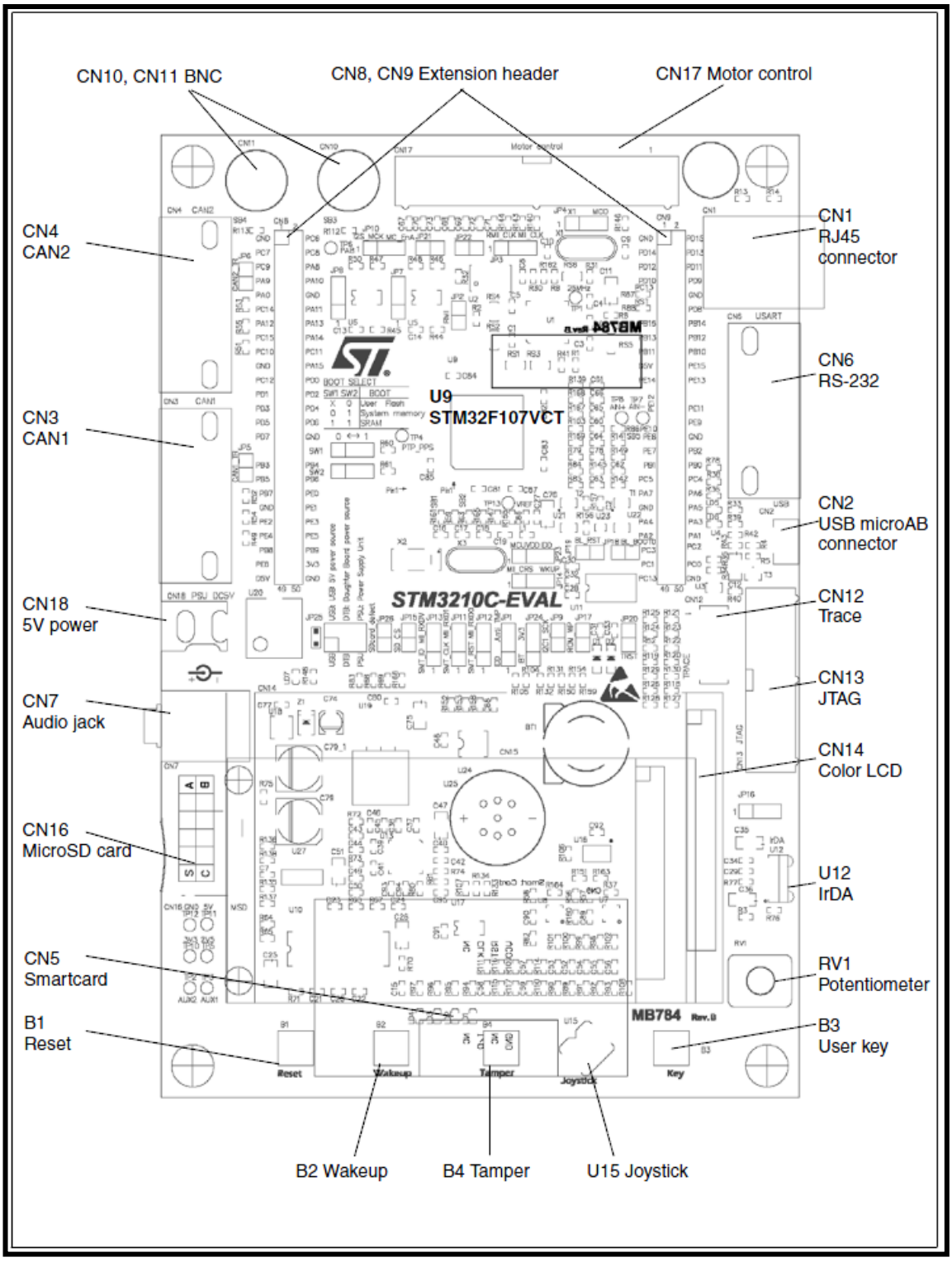


Figura 4: Esquema de la placa de desarrollo STM3210C-EVAL (tomado de [5]).

El microcontrolador, situado en el centro de la placa tal y como muestra la figura anterior, está encapsulado como en un chip LQFP (Low Profile Quad Flat Package) de

100 pines. A continuación se muestra un diagrama de bloques indicando el tipo de comunicación que utiliza para comunicarse con los distintos periféricos de la placa[3].

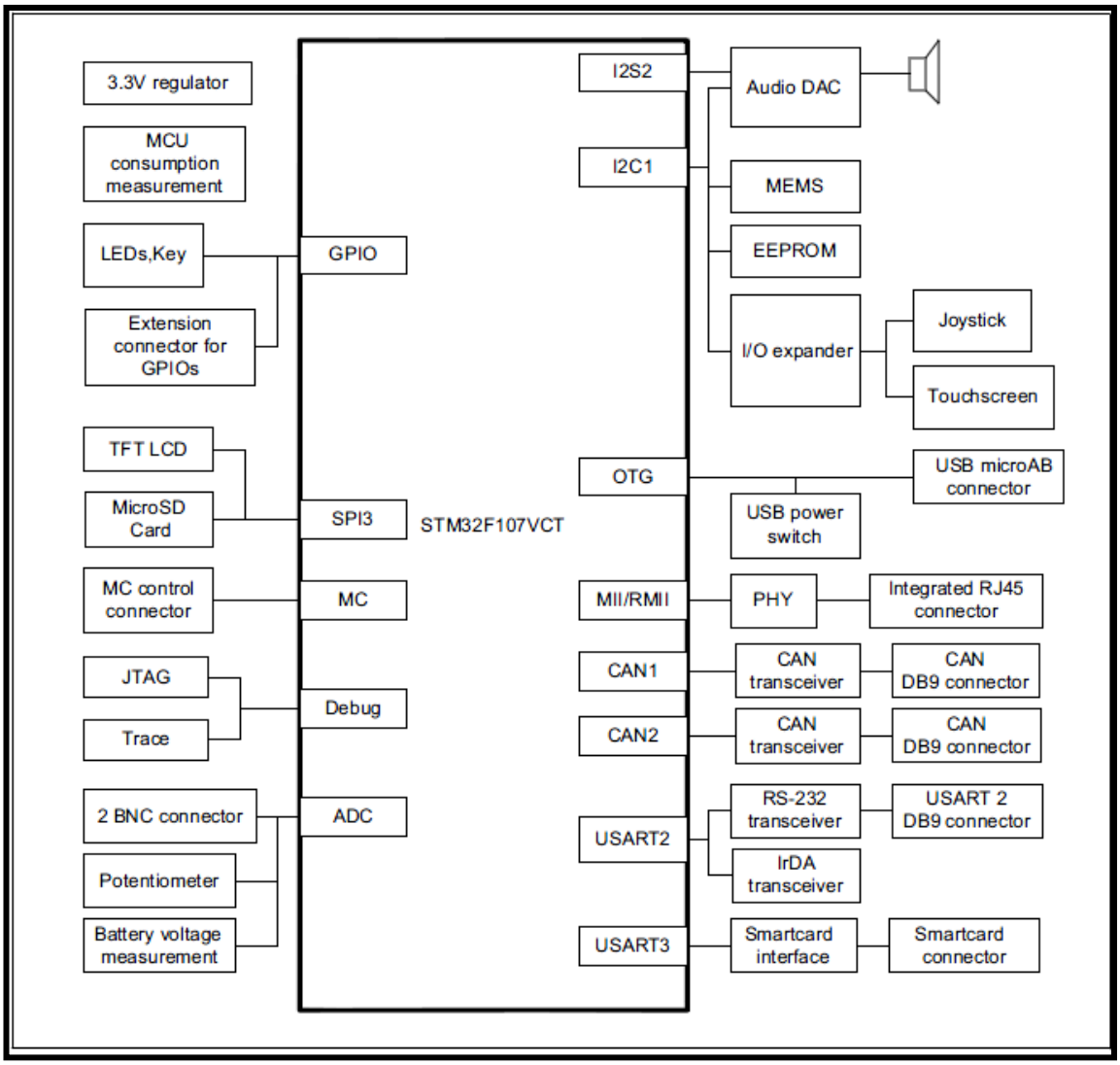


Figura 5: Diagrama de bloques del microcontrolador STM32F107VC (tomado de [5]).

Una vez conocidos todos los periféricos disponibles, se comenzará la explicación con las características relacionadas con la alimentación.

La STM3210C-EVAL está diseñada para trabajar con una alimentación de 5V DC, aunque dispone de un sistema de protección denominado PolyZen que protege a la placa de una tensión inadecuada.

[3] *Nota: No todos los periféricos mostrados en las figuras son utilizados en este proyecto. Más adelante se relacionará cada periférico utilizado con su tipo de comunicación y la función que realiza.*

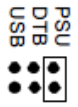
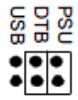
Jumper	Description	Configuration
JP25	JP25 selects one of the three possible power supply resources. For power supply jack (CN18) to the STM3210C-EVAL only, JP25 is set as shown: (**Default**)	
	For power supply from the daughterboard connectors (CN8 and CN9) to STM3210C-EVAL only, JP25 is set as shown:	
	For power supply from USB (CN2) to STM3210C-EVAL only, JP25 is set as shown:	
	For power supply from power supply jack (CN18) to both STM3210C-EVAL and daughterboard connected on CN8 and CN9, JP25 is set as shown to the right (the daughterboard must not have its own power supply connected):	
JP24	V_{bat} is connected to 3.3 V power when JP24 is set as shown: (**Default**)	
	V_{bat} is connected to battery when JP24 is set as shown:	

Tabla 2: Configuración Jumpers para la alimentación de la placa (tomado de [5]).

Se puede alimentar de tres formas distintas, sin influir en ningún momento en el rendimiento ni características operativas de la placa. La primera de ellas es conectando directamente a la red eléctrica, claro está, que debe pasar antes por un transformador que adecúe la señal de 230V AC a 5V DC. La segunda es a partir de otra placa con mayor potencial y la tercera, a partir de un cable MicroUSB tipo AB. Para seleccionar de entre todas las posibilidades hay que cambiar ciertos "Jumpers" de la placa, tal y como se puede ver en la Tabla 2.

Por otra parte, la placa está preparada para arrancar el sistema de tres formas distintas: desde una memoria Flash, desde su sistema de memoria central o desde una SRAM. Al igual que para la alimentación se deben configurar la placa externamente según se desee un modo u otro. Para el presente proyecto, el código del programa quedará grabado en la memoria tipo Flash.

Una vez mencionadas brevemente las características de alimentación y arranque del sistema, nos centraremos en explicar detalladamente los periféricos que se utilizarán

para la consecución de los objetivos. Para ello, es obligatorio comenzar explicando la arquitectura del sistema y de los buses, así como su distribución hacia los periféricos.

El sistema dispone de cuatro maestros, los cuales se comunican con los esclavos tal y como muestra la Figura 6, dónde los maestros y esclavos son los siguientes:

- ✓ Maestros: Núcleo Cortex M3, DMA1 y 2 y Ethernet DMA.
- ✓ Esclavos: Memoria Flash, Memoria SRAM y los puentes de los buses AHB y APB.

Para comprender las abreviaciones utilizadas en el diagrama de bloques es necesario redirigirse a la lista de abreviaciones que se incluye al principio del proyecto.

La siguiente figura es un diagrama de bloques especificando los maestros y esclavos del sistema, así como la conexión entre los diferentes buses y los distintos periféricos utilizados [9].

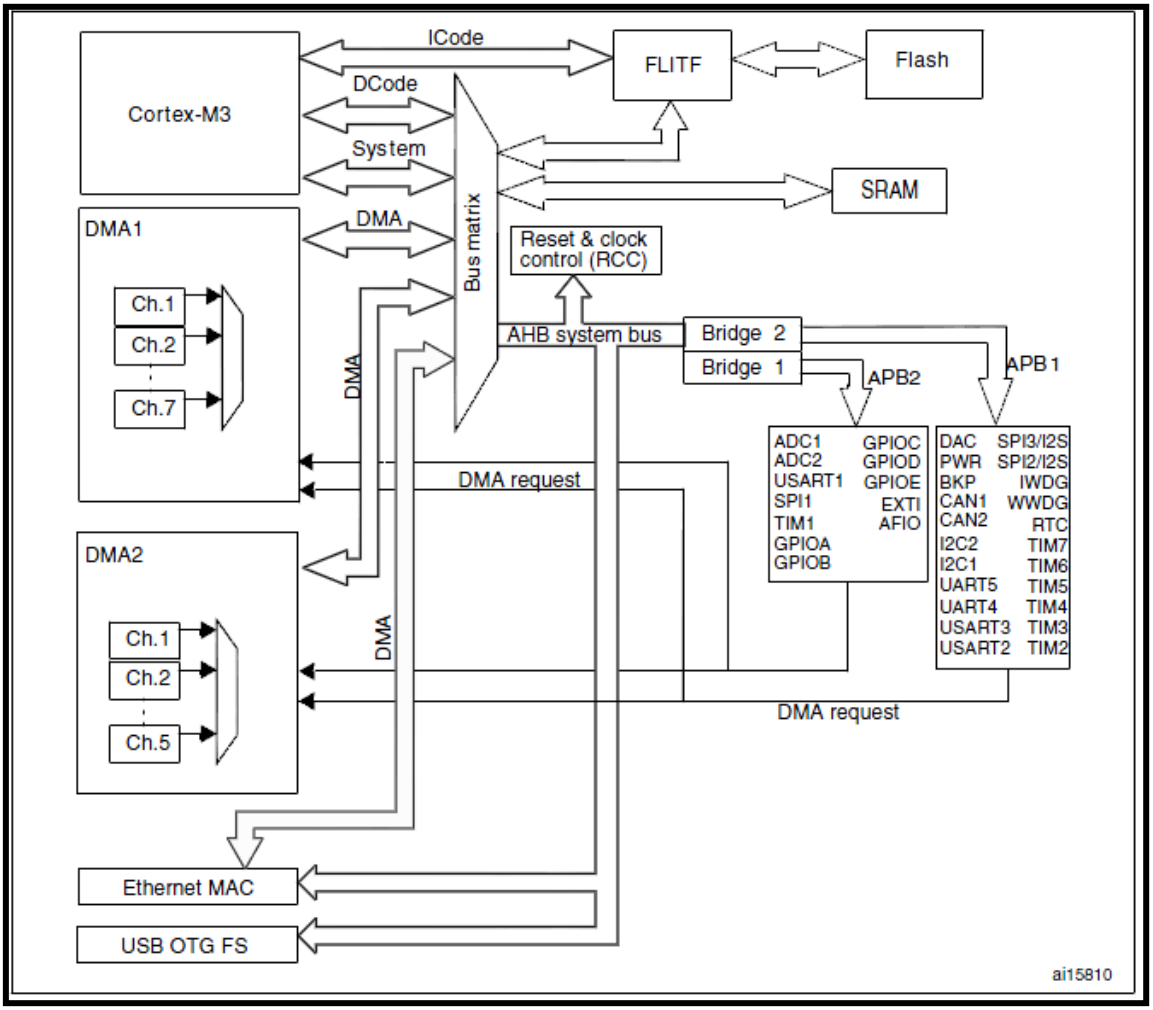


Figura 6: Diagrama de Bloques para la comunicación con los periféricos (tomado de [9]).

Los buses de comunicación entre los diferentes componentes son los siguientes:

- ✓ ICode bus: Conecta el núcleo Cortex M3 con las instrucciones en la memoria Flash. Se utiliza para buscar dichas instrucciones.
- ✓ DCode bus: Conecta el núcleo Cortex M3 con la memoria Flash para obtener los datos de ésta.
- ✓ System bus: Conecta el núcleo Cortex M3 con la matriz de buses encargada de manejar los permisos de ambos maestros (Núcelo y DMA).
- ✓ DMA bus: Permite el acceso directo a memoria del núcleo mediante la conexión del DMA a la matriz de buses encargada de facilitar el acceso del núcleo y el DMA tanto a las memorias Flash y SRAM, como a los periféricos.
- ✓ Bus Matrix (Matriz de buses): Se encarga de manejar los permisos del núcleo y del acceso DMA mediante un algoritmo Round Robin
- ✓ Bridges AHB/APB (Puentes AHB/APB): Estos puentes sirven para que la comunicación entre los diferentes buses sea posible. El AHB es el bus con mejores características, por lo que está conectado a la matriz de buses directamente permitiendo el acceso directo a memoria. Los buses APB son los que van a los periféricos y disponen de diferentes especificaciones. Mientras que el APB1 sólo puede trabajar a un máximo de 36 MHz, el APB2 puede trabajar hasta un máximo de 72 MHz.

Una vez que conocidas las características más básicas de la placa de desarrollo que se va a utilizar, se procederá a explicar los periféricos.

3.2 Periféricos

La explicación de los periféricos se va realizar detalladamente por ser una de las partes principales del proyecto, ya que gracias a ellos, se podrán cumplir los objetivos que en un principio se habían definido.

Para ello los periféricos se dividirán en dos grandes bloques con el fin de simplificar su entendimiento. Dichos bloques se dividen según la funcionalidad que desempeñe cada periférico en la totalidad del proyecto.

El primer bloque se centrará en definir los periféricos utilizados de forma general para el correcto funcionamiento del sistema. Entre ellos se destacan: el RTC y los modos de bajo consumo.

Mientras que en el segundo bloque se describirán los periféricos utilizados para el control de accesos: lector de tarjetas inteligentes y la pantalla táctil.

En este capítulo no se explicarán los periféricos que se utilizan de forma general en cualquier diseño (RCC, EXTI, NVIC, etc.) debido a que el objetivo de este capítulo no es describir la plataforma de desarrollo en su totalidad sino sólo los periféricos más importantes para alcanzar los objetivos propuestos.

3.2.1 RTC (Real Time Clock)

Este periférico es crítico para el correcto desempeño de las funciones de un sistema de accesos, ya que uno de los principales parámetros que debe estudiar continuamente la placa es el tiempo.

El reloj funciona con un temporizador independiente de la placa, capaz de funcionar de forma continua como contador. Esto permite, usando la configuración y el software apropiado, diseñar un reloj que mida el tiempo de forma real. Es decir, gracias a este reloj podemos definir un calendario propio dentro del sistema.

Sin embargo, en esta placa el RTC es un poco pobre en cuanto a posibilidades, porque tiene un único registro para almacenar el tiempo transcurrido y únicamente en un determinado número de unidades. La fecha (día, mes, año), así como la división en horas, minutos y segundos, debe definirse de forma adicional incluyendo una función que lo calcule. Esta opción está normalmente

integrada en los relojes en tiempo real de otros microcontroladores, pero en este caso ha tenido que desarrollarse manualmente.

Otra ventaja de un reloj en tiempo real es que su funcionamiento se apoya en los registros del "Backup", evitando la pérdida de información cuando se realiza un Reset o cuando entra en los modos de bajo consumo[4]. Para ello, la STM3210C-Eval dispone de una pila que mantiene la mínima tensión para guardar dicha información y que una vez se desconecte la fuente de alimentación, el contador de tiempo siga funcionando.

A continuación se enumeran las características principales de este periférico con la intención de demostrar la multitud de posibilidades que ofrece su utilización:

- ✓ Pre-escalado programable: factor de división hasta 2^{20}. Dispone de dos registros independientes funcionando como pre-escalado. Uno de ellos sirve para definir la frecuencia de la cuenta (f_{TR_CLK}), mientras que el otro sirve como almacenamiento de dicho valor. Gracias a este doble registro, el contador no necesita detenerse en ningún momento, consiguiendo así, una muy buena precisión.

$$f_{TR\_CLK} = \frac{f_{RTCCLK}}{(PRL[19:0] + 1)}$$

- ✓ Dispone de un registro de hasta 32-bits programables permitiendo medir grandes cantidades de tiempo.
- ✓ Dos relojes distintos: PCLK1 para el APB1 y el RTC (el cual debe ser al menos cuatro veces más lento que el PCLK1)
- ✓ El RTC se puede configurar a partir de cualquiera de los distintos relojes del sistema: HSE, dividido entre 128; Oscilador LSE; Oscilador LSI.
- ✓ Tres líneas de interrupción posibles. Las líneas de interrupción sirven para que el sistema automáticamente genere una interrupción. Se pueden definir tres modos: Interrupción por alarma, la cual se dispara

[4] *Nota: Existen tres modos de bajo consumo: Sleep Mode, Stop Mode y Standby Mode. Todos se explicarán más adelante, sin embargo, cabe destacar que el RTC mantiene su valor independientemente del modo en el que se encuentre/despierte*.

cada vez que el valor del contador llega a un valor definido; Interrupción en segundos, donde se puede definir una interrupción periódica de hasta 1 segundo de periodo; y la interrupción de Overflow, para detectar posibles desbordamientos.

En la siguiente figura se puede observar el diagrama de bloques del periférico como ayuda al entendimiento de su distribución, así como la función de sus registros principales.

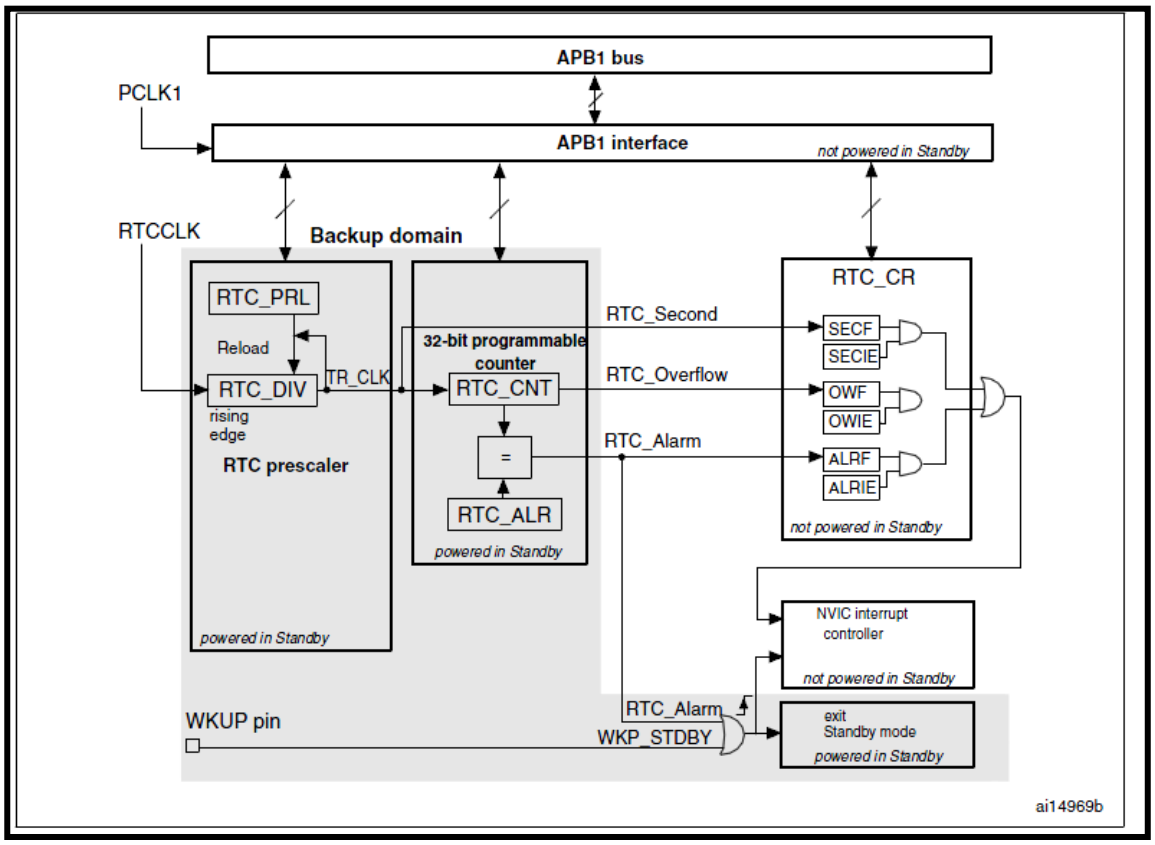


Figura 7: Diagrama de Bloques del RTC (tomado de [9]).

En ella se observa que el sistema se compone de tres bloques principales. El primero de ellos es el encargado de configurar el periodo de pulso, o en otras palabras, la frecuencia de cuenta. Esto se realiza configurando los registros de los pre-escalados que se habían mencionado anteriormente.

El segundo bloque consta del registro de 32-bits programable. Se utiliza para almacenar la cuenta de pulsos, es decir, como contador. Además, en este

segundo bloque también se puede configurar el valor de la alarma y compararlo con el valor del RTC, permitiendo así, la interrupción por alarma.

Y por último, en el tercer bloque se establecen los registros de control, donde se pueden configurar las interrupciones de las formas que ya se habían comentado: la interrupción por alarma, la interrupción por desbordamiento y la interrupción periódica, la cual se utilizará en este proyecto. Simplemente basta con activar dicha interrupción en los registros de control y programar el pre-escalado para que se dispare periódicamente cada segundo[5].

3.2.2 Modos de bajo consumo

Los modos de bajo consumo son modos de operación en los cuales los sistemas no se encuentran cumpliendo con su función primaria. Gracias a esta posibilidad de desconectar periféricos, son capaces de reducir notablemente su consumo energético.

Existen múltiples tipos de bajo consumo en relación con la energía que gasta y las características que anula para conseguirlo. Aunque sus definiciones varían según la fuente, existe uno estandarizado que es el modo Standby.

Dicho modo se define como el modo de bajo consumo que menor consumo consigue a partir de la desactivación de la mayoría de sus características. Sin embargo, es muy cierto lo que afirman los estudios realizados de que su uso gasta mucha más energía de lo que la sociedad piensa. Todos los aparatos electrónicos que “apagamos” y en verdad ponemos en modo Standby pueden llegar a suponer un 11% del consumo residencial.

En el presente proyecto, la STM3210C-EVAL dispone de tres modos de bajo consumo. Obviamente solo se utilizará el que mejor se adapte a las condiciones

[5] *Nota: Este procedimiento se explicará con más detalle en el capítulo 6 “DESARROLLO DEL SOFTWARE” en el apartado del RTC*

necesarias para satisfacer los objetivos propuestos, pero para tomar esa decisión es necesario estudiar los tres:

- ✓ **Sleep Mode**: Este modo es el que más alto consumo energético tiene ya que es el que menos periféricos desactiva. Pero a su favor tiene que es el más rápido en despertar y volver a funcionar de modo normal.

  Tan sólo desactiva el reloj de la CPU, todos los demás periféricos siguen en marcha y cumpliendo su función.

  Se puede entrar en este modo de dos formas distintas. Una de ellas, denominada como "SLEEP-NOW", donde el microprocesador entra en ese estado tan pronto como la condición programada se ejecute. Y la otra llamada "SLEEP ON EXIT" que lo hace si existe la menor prioridad en el ISR.

  Para salir, simplemente se tiene que ejecutar alguna interrupción o evento ya sea con algún periférico interno (p.e. RTC) o por alguna intrusión externa. **No hace falta que dicha interrupción/evento tenga una configuración determinada**.

- ✓ **Stop Mode**: En este caso existe un balance entre la energía que gasta y la rapidez en despertarse. En comparación con el anterior, su consumo es menor, ya que desactiva más características y induce al núcleo Cortex-M3 a un estado "dormido" y consecuentemente su tiempo de "wakeup" es mayor ya que necesita volver a activarlos.

  En este modo se apagan los siguientes relojes: el PLL, el HSI, y el HSE RC. Se mantiene a la espera hasta que una de las interrupciones o eventos configurados se ejecuta. Permite la posibilidad de regular la tensión a 1.8 V.

  Para entrar, se debe configurar los registros correspondientes tal y como muestra la tabla resumen del final de este apartado. Y para salir basta con que se produzca alguna interrupción o evento que esté **configurado en los registros de EXTI.**

- ✓ **Standby Mode**: Por último, este modo es el que más bajo consumo produce, tal y como se había anticipado en la explicación anterior, pero en contraposición es el más lento a la hora de alcanzar de nuevo el modo normal de funcionamiento.

  Además de desactivar todos los relojes, desactiva también el regulador de tensión y mantiene la alimentación a 1.8V. Se pierden también todos los registros de la SRAM que no se guarden en los "Backup", mientras que en los anteriores sí permanecían guardados sin necesidad de estos registros.

  Para entrar hay que seguir un procedimiento parecido al de Stop Mode pero con la diferencia que se activa otro bit adicional indicando el "apagado funcional" del sistema. Mientras que para salir requiere de algún aviso de determinados periféricos: WKUP pin, Alarma RTC, Reset externo en NRST, IWDG Reset.

Para facilitar la comprensión de estas configuraciones se añaden las tablas correspondientes a cada modo indicando: cómo entrar, cómo salir y el tiempo que tardar en despertar:

Sleep Mode:

Sleep-now mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: – SLEEPDEEP = 0 and – SLEEPONEXIT = 0 Refer to the Cortex™-M3 System Control register.
Mode exit	If WFI was used for entry: Interrupt: Refer to *Section 10.1.2: Interrupt and exception vectors on page 190* If WFE was used for entry Wakeup event: Refer to *Section 10.2.3: Wakeup event management*
Wakeup latency	None
Sleep-on-exit	**Description**
Mode entry	WFI (wait for interrupt) while: – SLEEPDEEP = 0 and – SLEEPONEXIT = 1 Refer to the Cortex™-M3 System Control register.
Mode exit	Interrupt: refer to *Section 10.1.2: Interrupt and exception vectors on page 190.*
Wakeup latency	None

Tabla 3: Configuración del Sleep Mode (tomado de [9]).

Stop Mode:

Stop mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: – Set SLEEPDEEP bit in Cortex™-M3 System Control register – Clear PDDS bit in Power Control register (PWR_CR) – Select the voltage regulator mode by configuring LPDS bit in PWR_CR **Note:** To enter Stop mode, all EXTI Line pending bits (in *Pending register (EXTI_PR)*) and RTC Alarm flag must be reset. Otherwise, the Stop mode entry procedure is ignored and program execution continues.
Mode exit	If WFI was used for entry: Any EXTI Line configured in Interrupt mode (the corresponding EXTI Interrupt vector must be enabled in the NVIC). Refer to *Section 10.1.2: Interrupt and exception vectors on page 190*. If WFE was used for entry: Any EXTI Line configured in event mode. Refer to *Section 10.2.3: Wakeup event management on page 199*
Wakeup latency	HSI RC wakeup time + regulator wakeup time from Low-power mode

Tabla 4: Configuración del Stop Mode (tomado de [9]).

Standby Mode:

Standby mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: – Set SLEEPDEEP in Cortex™-M3 System Control register – Set PDDS bit in Power Control register (PWR_CR) – Clear WUF bit in Power Control/Status register (PWR_CSR)
Mode exit	WKUP pin rising edge, RTC alarm event's rising edge, external Reset in NRST pin, IWDG Reset.
Wakeup latency	Reset phase

Tabla 5: Configuración del Standby Mode (tomado de [9]).

Por último es muy importante resaltar que al activarse cualquiera de estos modos de bajo consumo, **el modo debug (depuración) no se puede mantener**. Es decir, la placa de desarrollo no está preparada para trabajar en bajo consumo mientras se depura.

Existe una opción para que el modo debug se mantenga activado cuando se utilizan los modos de bajo consumo. Sin embargo, no es de gran utilidad ya que simplemente evita que el sistema entre en ese estado.

3.2.3 Lector de tarjetas inteligentes:

La tarjeta inteligente se utilizará como primera opción para identificar al usuario que pretende acceder a un lugar de la empresa/institución. De esta forma, es imprescindible explicar cómo se hace posible la comunicación entre la tarjeta inteligente y el microcontrolador.

Antes de nada, es importante destacar que se deben modificar ciertos "jumpers" en la placa de desarrollo para posibilitar el uso de este periférico. Estas modificaciones se resumen en la siguiente tabla[6]:

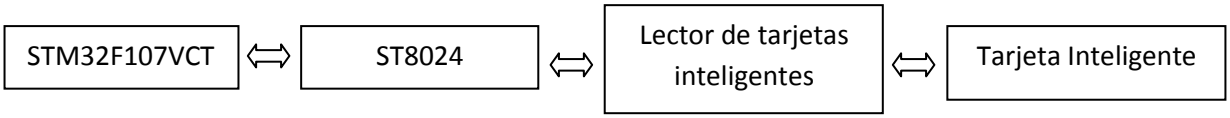
Jumper	Description	Configuration
JP11	PD10 is connected to ethernet PHY as signal MII_RXD1 when JP11 is set as shown: (**Default**)	
	PD10 is connected to smartcard interface chip as signal smartcard_CLK when JP11 is set as shown:	
JP12	PD9 is connected to ethernet PHY as signal MII_RXD0 when JP12 is set as shown: (**Default**)	
	PD9 is connected to smartcard interface chip as signal smartcard_RST when JP12 is set as shown:	
JP13	PD8 is connected to ethernet PHY as signal MII_RX_DV/RMII_CRSDV when JP13 is set as shown: (**Default**)	
	PD8 is connected to smartcard interface chip as signal smartcard_IO when JP13 is set as shown:	

Tabla 6: Configuración Jumpers para Tarjeta Inteligente (tomado de [5]).

Para utilizar la tarjeta inteligente como medio de identificación, se definen cuatro bloques necesarios para establecer la comunicación. El primero de ellos es sin ninguna duda el microcontrolador, encargado de manejar todo el sistema. El segundo bloque es un chip, denominado ST8024, especializado en traducir las instrucciones programadas del microcontrolador en operaciones eléctricas dentro del chip de la tarjeta, y viceversa. El tercer bloque es el lector de tarjetas, periférico de la placa y dispositivo físico donde se conecta la tarjeta.

[6] *Nota: En esta tabla se observa la imposibilidad física que se tiene de configurar esta placa de desarrollo para utilizar simultáneamente la tarjeta inteligente y la comunicación por Ethernet. Esta aspecto se comentará de nuevo en capítulos posteriores.*

Y por último la tarjeta inteligente. Se añade un diagrama de bloques como esquema gráfico con el fin de facilitar la comprensión.



Cada uno de los bloques definidos tiene su propia función dentro de la comunicación. El microcontrolador será el encargado de ejecutar las instrucciones programadas, así como, mandar/recibir datos de la tarjeta.

Sin embargo, dicha comunicación no es directa. Como ya se había mencionado antes, el chip ST8024 es el encargado de traducir dichas instrucciones en operaciones eléctricas detectables por la tarjeta inteligente. Por lo que es quién realmente se comunica con la tarjeta.

Por lo tanto el primer aspecto que a tratar es la conexión entre el microcontrolador STM32F107VC y el chip ST8024. Aunque dicha conexión se adjunta detalladamente en el Anexo 9.2.1 "*SmartCard*", se incluye en esta sección la siguiente tabla para identificar los pines necesarios que se van a utilizara lo largo del desarrollo.

Signals of ST8024	Description	Connect to STM32F107VCT
5V/3V	Smartcard power supply selection pin	PC0
I/OUC	MCU data I/O line	PD8
XTAL1	Crystal or external clock input	PD10
OFF	Detect presence of a card, interrupt to MCU	PE7
RSTIN	Card reset input from MCU	PD9
CMDVCC	Start activation sequence input (active low)	PE14

Tabla 7: Definición de GPIOs para establecer comunicación con una tarjeta inteligente (tomado de [5]).

Observando la tabla, se comprueba que este chip resulta muy útil para comunicarse con una tarjeta inteligente ya que reduce todas sus complejas conexiones a la configuración de 6 GPIOs. Una vez que dichos GPIOs se

configuren correctamente, se podrá establecer comunicación entre la tarjeta y el microcontrolador.

La conexión completa entre el chip ST8024 se encuentra en el Anexo 9.2.2 "*ST8024*" porque su explicación, además de ser demasiado compleja para detenerse en ella, es irrelevante ya que la función del ST8024 reside exactamente en simplificar el conexionado y la comunicación entre microcontrolador y tarjeta.

El segundo aspecto a tratar es el tipo de comunicación. Se va a utilizar una comunicación por puerto USART, configurada de forma asíncrona y semi-dúplex[7]. Sin embargo, se observa en la tabla anterior que dispone de un pin de reloj para establecer la comunicación, esto se debe a que el sistema debe proporcionar una salida de reloj para calcular el baud rate.

Este término se define como el número de unidades de señal por segundo y es imprescindible su correcta definición ya que al tratarse de una comunicación asíncrona, si no se define bien, la frecuencia de recepción y la de envío serán distintas. Provocando un desfase entre los datos y por tanto una comunicación errónea.

Tal y como se había adelantado en el capítulo 2, apartado *"Tarjetas Inteligentes"*, para iniciar la comunicación con una tarjeta inteligente es necesario realizar un reset en frío y esperar el ATR. Dicho proceso no es tan sencillo como en un principio parece, ya que la comunicación con la tarjeta se va a realizar de forma bidireccional y por un solo canal, es decir, semi-duplex.

Para finalizar, y al tratarse de un aspecto más relacionado con el hardware que con el software, se explicará en este apartado el procedimiento específico que debe seguirse para iniciar la comunicación. Dicho procedimiento, estandarizado en ISO 7816-3, se resume a continuación:

- ✓ Conexión mecánica de los contactos entre la tarjeta y la placa.

[7] *Nota: Este tipo de puerto tiene la posibilidad de funcionar de forma asíncrona o síncrona y con comunicación full-duplex o semi-duplex.*

- ✓ Activación de los contactos según la siguiente secuencia:
  - PC0 = 1. Alimentación activada a 5V.
  - PD9 = 0. Reset de entrada desactivado.
  - PD8. Configurado como recepción.
  - PD10. Debe llegarle una señal de reloj interna.
- ✓ Reset frío de la tarjeta.
  - PE14 = 0. Preparado para la transmisión
  - PD9 = 1. Reset de entrada activado.
- ✓ Espera del ATR.

3.2.4 Pantalla Táctil

La pantalla táctil servirá de interfaz entre el sistema y el usuario. Esto se traduce en el desempeño de una función fundamental y que conlleva una explicación pormenorizada de sus características.

Se trata de una pantalla táctil resistiva de tecnología TFT-LCD de 3,2". Las pantallas del tipo resistivas están formadas por una serie de capas que, a través de la presión externa, generan una pequeña resistencia entre dos capas de un material conductor separadas por pequeños huecos [10]. Dicha resistencia es medida para obtener el punto exacto de presión, por lo que el material del objeto no influirá para detectar el contacto con la pantalla.

Las pantallas con tecnología TFT-LCD (Thin Film Transistor-Liquid Crystal Display) utilizan una matriz de transistores que mediante una serie de capas de polarización y de color, consiguen definir una imagen de buena calidad [11].

Las dimensiones de dicha pantalla son de 3,2", tal y como se definió anteriormente, y con una resolución de 240x320 píxeles.

La pantalla se conecta a la plataforma gracias al conector de 34 pines denominado CN14[8] y utiliza una comunicación SPI (Serial Peripheral Bus) donde la información se transmite de forma serie síncrona.

Por último comentar que dicha interfaz de comunicación se puede configurar de hasta cuatro modos diferentes, pero el principio de funcionamiento es el mismo en todos ellos. Utiliza la comunicación Full-Dúplex mediante dos líneas de transmisión de datos: MOSI (Master-Out/Slave-In) y MISO (Master-In/Slave-Out). La primera de ellas transmite la información del maestro al esclavo, mientras que línea MISO transmite del esclavo hacia el maestro. Dispone de otra línea para el reloj, ya que se trata de una comunicación síncrona, y en ocasiones un cuarto hilo trabajando como Chip-Select [12].

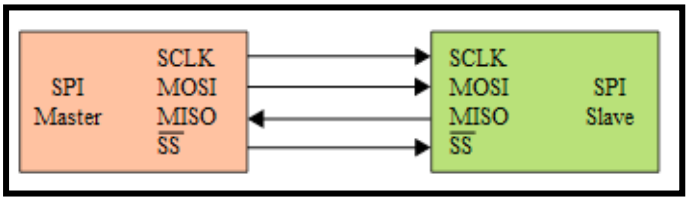


Figura 8: Diagrama de Bloques de Comunicación SPI (tomado de [12]).

[8] *Nota: La STM3210C-EVAL dispone de la posibilidad de conectar una pantalla TFT LCD auxiliar (no táctil) de 2,4". No se contempla su utilización, por lo que no se explicará nada sobre ella.*

4 METODOLOGÍA Y REQUISITOS

Este capítulo se va a dividir en dos bloques fundamentales. El primero de ellos es el apartado de metodología, el cual se centrará en explicar cuál ha sido la metodología de trabajo, cómo se ha enfocado el desarrollo y cuál ha sido su evolución.

En el segundo se explicarán los objetivos que se han definido en este proyecto así como su utilización e importancia en la sociedad.

4.1 Metodología

El trabajo para conseguir elaborar el presente proyecto fin de carrera se ha dividido en cuatro fases:

1. Aprendizaje de conceptos básicos sobre los microprocesadores. Esta fue la fase inicial por las pocas nociones que se tenía de esta tecnología.

   Aquí se recogió gran cantidad de información procedente de tres fuentes principales: apuntes de la asignatura de Microprocesadores [8], documentación suministrada por Raúl Sánchez Reíllo [3] [4] y documentación encontrada a través de Internet.
2. Localización de documentación de la placa de desarrollo STM3210C-EVAL. Tras aprender los conceptos básicos de los microprocesadores se empezó a buscar información sobre la placa con la que se trabajaba.

   Esta búsqueda fue un fracaso por la poca documentación que existía al principio en la página oficial de STMicroelectronics. En dicha página, no había ni bibliotecas, ni ejemplos disponibles. Por lo que se tuvo que empezar la fase 3 basándonos en los ejemplos que el entorno de desarrollo suministraba al instalarse. Estos ejemplos no servían para la placa STM3210C-EVAL, pero tuvieron que ser utilizados para no retrasar demasiado el comienzo del

proyecto. Aún así, se decidió mandar un email a la empresa indicando que no existía documentación suficiente para trabajar con esta placa de desarrollo.

3. Comienzo del proyecto sin bibliotecas específicas. Esta fase fue una de las más costosas por la necesidad de estar traduciendo los ejemplos de otros microcontroladores al microcontrolador STM32F107VC. Aquí sólo se desarrollaron algunos programas para empezar a aprender cómo funcionaba el microcontrolador y cómo configurar sus periféricos.
4. Disponibilidad de bibliotecas y ejemplos. Al aparecer varios recursos en la página oficial (bibliotecas, ejemplos y manuales específicos de ciertos periféricos) el desarrollo de la aplicación fue mucho más sencillo y rápido.

   Fue en esta etapa cuando se comenzó definitivamente a aprender cómo trabajar con el microcontrolador.

   Aunque no toda la documentación colgada era válida, ya que algunos de los manuales o ejemplos no funcionaban, se pudo desarrollar el proyecto con relativa rapidez.
5. Tarjetas inteligentes. Una vez que se aprendió a manejar el microcontrolador, el documentarse sobre las tarjetas inteligentes fue el siguiente objetivo.

   Esta documentación [3] fue suministrada por Raúl Sánchez Reíllo y una vez estudiada, se comenzó a incluir la tarjeta inteligente como método de identificación.

   La configuración de las tarjetas inteligentes fue realizada de forma específica por el tutor de este proyecto, utilizando herramientas internas del Grupo de Investigación y utilizando su ordenador de trabajo.
6. Finalización del proyecto y pruebas. Como última fase, se terminó de desarrollar el código del programa y se comenzó a optimizar el diseño hasta llegar al resultado final.

Por último añadir que hay dos posibles formas de programar los microcontroladores. Una es utilizando las funciones definidas por bibliotecas o ejemplos y la otra es mediante la modificación directa de los registros.

En este proyecto se han utilizado ambas ya que cada una tiene sus correspondientes ventajas e inconvenientes.

4.2 Requisitos

Los sistemas de control de accesos han sido, son y serán siempre uno de los grandes objetos de estudio existentes en las empresas/instituciones. Actualmente en la sociedad existen numerosos tipos de sistemas de control de accesos, teniendo todos ellos como propósito el control de personas o vehículos. En cualquiera se debe cumplir una serie de características básicas:

- ✓ Seguridad: El sistema se define como control de accesos, por lo que debe tener un nivel de seguridad acorde con el fin que se quiera.
- ✓ Fiabilidad: Muy relacionada con la seguridad, la fiabilidad es un aspecto clave para que la seguridad sea estable.
- ✓ Precio: Cualquier sistema que conlleve la participación de un gran número de personas no debe ser demasiado caro o su participación en el mercado se verá afectado por el de sus competidores.
- ✓ Flexibilidad: Este tipo de sistemas suelen ser de carácter marcadamente dinámico, como consecuencia se suelen buscar diseños que tengan relativamente poca complejidad a la hora de ser modificados para: incluir/borrar usuarios, modificar contraseñas de seguridad, actuaciones en caso de emergencia, etc.
- ✓ Accesibilidad. Hay que recordar en el diseño de cualquier sistema que existen personas con discapacidades que tienen los mismos derechos que cualquier otro ciudadano. Por lo tanto siempre tiene que haber un estudio para facilitar el acceso a estas tecnologías a dichos usuarios.

En términos técnicos se da una fuerte contraposición entre seguridad y flexibilidad. Ambas características son aspectos vitales en el diseño de un sistema de control de accesos, pero en la mayoría de los casos se debe optar por elegir una de ellas como primer foco de estudio, quedando la segunda relegada al segundo puesto. En otras palabras, cuánto más seguro se quiere que sea el sistema, menos flexible va a ser a la hora de realizar cambios inesperados. Y por supuesto, cuánto más rápido se quiera adaptar al cambio, menos seguro será.

Un buen equilibro entre las características antes mencionadas asegura un buen diseño de un sistema de control de accesos. Por otra parte, la mejora de cada uno de esos factores está relacionada con un posible uso en la sociedad. A continuación se enumerarán ciertos ejemplos de posibles usos de un sistema de control de accesos y en qué característica se debería de focalizar su diseño.

- ✓ Empresa/Institución: Este sea posiblemente la utilización más común de este tipo de sistemas. Se utiliza para controlar a los empleados/visitantes mediante tarjetas que gestionen los accesos y guarden fechas y hora tanto de entrada como de salida. Se busca un perfecto equilibrio entre la seguridad y la flexibiliad.
- ✓ Centros Médicos: En este caso, el propósito es controlar la localización de personas con discapacidad, personas mayores o con algún tipo de enfermedad. En este caso no es tan importante la seguridad y si lo es la fiabilidad y accesibilidad.

El presente proyecto no está diseñado para ninguno de los tipos definidos y por tanto debe ser lo más general posible para poder adaptarse a cualquiera de ellos.

Para finalizar con esta exposición sobre requisitos, se enumeran algunos de ellos que se creen importantes para la correcta consecución de los objetivos y conseguir una base adecuada para desarrollar sistemas completos cualquiera que sea su uso futuro:

- ✓ Seguridad: El sistema debe mantener un nivel de seguridad medio-elevado para poder ser utilizado en cualquier empresa/institución que precise de dichos niveles.
- ✓ Fiabilidad: El sistema debe estar preparado para seguir funcionando en caso de que haya algún fallo interno o externo
- ✓ Accesibilidad: Evitar posibles problemas característicos para la accesibilidad.
- ✓ Precio: Mantener una buena relación calidad-precio

5 DISEÑO DEL SISTEMA

Es de fuerte importancia volver a matizar que no se va a realizar un sistema de control de accesos definitivo, sino un prototipo que muestre algunas de las posibilidades que tiene este tecnología aunque para ello se comprometa el nivel de seguridad del sistema.

5.1 Sistemas de identificación

El primer aspecto a tratar fue decidir cuál o cuáles iban a ser los sistemas de identificación de los usuarios. Se disponía de un gran abanico de posibilidades: sistemas de identificación biométrica, usuario y contraseña, tarjetas inteligentes, tarjetas magnéticas, etc.

La primera opción fue elegir los sistemas de identificación biométrica. Dichos sistemas contienen de forma implícita una gran seguridad ya que se basan en parámetros únicos de cada individuo. Sin embargo, tal y como se explicaba anteriormente, disponen de muy poca facilidad para introducir nuevos usuarios y además, es un sistema tan delicado que repercuten de forma muy drástica los desgastes de los componentes.

Aparte de lo anterior, la placa de desarrollo no dispone de ningún periférico que soporte dicha forma de identificación y habría que añadírselo por alguno de los puertos que tiene. De esta forma el sistema además de encarecerse, puede producir fallos de compatibilidad.

La segunda opción, y además elegida, fue basar el sistema en identificación por tarjetas inteligentes. Este tipo de tecnología, tal y como se ha descrito en el capítulo 2 "*ESTADO DE LA TÉCNICA*", es un método de identificación muy seguro, que permite una fácil adaptación a nuevos usuarios (simplemente configurando una nueva tarjeta), una enorme fiabilidad y precio económico.

Esta decisión fue promovida por el gran desconocimiento que se tenía de esta potente tecnología y la gran personalización que ofrece trabajar con ella. Otro incentivo fue saber que la placa con la que se iba a trabajar disponía ya del lector de tarjetas haciendo, en un principio, más barata y sencilla la consecución de los objetivos.

Aunque las tarjetas tienen un grado de seguridad aceptable de serie, ya que es imprescindible saber dónde se encuentra la información para poder leerla se optó por incluir mayor seguridad ya que iba a ser nuestro principal sistema de identificación.

Este sistema de seguridad adicional fue configurar un código PIN para acceder a la información que hay en ellas. Es decir, no basta con introducir la tarjeta en el sistema para que nos de acceso a la zona deseada, además se necesita teclear un código de seguridad numérico que ha de coincidir con el grabado en la tarjeta. Esto evita posibles problemas con el robo/pérdida de tarjetas.

Los demás tipos de tarjetas se descartaron debido a la poca fiabilidad, seguridad e innovación que suscita una tecnología que va a quedarse desfasada en unos años y que es tan susceptible a efectos externos.

Gracias a esta primera elección se consiguió el equilibro perfecto entre seguridad, flexibilidad, fiabilidad y precio en este tipo de sistemas.

A continuación detallaremos la configuración de las tarjetas inteligentes que se ha diseñado para el sistema. Se han definido tres tarjetas inteligentes distintas cada una con su respectivo usuario, código PIN y permisos. La información se ha grabado con la siguiente estructura [13]:

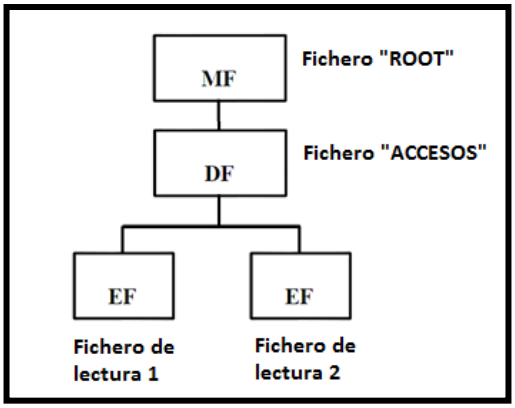


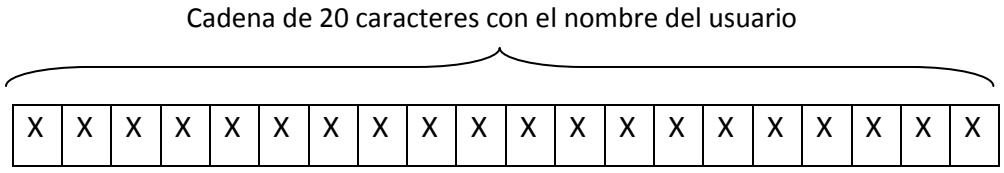
Figura 9 : Estructura de ficheros en las tarjetas inteligentes (tomado de [13]).

Para acceder a los ficheros EF debe pasarse una verificación de seguridad. Es decir, se debe introducir un código PIN y que éste coincida con la clave que tiene la tarjeta programada para dar permisos de lectura. En caso de no coincidir, dispone de hasta 10 intentos antes de que la tarjeta se bloquee.

Una vez que se ha superado la prueba de seguridad, ya se pueden leer los ficheros 1 y 2 de lectura. El primer fichero contiene simplemente el nombre del usuario, mientras que el segundo contiene los permisos de dicho usuario.

Fichero 1:

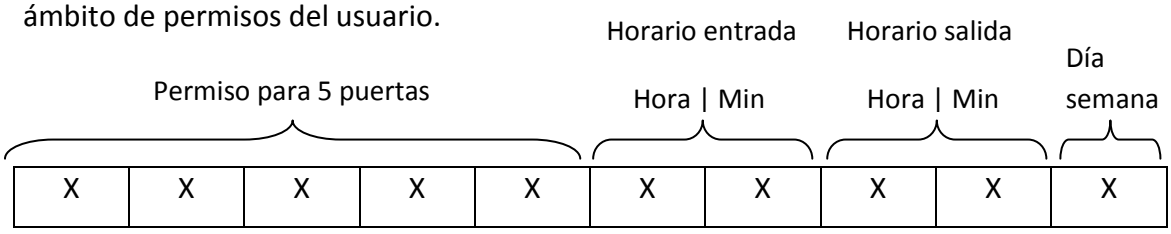
Contiene en una cadena de 20 bytes el nombre del usuario de la tarjeta. Por lo consiguiente, una lectura de dicho fichero tendría la siguiente estructura:



Seguida de los bytes de estado indicando que la lectura se ha realizado de forma correcta.

Fichero 2:

Este fichero contiene una cadena de 10 bytes cada uno con su propio significado en el ámbito de permisos del usuario.



Los cinco primeros bytes indican las puertas a las que se tiene acceso. Por ejemplo, si tuviese acceso a las puertas 1, 4, 5, 10 y 17, la definición de esos cinco primeros bytes sería:

0x01	0x04	0x05	0x0A	0x11

Los cuatro siguientes bytes indican la hora y minutos desde que tiene permitido el acceso, hasta cuando no lo tiene. Por ejemplo si entrase a las 9:30 y saliese a las 14:30 dichos bytes tendrían los siguientes valores:

0x09	0x1F	0x0E	0x1F

Y el último byte indica los días que se pueden pasar. Cada bit del byte al estar activo indica un día de la semana. Por ejemplo, si tuviese acceso de lunes a sábado a excepción del martes[9]:

0x3D	⇨		D	S	V	J	X	M	L
		X	0	1	1	1	1	0	1

Conocida la configuración establecida dentro de las tarjetas inteligentes, queda definir los tres usuarios para cada tarjeta.

<u>Usuario 1:</u>

- Código PIN: 12341234
- Nombre: Mario
- Permisos
  - Puertas: 1, 2, 3, 4, 5.
  - Horario: de 9:00 a 20:00
  - Días: de lunes a viernes

[9] *Nota: No se utiliza el bit más significativo del byte.*

- Fichero 1 = 0x4D, 0x61, 0x72, 0x69, 0x6F, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20.
- Fichero 2 = 0x01, 0x02, 0x03, 0x04, 0x05, 0x09, 0x00, 0x14, 0x00, 0x1F

Usuario 2:

- Código PIN: 56785678
- Nombre: Raul
- Permisos
    - Puertas: 5, 6, 7, 8, 9.
    - Horario: 24h
    - Días: de lunes a domingo
- Fichero 1 = 0x52, 0x61, 0x75, 0x6C, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20.
- Fichero 2 = 0x05, 0x06, 0x07, 0x08, 0x09, 0x00, 0x00, 0x17, 0x3B, 0x7F

Usuario 3:

- Código PIN: 90129012
- Nombre: Ana
- Permisos
    - Puertas: 9, 10, 11, 12, 13.
    - Horario: de 16:00 a 19:00
    - Días: de lunes, miércoles y viernes
- Fichero 1 = 0x41, 0x6E, 0x61, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20.
- Fichero 2 = 0x09, 0x0A, 0x0B, 0x0C, 0x0D, 0x10, 0x00, 0x13, 0x00, 0x15

Tal y como se comentaba en el capítulo 2 ("ESTADO DE LA TÉCNICA") existen múltiples combinaciones para los bytes de estado que se reciben después de realizar algún comando. La comprobación y significado de cada uno de ellos resulta una tarea muy

compleja, por lo que se decidió omitir dicha comprobación y evaluar dichos bytes de la siguiente forma:

- ✓ Si el primer byte de estado es SW1 = 0x9X ó SW1 = 0x6X sólo se contempla que la ejecución ha sido correcta y esperamos el SW2. Si recibimos cualquier otro SW1, lo consideramos parte de los datos que queremos recibir y no cómo byte de estado.

Una vez elegidas las tarjetas inteligentes como medio de identificación se llegó a la primera encrucijada del proyecto. Si todo nuestro sistema se basa en la identificación por tarjetas, ¿qué pasa si se te olvida la tarjeta? Para evitar la formulación de esa pregunta y conseguir un sistema más completo, se optó por incluir una segunda opción de usuario y contraseña como sistema de identificación.

Para que el sistema funcione con la segunda opción (usuario y contraseña) es totalmente necesario que se incluya una base de datos personalizada para cada usuario. Para ello la placa dispone de varias opciones entre ellas comunicación serie por Ethernet. Sin embargo, no fue hasta más adelante cuando se observó que no existe compatibilidad entre la tarjeta inteligente y la comunicación por Ethernet en esta placa de desarrollo (tal y como muestra la Tabla 6: "Tabla 6: Configuración Jumpers para Tarjeta Inteligente" del capítulo 3). Al colocar los "jumpers" para una configuración, se desconfigura la otra. Por lo tanto para posibilitar un sistema seguro a partir de usuario y contraseña se necesitaría acceder a una base de datos de forma inalámbrica, pero esto se detalla en el Capítulo 8 "*CONCULSIONES Y LÍNEAS FUTURAS*".

Se optó entonces por diseñar el sistema con dos modos de identificación, pero hay que destacar que no fue buscando más seguridad, sino buscando ofrecer más posibilidades. El primer y principal método es el de la tarjeta inteligente con código PIN, el cual sí consigue un nivel de seguridad elevado. Por el contrario, para el segundo método no se puede solicitar un usuario y contraseña personalizado para cada miembro ya que no se comunica con bases de datos, y evidentemente, no se va a introducir la propia base de datos en la placa de desarrollo. Por lo que el acceso sin tarjeta se definió con una contraseña general para todos los usuarios.

Como consecuencia de lo anterior, el sistema no consigue un nivel de seguridad adecuado. Aunque se presentan dos tipos de identificación, siendo uno de ellos muy seguro, la opción de contraseña sirve solo para ejemplarizar que también se pueden realizar métodos de identificación adicionales, ya que cualquier persona que conozca la contraseña puede acceder a la zona sin problemas, y por supuesto eso es una brecha de seguridad muy importante[10].

5.2 Medición del tiempo

El segundo aspecto a considerar en este tipo de sistemas es el tiempo. La correcta y fiable medición en tiempo real es una calve muy importante para ejercer como sistema de control de accesos. No sólo basta con limitar la zona a la que se quiere entrar por un método de identificación, también hay que limitarla en horario y días ya que no a todas las zonas se debería poder acceder cuando el usuario quiera.

El ejemplo más sencillo sería una oficina de un banco. Aunque quede reflejado quién entra y quién sale, no debería existir ninguna tarjeta habilitada para abrir el banco por la noche. Si se perdiera una tarjeta o se consiguiera duplicar alguna con dichos permisos, estaríamos ante un grave problema de seguridad.

Para cumplir ese requisito, un microprocesador dispone de varios periféricos capaces de llevar a cabo esa función. Sin embargo si se conocen bien todos, es fácil decantarse por uno de ellos dejando a los demás de lado por enormes inconvenientes que introducen en el sistema.

Se habla de la gran diferencia de funcionamiento entre un temporizador (Timer) de uso general y el RTC. El Timer de uso general dispone de una serie de configuraciones para llevar el contador de tiempo al periodo deseado, tal y como se hace en el RTC. Sin embargo, dicho contador está sujeto al reloj general del sistema con los problemas que ello acarrea.

[10] *Nota: Los permisos para acceder sin tarjeta se limitan a comprobar la hora en la que se está accediendo. No se contemplan más comprobaciones por no tratarse de un método seguro y su complicación sería ilógica. El horario permitido será de 16:00 a 19:00.*

Por ejemplo, si se configura el sistema con interrupciones, el Timer de uso general se verá influenciado por el tiempo que tarde en volver al programa principal o a su propia interrupción, dejando entonces desfasado su cuenta respecto al tiempo real.

Por el contrario, el RTC dispone de un reloj independiente al sistema que no se para sin importar la instrucción que se esté ejecutando. Además de que se apoya en los sistemas de backup siendo capaz de mantener el contador activo sin fuente de alimentación externa, simplemente con la batería.

Por todos estos motivos, incluyendo lo intuitivo que resulta utilizar el "Real-Time Clock" para aplicaciones en "tiempo real" se escogió este periférico para cumplir las funciones de horario que precisa el sistema.

Para nuestra sorpresa, tal y como comentábamos en el capítulo 3, apartado 3.2.1 "*RTC (Real Time Clock)*", el RTC no dispone de registros para mantener la fecha, tan sólo para mantener la hora. Esto provocó que se tuvieran que diseñar una serie de funciones con el propósito de definir un calendario interno dentro del sistema.

Además se advirtió la necesidad de incluir la posibilidad de modificar fecha y hora en el sistema. Esta necesidad nace de la probabilidad del sistema de quedarse sin batería y perder fecha y hora con cualquier corte eléctrico[11]. Por otra parte, existen cambios nacionales de hora que de no incluirse esta funcionalidad, obligaría a la empresa/institución a desmontar el soporte del dispositivo y reprogramarlo mediante un ordenador.

Por el contrario, esta funcionalidad añade una brecha de seguridad muy importante. Cualquier usuario que sea capaz de acceder a dichas modificaciones no dependerá de sus permisos personales de fecha y hora para acceder a la zona en cuestión.

Este problema se soluciona añadiendo bloques de seguridad para impedir el acceso a estas funciones a usuarios no autorizados. Estos bloques adicionales no se contemplan para el desarrollo ya que no es un diseño definitivo sino un prototipo, pero sí se ha creído conveniente enfatizar en este aspecto para dejar claros los conceptos de que

[11] *Nota: La batería de nuestra placa de desarrollo se agotó mientras realizábamos las pruebas del funcionamiento del sistema. Aún así, se pudo comprobar que los registros de backup funcionaban correctamente.*

reduce el nivel de seguridad del dispositivo pero añade una función imprescindible en este tipo de sistemas.

Por último añadir que esta funcionalidad además facilita el proceso de pruebas en el desarrollo ya que no es necesario estar reprogramando el microcontrolador para cambiar estos parámetros y comprobar que los usuarios dejan o no de tener acceso.

5.3 Control de energía

Una vez tratados los dos aspectos más importantes en un sistema de control de accesos: método de identificación y medida en tiempo real, se decidió hacer más completo el sistema añadiendo más funcionalidades.

Independientemente de los objetivos que vaya a tener un sistema, su consumo energético es siempre uno de los factores a estudiar, ya sea por el bien económico de la empresa/institución o bien, por el bien del medioambiente.

Estamos hablando de intentar ser eficientes en la relación productividad y gasto eléctrico. Conseguir esa eficiencia conlleva enormes ventajas que pueden convertir un sistema mediocre en un gran sistema.

Incluir un modo de bajo consumo se convirtió entonces en un objetivo a cumplir en el presente proyecto. Para ello bastaba con configurar el microcontrolador STM32F107VC en uno de los tres modos de bajo consumo que presenta (véase en el capítulo 3, apartado 3.2.3 “*Modos de bajo consumo*”).

De entre los tres modos de bajo consumo hay que elegir el que más se adapte a nuestras condiciones de trabajo y no tiene porqué ser el elegido en este proyecto el mejor para todas las situaciones.

Para el desarrollo del sistema, se escogió el Stop Mode por ser el que mejor relación “ahorro de energía - tiempo en despertar” conseguía. El Sleep Mode aunque su tiempo de recuperación es más corto, su ahorro energético nos parece insuficiente. Sin embargo, el Standby Mode consigue un ahorro energético mayor, pero en contra, su

tiempo de recuperación es muy lento. También fue desestimado porque detiene los periféricos que se pensaba utilizar para despertar al sistema.

Además de que el objetivo del diseño del presente proyecto no es conseguir que sea lo más rápido posible ni que ahorre la mayor cantidad de energía, sino mostrar todas las posibilidades que presenta para que cada uno adapte su configuración a sus condiciones de trabajo.

5.4 Diagrama de bloques

Una vez conocidas todas las características que buscamos en el presente proyecto, procederemos a incluir un pequeño diagrama de bloques con el diseño general del sistema para en el siguiente capítulo comenzar a desarrollarlo.

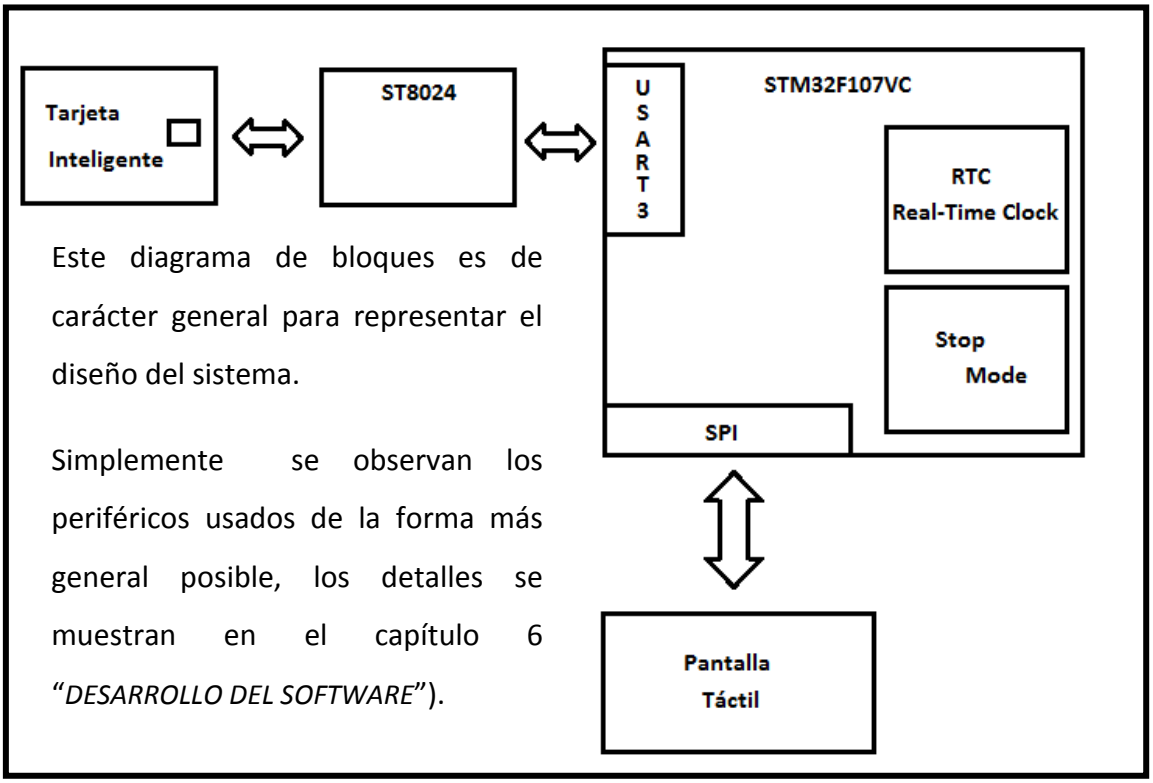


Figura 10: Diagrama de Bloques del Sistema General (tomado de [13]).

6 DESARROLLO DEL SOFTWARE

En este capítulo nos centraremos en explicar en detalle el desarrollo del sistema. Para ello, dividiremos el programa por bloques indicando cómo ha sido el desarrollo particular de cada uno, los problemas que se han encontrado y cuál es la solución final.

Antes de nada se incluye un diagrama de flujo indicando de forma muy genérica cuál es el funcionamiento y proceso de nuestro sistema:

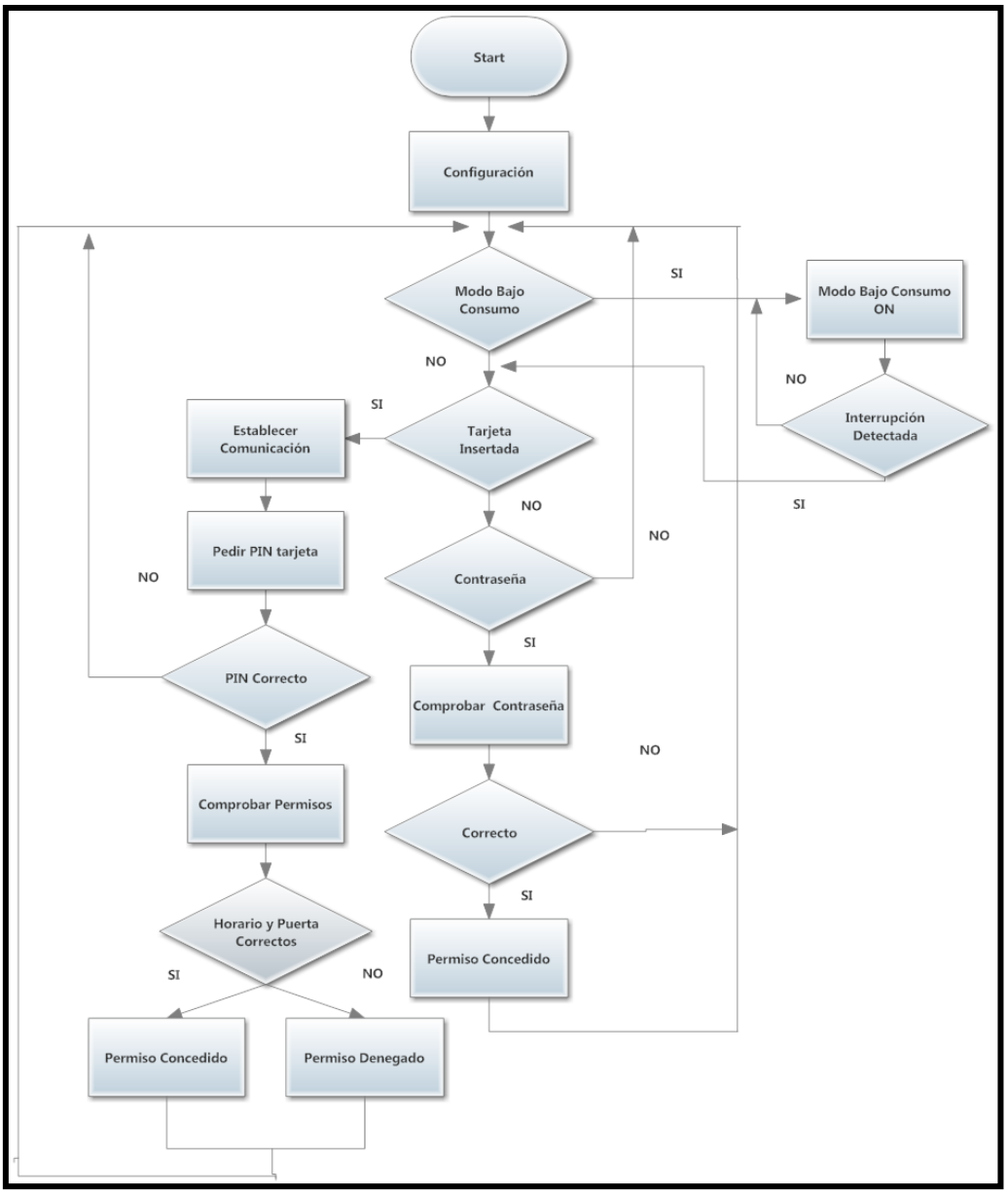


Figura 11: Diagrama de Flujo. Funcionamiento Global (tomado de [13]).

El procedimiento para explicar este capítulo será inicialmente una breve introducción de qué es lo que hace ese bloque y cómo se ha desarrollado. Se incluirá después un diagrama de flujos en relación a este principal para simplificar su comprensión y agilizar su lectura.

Para ello se ha dividido el programa en seis bloques:

- ✓ Configuración: Aquí se detallan todas las funciones y parámetros que se han introducido/modificado para conseguir que los periféricos funcionen como se precisa.
- ✓ Main: Programa principal dónde se lleva a cabo todas las ejecuciones. Cabe destacar que es un bucle sin fin, ya que el sistema no debe quedarse parado en ningún punto.
- ✓ Tarjeta Inteligente: Todas las funciones que se han utilizado en relación con la tarjeta inteligente.
- ✓ Pantalla Táctil: Funciones definidas para interactuar con la pantalla.
- ✓ Interrupciones: Procesos que se ejecutan fuera del programa principal
- ✓ Funciones de carácter general.

Por último añadir que se han omitido las funciones creadas para mostrar mensajes en la pantalla ya que suponen una explicación complicada y no añaden funcionalidad, simplemente sirven para embellecer el sistema y hacerlo más atractivo. Por otra parte, se ha incluido como anexo el código completo del programa (véase Apartado 9.3 "*Anexo 3: Código Total del diseño*"), y además, se mostrarán en el capítulo 7, apartado 7.2 "*Resultado Final*", fotografías del sistema para mostrar la interfaz gráfica.

6.1 Configuración

6.1.1 Introducción

Este es el primer bloque centrado en explicar cómo se han definido las características generales del microprocesador que consiguen que los periféricos se comporten de la forma requerida.

Para ello la explicación se dividirá por bloques funcionales siendo los siguientes bloques los definidos:

- ✓ RCC_Configuration: Registros que configuran los relojes que vamos a utilizar.
- ✓ USART3_Configuration: Configuración de la USART3, la cual utilizamos para comunicarnos con las tarjetas inteligentes.
- ✓ GPIO_Configuration: Aquí se detalla la configuración de los GPIOs que se van a utilizar.
- ✓ RTC_Configuration: Configuración del reloj en tiempo real.
- ✓ NVIC_Configuration: Describe cómo el sistema va a procesar las interrupciones definidas.
- ✓ EXTI_Configuration: Se trata de las interrupciones externas que se contemplan.

Es importante destacar que el desarrollo de esta parte aunque ha sido totalmente propia y una de las más importantes, ha sido relativamente la parte menos costosa de todas. Esto se debe a que en el Reference Manual del microcontrolador [9] vienen descritas todas las opciones y configuraciones que se deben llevar a cabo para conseguir que el sistema se comporte de una forma u otra.

6.1.2 Descripción funcional

6.1.2.1 RCC_Configuration

En esta función se configuran los relojes del sistema. Sólo hay que destacar dos características importantes ya que no hay que cumplir ningún requisito de velocidad en ninguno de los periféricos utilizados.

Dichas características son que el APB2 (bus donde se conectan parte de los periféricos que vamos a utilizar) no puede ir a más de 72 MHz, por lo que configuraremos el reloj del sistema a esa velocidad y sin modificarle llega

directamente al APB2. La otra característica es que el APB1 (en el cual se conecta la otra parte de los periféricos utilizados) no puede ir a más de 36 MHz. Para ver dónde se conecta cada periférico, ver la "*Figura 6: Diagrama de Bloques para la comunicación con los periféricos*" en el capítulo 3.

Para comprender cómo y el porqué se han configurado así los relojes es necesario detenerse en el siguiente esquema que muestra cuál es la distribución de los relojes en el sistema.

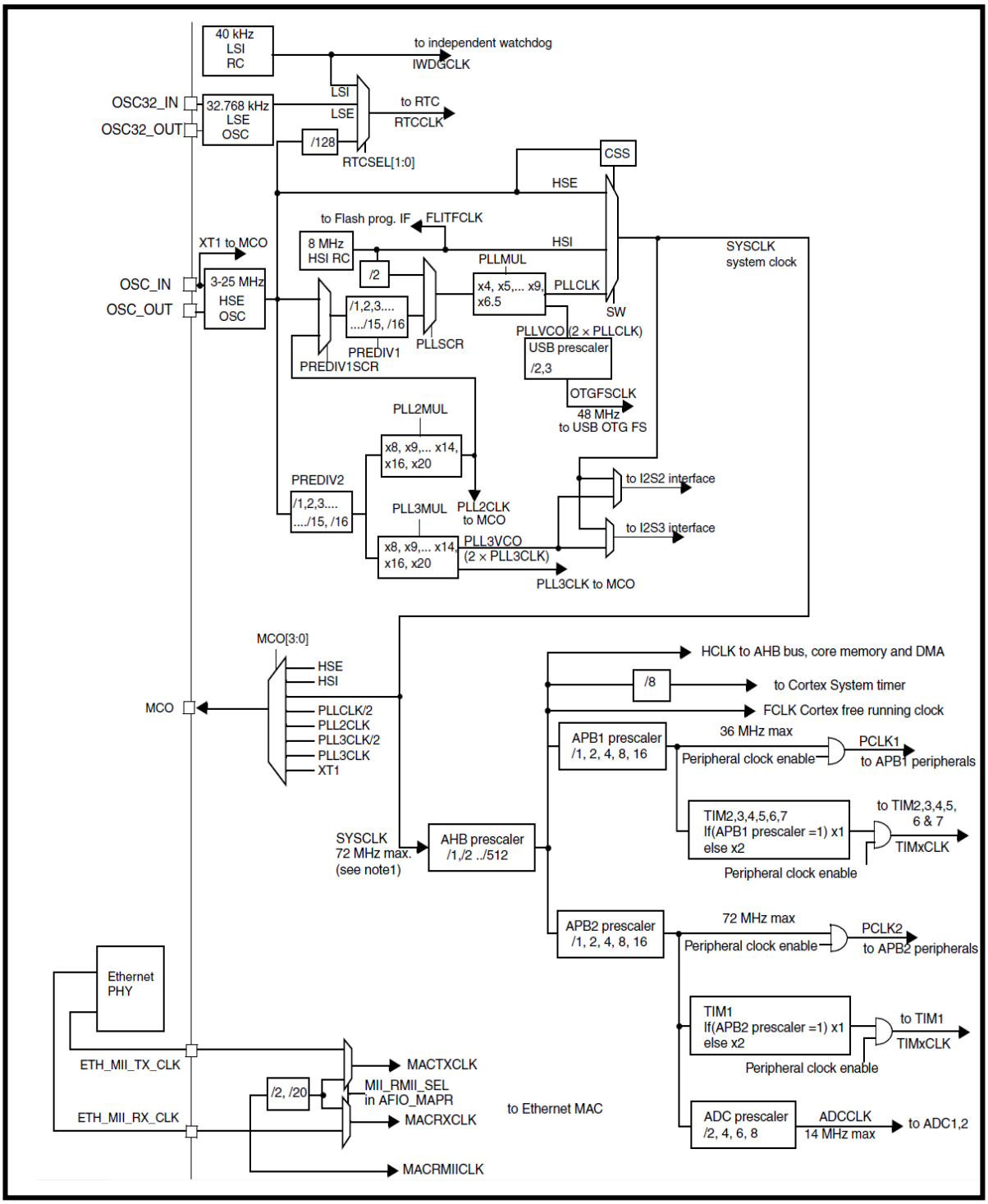
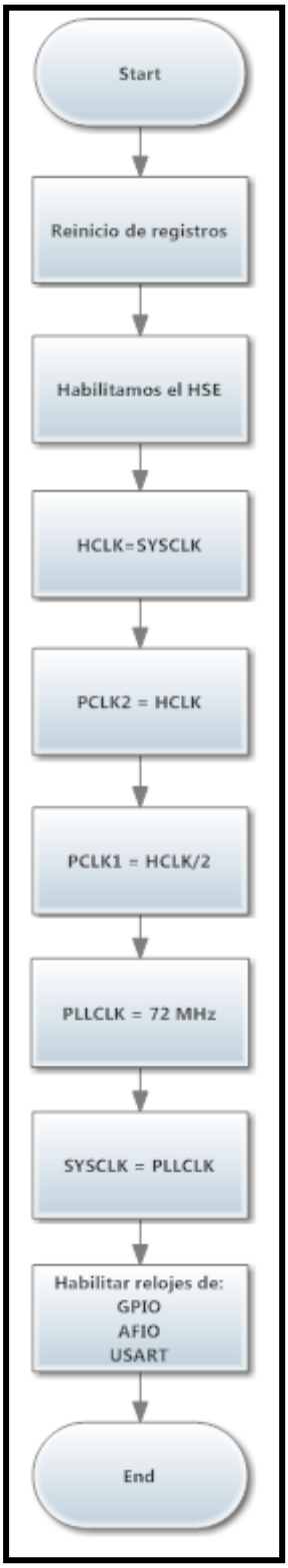


Figura 12: Esquema de la Configuración del RCC (tomado de [9]).

Por defecto el HSE se configura a 8 MHz, por lo que utilizamos esa configuración para definir el resto de nuestro sistema.

A continuación se muestra en un diagrama de bloques las operaciones realizadas en esta función de configuración.



Configuración por defecto de los registros mediante su reinicio

.

Utilización del SYSCLK sin alterar para el AHB

Utilización del AHB sin alterar para el APB2

Ya que la frecuencia máxima del APB1 es la mitad del APB2, se divide entre dos.

Configuración del PLLCLK para conseguir los 72 MHz deseados, multiplicando por 9 el HSE configurado con 8 MHz

Utilización del PLLCLK como SYSCLK

Habilitación de los relojes del APB1 y APB2 para poder utilizar los periféricos requeridos

Figura 13: Diagrama de Flujo de RCC_Configuration (tomado de [13]).

Por último mostrar un pequeño resumen del camino que se ha seguido para conseguir esta configuración.

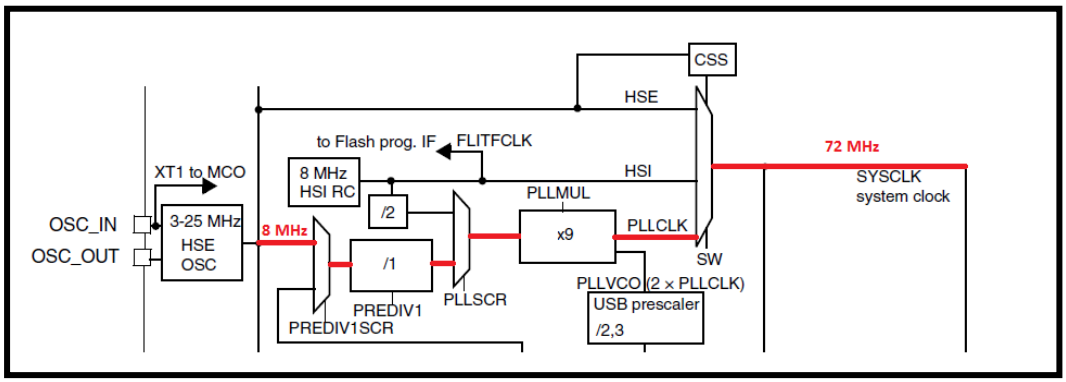


Figura 14: Procedimiento de Configuración para el RCC (tomado de [13]).

6.1.2.2 USART3_Configuration

El sistema dispone de hasta 5 puertos USART, pero no todos ellos permiten utilizar el modo de comunicación necesario para una tarjeta inteligente.

USART modes	USART1	USART2	USART3	UART4	UART5
Asynchronous mode	X	X	X	X	X
Hardware Flow Control	X	X	X	NA	NA
Multibuffer Communication (DMA)	X	X	X	X	NA
Multiprocessor Communication	X	X	X	X	X
Synchronous	X	X	X	NA	NA
Smartcard	X	X	X	NA	NA
Half-Duplex (Single-Wire mode)	X	X	X	X	X
IrDA	X	X	X	X	X
LIN	X	X	X	X	X

1. X = supported; NA = not applicable.

Tabla 8: Periféricos posibles para cada USART (tomado de [9]).

Gracias a esta tabla se comprueba que sólo se pueden utilizar los tres primeros puertos. Para el desarrollo, se decidió la USART3 por las facilidades que proporciona para configurarse en el modo Smartcard. Dichas facilidades

así como su conexionado se explican en el siguiente apartado "GPIO_Configuration".

Sin embargo, una vez elegida la USART3 para comunicarse con la tarjeta, es obligatorio cumplir una serie de requisitos en su configuración para hacer posible el correcto entendimiento entre la información recibida/transmitida por la tarjeta y la transmitida/recibida por el micro.

Este fue uno de los mayores problemas encontrados en el desarrollo del sistema debido a la escasa información que había respecto al tema. Como ya se comentó en la metodología (capítulo 4 "*METODOLOGÍA Y REQUISITOS*") los manuales y ejemplos para esta placa aparte de ser bastante reducidos, estuvieron disponibles mucho después de lo previsto.

Consecuentemente, la primera vez que se configuro la USART fue con las características de otra placa y su resultado fue pésimo. Aunque se conseguía detectar la tarjeta (ya que no es necesario tener bien configurada la USART porque depende de los GPIOs) no se recibía nada del ATR.

Esto provocó buscar una solución fuera del software y meterse de lleno en hardware. En un laboratorio de GUTI (Grupo Universitario de Tecnologías de Identificación) gracias a Sandra Cid y a un osciloscopio digital se pudo observar qué estaba fallando en la comunicación.

Los primeros fallos detectados fueron que la comunicación no se realizaba a la misma velocidad, por lo que era necesario modificar el baudrate de USART.

Esta solución fue más sencilla cuando se consiguió localizar el manual dónde se describía cuales eran las características de la comunicación [14].

- ✓ Stop Bits =1,5
- ✓ Word length = 8 + Paridad = 9 bits
- ✓ Baud Rate =9600

Sin embargo, la velocidad del reloj seguía estando mal configurada e imposibilitaba una correcta comunicación entre la tarjeta y el microprocesador.

Midiendo la velocidad del reloj que suministrábamos a la tarjeta en el pin correspondiente (PD10) se consiguió detectar que esa velocidad era muy alta para permitir la comunicación. Después de reducirla varias veces conseguimos recibir el ATR de forma correcta. Lo que significaba que la USART ya estaba bien configurada.

El baremo para conocer de que orden tenía que ser dicha velocidad se obtuvo del manual de tarjetas donde se indicaba que cualquier tarjeta del tipo A se comunica entre 1 y 5 MHz [3].

Cabe destacar que la comunicación en ese entonces se estaba realizando por interrupciones. Es decir, el pin de recepción/transmisión estaba configurado como una interrupción externa de tal forma que cada vez que el microprocesador detectase algo en el registro de datos, la interrupción saltaba. Esto acarreó varios problemas que se describirán más adelante y que obligó a cambiar de interrupciones a esperas activas para conseguir la completa y definitiva comunicación con la tarjeta.

Hay ciertas características de la USART que se pueden modificar, sin embargo, su configuración por defecto sirve para una correcta funcionalidad y no se ha requerido modificar dichas características.

A continuación se muestra el diagrama de bloques de esta función con las ejecuciones más importantes de la configuración.

➔ Para conseguir esta velocidad es necesario dividir la del APB1 por 8:

APB1 Clock (=36 MHz) / 8 = 4,5 MHz (= USART 3 Clock)

➔ Se habilita tanto la recepción como la transmisión ya que se trata de una comunicación semi dúplex y ambas van a ir por el mismo canal.

Figura 15: Diagrama de Flujo para USART3_Configuration (tomado de [13]).

6.1.2.3 GPIO_Configuration

En este apartado hay que tener muy presente las conexiones que se comentan en el capítulo 3, "*Tabla 7: Definición de GPIOs para establecer comunicación con una tarjeta inteligente*".

La configuración de estos pines se ha realizado de forma directa sin utilizar ninguna función, es decir, se han modificado directamente los registros.

En primer lugar hay que tener claro que hay que remapear los pines para conseguir conectar la USART3 al chip ST8024. Este redireccionamiento se hace mediante los registros del AFIO y según la siguiente tabla:

Alternate function	USART3_REMAP[1:0] = "00" (no remap)	USART3_REMAP[1:0] = "01" (partial remap) (1)	USART3_REMAP[1:0] = "11" (full remap) (2)
USART3_TX	PB10	PC10	PD8
USART3_RX	PB11	PC11	PD9
USART3_CK	PB12	PC12	PD10
USART3_CTS	PB13		PD11
USART3_RTS	PB14		PD12

Tabla 9: Remapeo de la USART3 según los registros AFIO (tomado de [9]).

Se observa que si se remapea tal y como dice la tabla, se consiguen conectar los pines de la misma forma que nos exige el chip ST8024.

En un principio se creyó que con utilizar la USART3 era suficiente ya que se desconocía la posibilidad de redireccionamiento. Al comprobar que la comunicación no era posible se estudió el caso y se verificó que era necesario modificar los registros del AFIO para conectar correctamente la USART3 a los pines donde si iba a establecer la comunicación.

Una vez hecho esto, basta con configurar cada pin como se requiere:

- ✓ PD8: Alternate Function Push-Pull ya que se va a utilizar tanto de entrada como de salida, al tratarse de una comunicación semi dúplex.
- ✓ PD9: Se utiliza para introducir el Reset en frío Output Push-Pull
- ✓ PD10: Utilizado como reloj, se configura como Output Push-Pull.
- ✓ PC0: Será quién alimente la placa así que también como Output Push-Pull

- ✓ PE7: Es el pin encargado de comprobar si hay tarjeta introducida, por lo tanto tiene que ser Input Floating[12].
- ✓ PE14: Este pin se utiliza para poder realizar el Reset en frío y pedir el envío del ATR a la tarjeta. Por lo tanto tiene que ser Output Push-Pull

Si se observa el manual que nos facilita ST para el correcto desarrollo de un programa utilizando una tarjeta inteligente encontramos la siguiente tabla [14].

Smartcard pins	STM3210B-EVAL	STM3210E-EVAL	STM3210C-EVAL	Function
C3: CLK	USART3 CK: PB12	USART3 CK: PB12	USART3 CK: PD10	Smartcard clock: alternate function push-pull
C7: IO	USART3_TX: PB10	USART3_TX: PB10	USART3_TX: PD08	IO serial data: alternate function open drain
C2: RST	PB.11	PB.11	PD.09	Reset to card: output push-pull
C1: V_{CC}	PE.07	PC.06	PD.07	Supply voltage: output push-pull
OFF	PE.14	PC.07	PE.07	Smartcard detect: input floating
3/5V	PD.11	PB.00	PC.00	3 V or 5 V: output push-pull

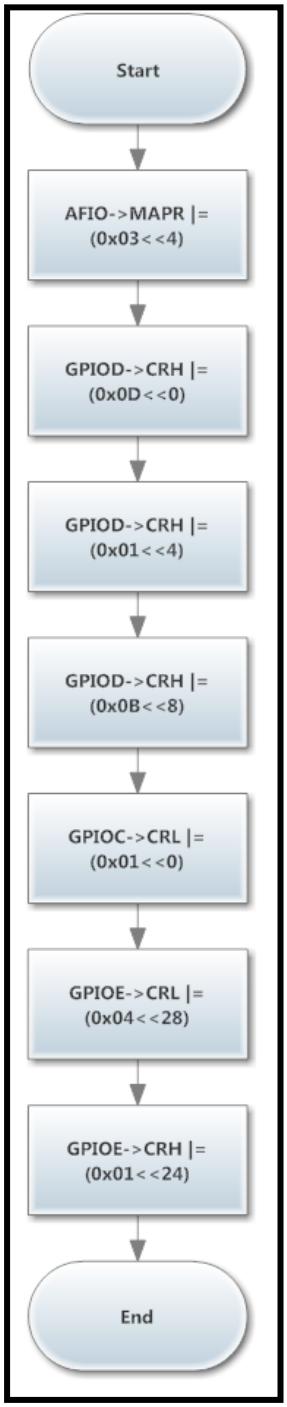
Tabla 10: Configuración de GPIOs para SmartCard (tomado de [14]).

En el desarrollo del sistema, se comprobó que esta configuración de los pines era innecesaria. Se modificó para ver su nuevo comportamiento y éste no se vio modificado en ningún aspecto, por lo que se decidió seguir trabajando de la misma forma ya que se estaban consiguiendo grandes resultados.

Si se realiza todo como se ha descrito anteriormente, ya es posible establecer una comunicación correcta entre la tarjeta inteligente y el microcontrolador STM32F107VC.

Para acabar con esta función, incluir un diagrama de bloques indicando que operaciones exactas se realizan.

[12] *Nota: Hay que destacar que dicho pin no comprueba la conexión eléctrica entre los pines del chip y los pines de la tarjeta, sino que comprueba si hay algo físico introducido en el lector, ya sea una tarjeta o un trozo de cartón con la misma forma.*



➔ Se remapea la USART3 como “11”

➔ PD8: Alternate Function Push-Pull

➔ PD9: Output Push-Pull

➔ PD10: Output Push-Pull

➔ PC0: Output Push-Pull

➔ PE7: Input Floating

➔ PE14: Output Push-Pull

Los pines PE14 y PC0 deben estar activados para permitir enviar el Reset en frío y alimentar la tarjeta respectivamente.

Figura 16: Diagrama de Flujo de GPIO_Configuration (tomado de [13]).

Se decidió configurar la mayoría como Push-Pull para evitar posibles interferencias por corrientes parásitas y asegurar una buena rapidez de transmisión de información.

La posibilidad de Open-drain se descartó ya que se suele aconsejar su uso cuando se realizan múltiples conexiones. Aunque esta decisión tampoco fue objeto de mucho estudio.

6.1.2.4 RTC_Configuration

Una vez que se ha configurado todo lo necesario para permitir la comunicación entre una tarjeta inteligente y nuestra placa STM3210C-EVAL se procede a configurar los otros periféricos que se utilizan en el desarrollo.

Este caso se centrará en explicar la configuración del RTC, utilizado para mantener la hora en el sistema.

Al igual que en el caso de la configuración del RCC, está parte no ha supuesto ningún problema al tratarse de una funcionalidad muy utilizada en los microprocesadores donde la documentación está muy bien explicada ya sea para este micro o cualquier otro.

Lo primero que hay que conocer para poder realizar una buena configuración es el esquema de conexionado para el RTC. Para ello hay que centrarse en una parte de la "Figura 7: Diagrama de Bloques del RTC":

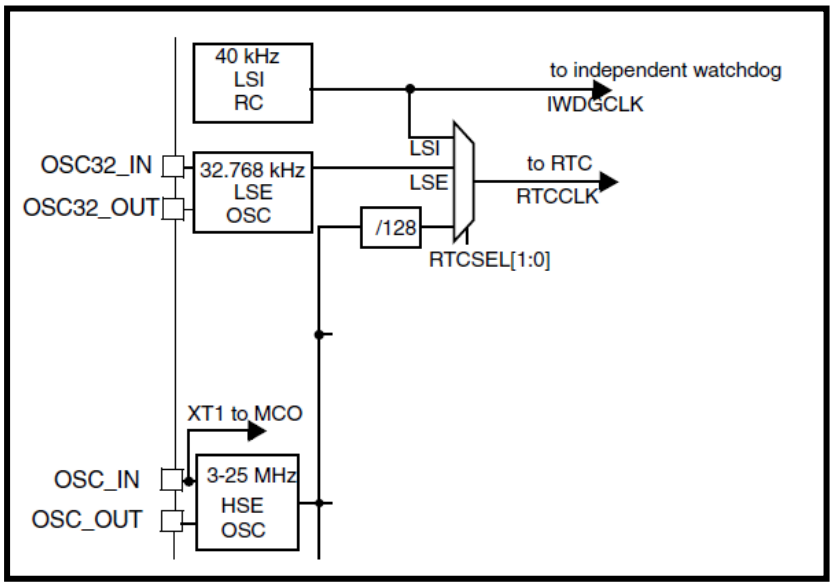


Figura 17: Esquema para la Configuración del RTC (tomado de [9]).

Para definir el reloj del RTC (RTCCLK) se utiliza como fuente el LSE de 32.768 kHz. Esta decisión no se tomó por ningún aspecto particular.

Una vez que se selecciona la fuente del reloj, hay que configurarlo para lo que se quiera. En nuestro caso se pretende utilizar este reloj para medir el tiempo, por lo que lo óptimo será configurar su contador para que cuente segundos.

Para ello, basta con programar el pre-escalado (prescaler) de la forma que hablábamos en el Capítulo 3 "*PLATAFORMA DE DESARROLLO*" Apartado 3.2.2 "*RTC (Real Time Clock)*".

$$f_{TR\_CLK} = \frac{f_{RTCCLK}}{(PRL[19:0] + 1)}$$

Al programar el prescaler con un valor de 32767 (PRL[19:0]=0x7FFF), se consigue que la frecuencia del contador sea un segundo.

Es importante enfatizar que esta función del sistema es crucial para un correcto desempeño en las funciones de un sistema de control de accesos. No debe haber ninguna instrucción que intervenga en dicho contador o de lo contrario se puede producir un desfase entre la hora y el valor del contador provocando malas medidas.

Aunque el contador del RTC es independiente al sistema y por tanto siempre se lleva a cabo, se optó por definir una interrupción cada vez que dicho contador se incrementase. Esto se comentará en el siguiente apartado "EXTI_Configuration, NVIC_Configuration y Modo bajo consumo".

Antes de incluir el diagrama de bloques perteneciente a esta función es necesario recalcar dos detalles muy importantes.

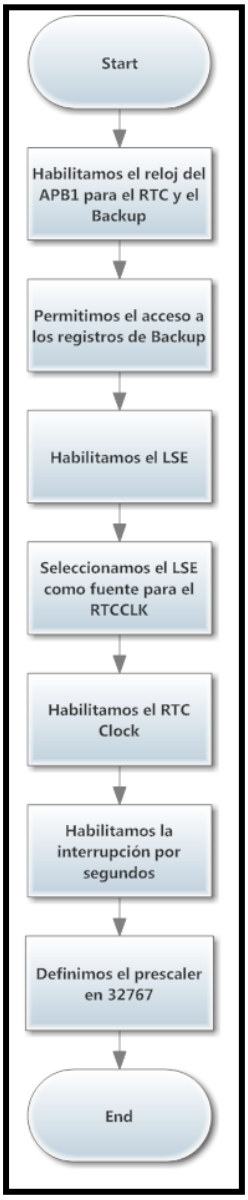
El primero de ellos es la obligatoriedad de habilitar el reloj para los registros de Backup así como permitir su acceso. Este detalle es muy importante ya

que los registros del RTC se sustentan en registros de Backup para guardar la información aunque la fuente de alimentación se desconecte.

El segundo de ellos es un gran inconveniente existente en este micro. No dispone de registros para mantener la fecha además de la hora. Es decir, sí permite utilizar el RTC para controlar la hora, pero no dispone de ningún mecanismo para hacer lo análogo con la fecha (día, mes y año).

Debido a la obligación existente en este tipo de sistemas de controlar además de la hora, la fecha, se tuvieron que definir una serie de funciones que se detallarán en el apartado 6.6 "*Funciones de carácter general*" pero que en resumidas cuentas definen un calendario completo.

Por último, añadimos el diagrama de bloques referente a esta función de configuración:



Con el fin de no complicar el diagrama, se ha decidido omitir ciertas instrucciones de espera necesarias para la correcta configuración del RTC. Dichos bucles de espera se detallan a continuación en los puntos correspondientes y explicando su utilidad:

➔ Antes de seleccionarle como fuente, es necesario esperar a que salte el aviso de que se ha habilitado correctamente

➔ Al activarse el RTC se produce una sincronización de registros, por lo que es necesario esperar hasta que esta finalice para habilitar la interrupción por segundos. A partir de aquí, cada vez que se escriba algo en el registro del RTC hay que incluir un bucle de espera el cual indica que se han realizado todas las modificaciones en los registros de forma completa y no hay ninguna escritura pendiente.

Figura 18: Diagrama de Flujo para RTC_Configuration (tomado de [13]).

6.1.2.5 EXTI_Configuration, NVIC_Configuration y Modo bajo consumo

Estas tres funciones se han unificado en un mismo apartado por la simplicidad de cada una de ellas. Primero nos centraremos en las interrupciones externas ya que son muy sencillas y breves de explicar.

Se han configurado los botones de Wakeup y Tamper para que trabajen como interrupciones externas. Esto se pudo hacer de dos formas distintas, mediante la configuración de sus registros correspondientes tal y como hicimos para los GPIOs de la tarjeta o mediante la llamada de una función que los configura automáticamente.

Se decidió hacer de la segunda forma por su sencillez y rapidez ya que la otra forma de configurarse hubiera sido más costosa y no hubiera aportado nada adicional por tratarse del mismo procedimiento que para los anteriores GPIOs. Las líneas de interrupción externa para cada uno son:

- ✓ Botón WAKEUP: Línea 0 de la EXTI → EXTI0_IRQHandler
- ✓ Botón TAMPER: Línea 13 de la EXTI → EXTI15_10_IRQHandler

Simplemente hay que recalcar de esta función, que la prioridad por defecto de dicha interrupción es la más baja y que según el botón, se configura como flanco de bajada (WakeUp) o de subida (Tamper).

Además, la función de estos botones no es principal en el sistema. Ambos sirven para "despertar" al sistema del modo de bajo consumo.

Sin embargo, para con el botón Tamper se incluye una funcionalidad extra que sirve tanto para el cambio de hora como para el cambio de fecha en el sistema. Pero eso se detallará en el apartado 6.6 "*Funciones de carácter general*", aquí solo nos limitaremos a decir que está configurado como interrupción externa.

Lo segundo en explicar será del NVIC, encargado de procesar las interrupciones, tanto externas como internas. En este caso, solo se define la

interrupción del reloj RTC ya que, como se comentó anteriormente, las de los botones Tamper y Wakeup se hacen automáticamente con una función.

En esta configuración hay que enfatizar un aspecto importante, la prioridad. Tal y como ya habíamos mencionado, el mantenimiento de la hora es prioritaria, por lo que su representación debe llevarse a cabo siempre.

Para conseguir esta necesidad, es necesario aumentar la prioridad de esta interrupción frente a las demás. Por lo que en la definición de la interrupción del RTC en el NVIC, se define la prioridad como “0x01” a diferencia del resto que estaba como “0x00”.

Por último solo queda mostrar la configuración necesaria para que el sistema permita el uso de modos de bajo consumo.

Para este microprocesador no se necesita activar/desactivar ningún registro ya que siempre está permitido la utilización de los modos de bajo consumo. Para entrar en ellos sólo se necesita llamar a la función del tipo de bajo consumo que se quiera utilizar, especificando el modo de salida del mismo.

En este caso, se utiliza el Stop Mode, el cual tiene la posibilidad de reducir el consumo de un regulador interno de tensión. Por lo que en la función, además de indicar si se quiere salir por interrupción o por evento, también se debe indicar si dicho regulador se mantiene encendido o si por el contrario pasa también a bajo consumo.

6.2 Programa Principal (main)

6.2.1 Introducción

Una vez configurados todos los periféricos del sistema, esta sección se dedicará a explicar el núcleo del programa.

El núcleo del programa es donde se van a llevar a cabo todas las actividades principales del sistema de control de accesos y donde se ejecutan todas las funciones, ya sea directa o indirectamente.

Antes de seguir hay que destacar un detalle muy importante. Se trata de la subdivisión existente dentro del la función principal del sistema (main). Se establece dicha subdivisión para separar las operaciones de inicialización, de las operaciones continuas que se ejecutan en un bucle infinito que está llamado a ser la verdadera interfaz con el usuario.

Por lo tanto en el siguiente apartado de "*Descripción funcional"* la explicación se dividirá en dos subapartados:

- ✓ Funciones iniciales: Esta parte del código solo se ejecuta una vez y al principio. Sirve para inicializar todos los parámetros de configuración.
- ✓ Bucle del programa: Se define como un bucle infinito donde se estarán ejecutando todas las actividades del sistema, según vaya marcando la interactuación con el usuario.

6.2.2 Descripción funcional

6.2.2.1 Funciones iniciales

Las actividades específicas que se llevan a cabo al principio están definidas en su mayoría en la parte de configuración. Sin embargo, es importante resaltar el orden de ejecución así como ciertas funciones que no se han desarrollado personalmente y son también importantes para la correcta inicialización de todos los periféricos.

Para simplificar su explicación se incluye un diagrama de flujo completo que resume las operaciones que se ejecutan hasta llegar al bucle del programa.

Función que inicializa la placa de desarrollo. Tomada de las librerías

Configura el Systick con un pulso cada 10 ms. (Se trata de los pulsos que utiliza la función de Delay)

Esta configuración es realizada por una función de la librería tal y como se comentó anteriormente

Comprobación para verificar que la inicialización de los periféricos ha sido correcta. De no ser así, el programa no puede continuar y fuerza su reinicio.

Las funciones de configuración se ejecutan en este momento. ←

Estas dos funciones se añaden para permitir al usuario definir la puerta y la contraseña que tendrá esa placa en relación al sistema completo. Obviamente esto conlleva unos problemas de seguridad muy importantes, pero no se ha llevado a cabo por esos motivos sino por otros que se explicarán a continuación

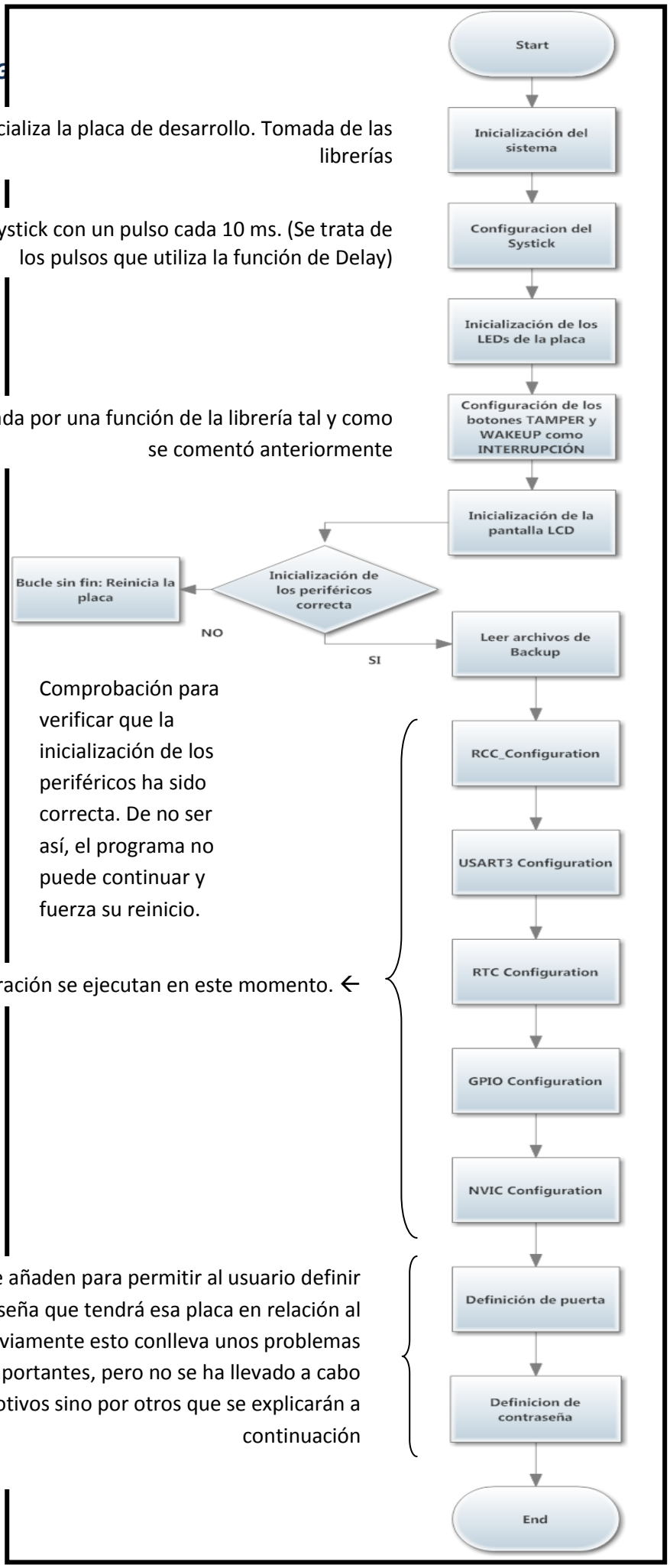


Figura 19: Diagrama de Flujo Anterior al Bucle del Programa (tomado de [13]).

Se han incluido dos actividades sin relación a la configuración del sistema. La primera de ellas es una lectura de los registros de Backup como consecuencia de la incapacidad para definir la fecha con los registros del RTC. Y la segunda son dos funciones que permiten introducir puerta y contraseña del sistema.

El principal motivo para permitir definir tanto la puerta como la contraseña del sistema, fue facilitar al autor tanto el desarrollo como las pruebas del sistema. La fecha y el número de puerta son aspectos cruciales que deben ser probados de todas las formas posibles, por lo que una constante reprogramación para cambiar dichos aspectos produciría enormes pérdidas de tiempo.

Como ya se comentó, esto introduce una brecha de seguridad muy importante, sin embargo, esta funcionalidad podría tener más sentido si se lleva a cabo un diseño que asegure la estanqueidad e inaccesibilidad a dichas funciones por parte de usuarios no autorizados como se mencionó en el capítulo 5 "*DISEÑO DEL SISTEMA*".

Por último añadir que las funciones que inicializan la placa y comprueban su correcta inicialización han sido la única parte de todo el desarrollo que no ha sido elaborada personalmente, sino que han sido tomadas de las librerías.

6.2.2.2 Bucle del programa

Este es el apartado principal en el desarrollo del sistema. Anteriormente nos hemos limitado a configurar la STM3210C-EVAL para posibilitar el uso de los periféricos deseados.

Sin embargo, es importante matizar que aquí no se explicará el detalle de cada una de las funciones empleadas y sí el funcionamiento en profundidad del sistema. Es decir, se mencionará el objetivo de las funciones empleadas para detallar el funcionamiento global del sistema, y no en explicar cómo se

han elaborado cada una de las funciones. Esa explicación se retrasará hasta su correspondiente apartado según la función que sea.

Antes de incluir su correspondiente diagrama de flujo es necesario resumir brevemente el proceso completo, ya que por su complejidad se han tenido que omitir ciertas operaciones en el diagrama, y añadir una serie de comentarios numerados con el fin de explicar determinados puntos de interés.

El proceso comienza incrementando un contador de bajo consumo, el cual se reinicia cada vez que se interactúa con el sistema. Si pasan aproximadamente 15 segundos y no ha habido interacción, el sistema entra en modo de bajo consumo. Para salir de éste, basta con pulsar cualquiera de los botones definidos como interrupción (TAMPER y WAKEUP). Al salir, se reconfigura el RCC ya que al entrar en bajo consumo se cambian registros que de permanecer así, hacen que el sistema no funcione correctamente.

Seguidamente comprueba si se ha insertado alguna tarjeta inteligente. Si la respuesta es afirmativa se procede al Reset en frío y a la recepción del ATR. Una vez hecho esto, se pide el código PIN al usuario. Si es correcto, lee la tarjeta y comprueba los permisos. Si no lo es, vuelve a solicitarlo hasta tres veces y si en ningún caso es correcto se deniega el acceso y vuelve al principio.

Si no se ha insertado tarjeta, comprueba la interactuación con la pantalla. Navegando por el menú, se puede introducir la contraseña. Si es válida se concede el permiso, si por el contrario no lo es, se permite hasta tres veces su introducción y si sigue sin ser válida se produce un tiempo de espera de penalización.

Por último añadir que si no se encuentra en modo de bajo consumo y se presiona el botón TAMPER, el sistema se detendrá para modificar tanto fecha y hora.

Leyenda de numeración:

1. Se utiliza la misma función de RCC que al principio para simplificar el diseño. Dicha función reinicia por defecto las características del RCC y vuelve a configurarlas de la forma necesaria para que el sistema funcione correctamente. Otra opción sería volver a cambiar los registros que se han visto modificados al entrar en el modo bajo consumo. Sin embargo, debido a la dificultad en estudiar dichos cambios (no permite utilizar el modo debug en bajo consumo) se decidió reconfigurar todo.

2. La función para comprobar pantalla táctil se utiliza a lo largo de todo el programa y va muy ligada al bloque principal. Sin embargo su explicación se hará en un bloque independiente "*Pantalla Táctil*" por su complejidad.

3. La variable opción se utiliza para definir el estado actual del sistema y navegar por el menú de interactuación con la pantalla. Según el estado en el que se encuentra y la parte de la pantalla que se presione, el sistema irá a un estado o a otro.

4. Las funciones Editar Fecha y Editar Hora son independientes pero están relacionadas. Según el número de veces que se pulse el botón TAMPER te dirigirá a una o a otra. Para interactuar y cambiar la hora se ha utilizado el Joystick como periférico con el propósito de mostrar todas las posibilidades que tiene la placa de desarrollo. Para volver al bloque principal hay que pulsar el botón central del Joystick, mientras que para cambiar fecha y hora se utilizan los botones de dirección (arriba, abajo, izquierda y derecha).

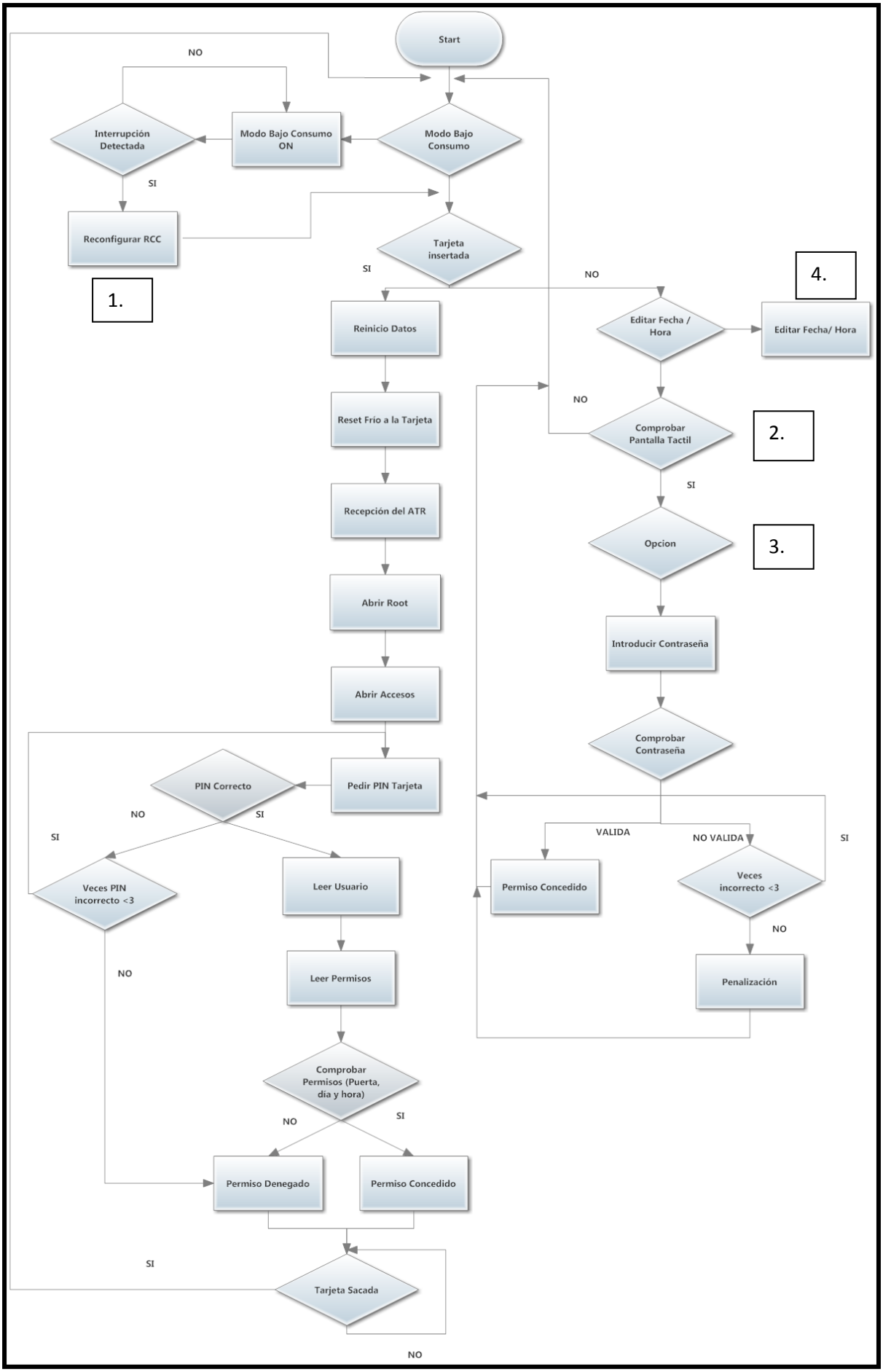


Figura 20: Diagrama de Flujo del Bucle Principal (tomado de [13]).

6.3 Tarjeta Inteligente

6.3.1 Introducción

Ya se dispone de todas las herramientas para entender el funcionamiento y la configuración del sistema, por lo que es momento de comenzar a explicar en detalle las funciones que utilizamos.

Primero se explicarán las funciones utilizadas en relación a la tarjeta inteligente:

- ✓ Reset Frío: No es una función como tal, sino una secuencia de operaciones eléctricas.
- ✓ Recepción del ATR: Esta función es la encargada de recibir el ATR
- ✓ Envío de Comandos: Se utiliza esta función tanto para abrir los ficheros, como para leer los datos, como para comprobar la validez del PIN de seguridad. La función depende de ciertos parámetros que explicaremos y que definen el tipo de operación que se va a llevar a cabo.
- ✓ Comprobación de Permisos: Sirve para comprobar la lectura de los permisos de la tarjeta con los parámetros dentro del microprocesador.
- ✓ Reinicio Datos Tarjeta: Al sacar una tarjeta, todas las variables utilizadas se reinician para un posible uso posterior.

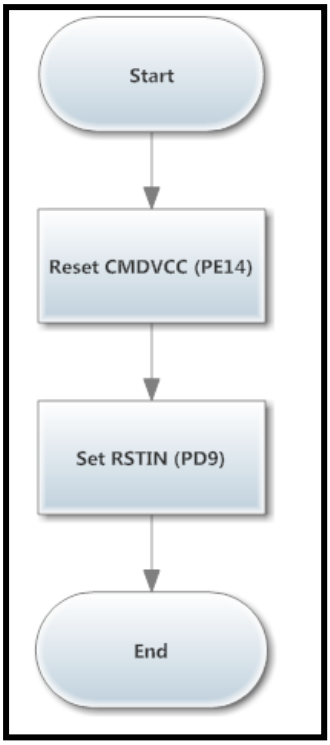
Todas las ejecuciones se realizan por “espera activa” por lo que para que el sistema sea más sólido y no de errores se incluyen sistemas de “time-out” que detectan una mala comunicación enviando un error.

6.3.2 Descripción funcional

6.3.2.1 Reset Frío

Para comenzar la comunicación con una tarjeta inteligente se necesita realizar un procedimiento eléctrico tal y como se comentaba en el capítulo 2 “ESTADO DE LA TÉCNICA”.

Dicho procedimiento eléctrico es el siguiente:



Hay que tener ciertos aspectos en consideración. El primero de ellos es que el PE14 para ser reseteado debe estar puesto inicialmente a 1.

Y el segundo que el PD9 para ser activado debe estar antes a 0. Por lo que en el código se introduce una instrucción adicional para asegurarse de que esto pasa.

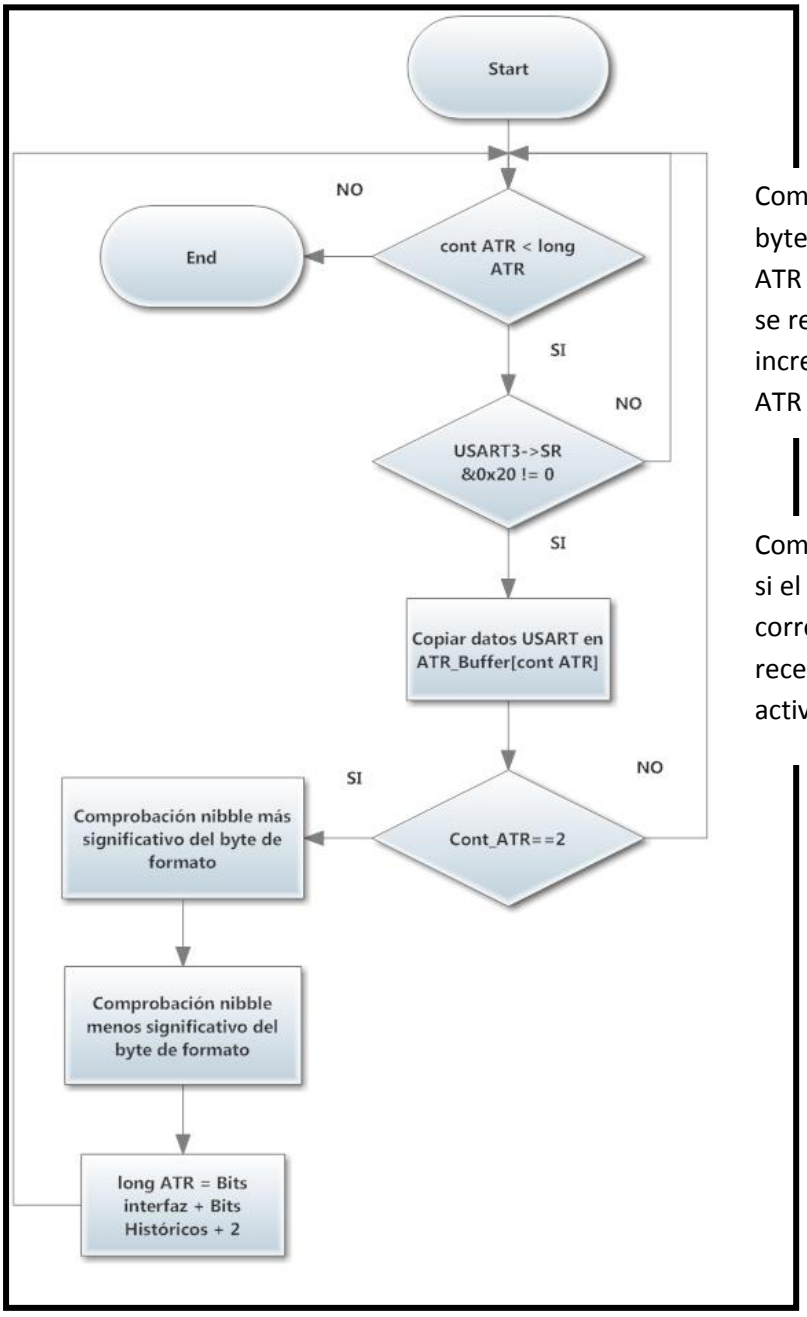
Figura 21: Diagrama de Flujo del Reset Frío para la tarjeta (tomado de [13]).

Al realizar esta secuencia, la tarjeta mandará el ATR, por lo que inmediatamente después debemos de prepararnos para recibirlo.

6.3.2.2 Recepción del ATR

La recepción del ATR se hace mediante ciclos de espera activa. Permanece a la espera hasta que en el registro de la USART hay datos. Entonces copia esos datos en un array y espera al siguiente dato.

La longitud del ATR es variable por lo que se introduce un algoritmo para calcular su longitud en función de sus caracteres de interfaz y caracteres históricos.



Comprobación de cuántos bytes se han recibido del ATR (cont ATR, cada vez que se recibe algo se incrementa) y la longitud del ATR (long ATR)

Comprobación para saber si el bit de estado correspondiente a la recepción de datos está activo o no.

Figura 22: Diagrama de Flujo para la Recepción del ATR (tomado de [13]).

Para conocer el número de bytes de interfaz hay que conocer la estructura completa del byte de formato. No nos detendremos mucho en ello pero con el fin de entender cómo se ha realizado el cálculo de la longitud del ATR se resumirá de forma muy breve.

El byte de formato tiene la siguiente estructura:

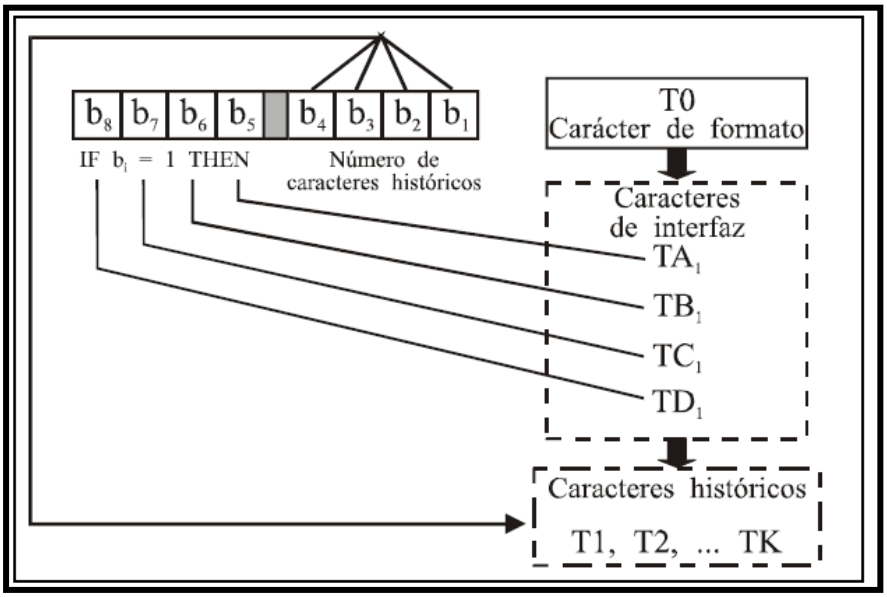


Figura 23: Estructura del byte de formato (tomado de [3]).

Por lo que para calcular la longitud del ATR, basta con fijarse que bit está activado en el nibble más significativo del T0 y cuántos son los caracteres históricos en el nibble menos significativo.

Para mayor detalle consultar el anexo con el código completo (Apartado 9.3 "*Anexo 3: Código Total del diseño*".

6.3.2.3 Envío de Comandos

Para esta función hay que tener muy claro los comandos que se pueden mandar a la tarjeta y cuál es su función. Para recordarlos ir al capítulo 2 apartado 2.3 "*Tarjetas Inteligentes*".

Para el desarrollo del proyecto utilizaremos tres tipos de comandos, siendo el quinto byte la longitud de la cadena que se va a mandar después:

- ✓ Abrir fichero por nombre: Donde la cadena que se va a mandar dependerá del archivo que se vaya a abrir, en nuestro caso puede ser:

  Abrir Root = 0x00, 0xA4, 0x04, 0x00, 0x04, 'ROOT'

  Abrir Access = 0x00, 0xA4, 0x04, 0x00, 0x07, 'ACCESS'

  En este caso no se espera ninguna respuesta, tan solo los bytes de estado indicando que la instrucción se ha ejecutado correctamente.

- ✓ Verificar PIN de seguridad: En este caso se tomará el código PIN insertado por el usuario y se mandará a la tarjeta esperando recibir los dos bytes de estado.
  Verificar PIN = 0x00, 0x20, 0x00, 0x00, 0x08, X (siendo X los 8 bytes del PIN)
- ✓ Leer datos: Este será el único caso en el que se espere recibir contestación de la tarjeta, aparte de los bytes de estado que siempre existen.
  En el sistema queremos leer dos archivos, el primero el nombre del usuario y el segundo sus permisos. El comando para leer los archivos depende de la codificación de la tarjeta y por tanto no se explicará nada al respecto. Para conocer el porqué de dicho comando pueden dirigirse a Manual de Uso Tarjetas FNMT WG10 Versión 2.0 [4]
  Leer 1 = 0x00, 0xB0, 0x81, 0x00, 0x14
  Leer 2= 0x00, 0xB0, 0x82, 0x00, 0x0A

La estructura de la función es la siguiente:

- ✓ Enviar Comandos (Comando, long comando, Respuesta, long respuesta)

Hay que tener muy clara esta estructura ya que el comportamiento de la función depende de los valores que se manden. Las variables Comando y Respuesta son punteros tipo char, donde el primero contiene el comando que se va a mandar y el segundo la respuesta que se va a recibir, mientras que long comando y long respuesta indican la longitud del comando y la longitud de la respuesta que se espera, respectivamente.

Una vez que ya conocemos las instrucciones con las se va a trabajar, se incluye el diagrama de flujo que explica en detalle el procedimiento de actuación de la función.

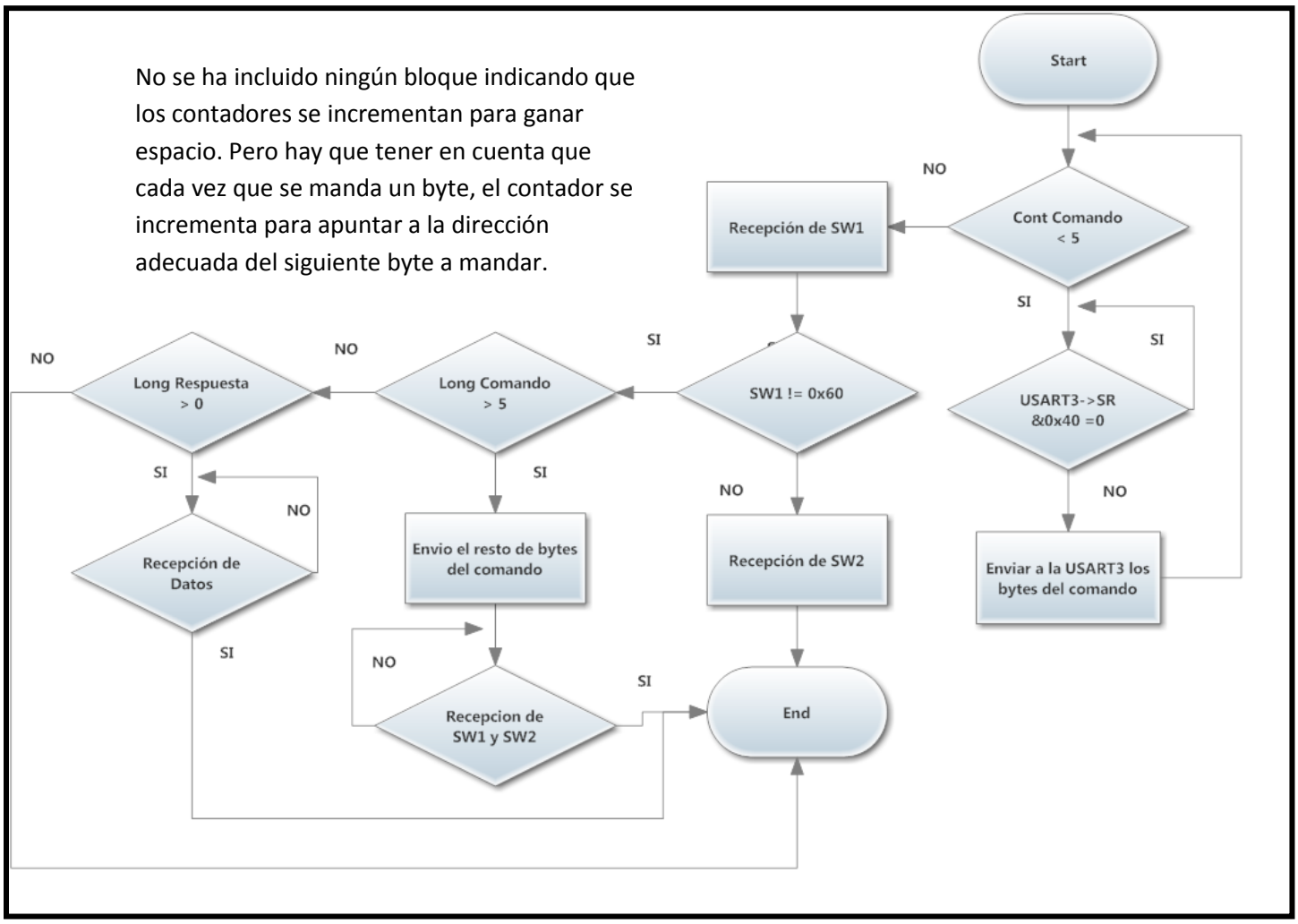


Figura 24: Diagrama de Flujo para Enviar Comando a la tarjeta (tomado de [13]).

Lo primero que se ejecuta en la función es el envío de los 5 primeros bytes del comando, los cuales indican qué tipo de instrucción se está llevando a cabo.

Antes de enviarse cada byte se comprueba que el bit de estado en la USART3 referente a la transmisión está disponible para mandarlo, de no ser así se espera hasta que lo esté.

Una vez que se han enviado los 5 primeros bytes el sistema espera a una posible recepción de bytes de estado, ya que si esto sucediese, el comando

que se ha mandado no iba a mandar más caracteres ni espera recibir datos de la tarjeta.

Si recibe algún byte que no se corresponde con los byte de estado de operación correcta, el sistema tiene dos opciones: la primera es terminar de mandar el comando, siempre y cuando el quinto byte indique una longitud de comando mayor que cero, y esperar la recepción de los bytes de estado.

La segunda opción, es que el sistema espere recibir una cadena de caracteres. Para ello el sistema esperará hasta recibir la cantidad de bytes que indicase la longitud de la respuesta siendo los dos últimos los bytes de estado.

6.3.2.4 Comprobación de Permisos

Los permisos del usuario tal y como se describió en el capítulo 5 "*DISEÑO DEL SISTEMA*" vienen determinados por la lectura del segundo fichero.

Lo primero que se comprueba es si tiene permiso para abrir dicha puerta. Esto se realiza con una comparación entre los 5 primeros bytes de la cadena y la puerta definida en el sistema. Tan pronto como uno de ellos sea igual, el sistema procederá a comparar los horarios y los días. En caso contrario, el sistema denegará el acceso indicando que no tiene permisos para dicha puerta.

En el caso de que sí pueda acceder a la puerta, lo siguiente es comprobar si tiene permisos para acceder en ese día de la semana. Para ello se comprueba si alguno de los bits del último byte coincide con el día actual.

Esta comparación se realiza haciendo un AND lógico entre dicho byte y una "máscara" que indica el día actual. Dicha máscara solo tiene un bit activado y es el correspondiente con el día de la semana actual[13]. Si realizamos el AND

[13] *Nota: Tiene el mismo formato que el último byte del segundo fichero de lectura. Dicho formato es explicado en el Capítulo 5 "DISEÑO DEL SISTEMA"*

lógico entre esta máscara y el último byte del archivo leído y se obtiene el mismo valor de la máscara, significa que ambos compartían ese bit a 1 y por lo tanto tiene permiso. De lo contrario, no puede acceder y saldrá un mensaje notificándolo.

Ya por último solo queda comparar la hora actual con el intervalo de tiempo permitido para el usuario. Para ello, basta con realizar la siguiente operación:

- ✓ Entrada= (lectura[5]*3600)+(lectura [6]*60)
- ✓ Salida= (lectura [7]*3600)+(lectura [8]*60)

Se pasan a horas los bytes que indican la hora de entrada y salida y a minutos los bytes que indican los minutos de entrada y salida, guardándolos en las correspondientes variables.

Siempre que la hora actual sea mayor que la entrada y menor que la salida significará que tiene todos los permisos para acceder a dicho lugar y se abrirá la puerta. En caso contrario, se notificará indicando que el horario no es el admitido.

6.3.2.5 Reinicio Datos Tarjeta

El cometido de esta función es asegurarse de que no se producen errores a la hora de recibir o transmitir información.

Reinicia todos los valores en relación a la tarjeta para evitar posibles interferencias con una comunicación anterior.

Las variables que se reinician son las siguientes:

- ✓ $\overline{\text{PE14}}$ = 1: Desactivación del pin CMDVCC para poder realizar un Reset Frío de nuevo.
- ✓ Reinicio del contador del ATR así como las variables para el cálculo de su longitud
- ✓ Reinicio tanto del array del ATR como del de respuesta.

Para realizar dichos reinicios basta con definir un bucle que guarde el valor 0 en el array tantas veces como posiciones tenga.

6.4 Pantalla Táctil

6.4.1 Introducción

La pantalla táctil es el periférico más utilizado en el diseño, sirve como interfaz de comunicación entre el usuario y la placa, y como método de identificación secundario.

Como consecuencia de lo anterior se han definido múltiples funciones que para su explicación se dividirá en dos bloques. El primero de ellos, será explicar la función utilizada como interfaz. Mientras que el segundo bloque contendrá todas las funciones que se han utilizado para que el usuario pueda escribir, así como los tipos de teclado definidos y las funciones para comprobar dichos datos.

Antes de comenzar a desarrollar las funciones es imprescindible conocer las dimensiones de la pantalla así como su cuadrícula y división de los puntos de presión.

En el desarrollo del proyecto se aprendió a utilizar este periférico mediante la creación de un programa independiente al sistema de control de accesos. Este programa se puede describir como un análogo al programa Paint de Microsoft donde se podían elegir hasta tres colores y pintar sobre la pantalla.

Este mini-desarrollo ayudó a entender y elaborar una cuadrícula, la cual se muestra a continuación, de las coordenadas que se utilizan para calcular los puntos de presión[14].

En la tabla inferior se muestra:

[14] *Nota: Dichas coordenadas se calculan a partir de una función de las librerías de ST, donde se estudia el estado del periférico y calcula una serie de coordenadas (x,y,z). El sistema solo se centrará en (x,y) ya que z no aporta ningún dato sobre la dirección, sino sobre la cantidad de presión ejercida.*

- ✓ Líneas: Se trata de la definición en el sistema de las 10 líneas sobre las que se puede mostrar mensajes por pantalla.
- ✓ Vertical: Corresponde a la coordenada "y" del punto de presión. Los números definidos corresponden siempre a la parte superior de la celda, siendo 0 la primera y 240 la última.
- ✓ Horizontal: Corresponde a la coordenada "x" del punto de presión. En este caso, los números corresponden a la parte izquierda de la celda, empezando en 0 y acabando en 320.

		horizontal																				
líneas	vertical	0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240	256	272	288	304	320
linea0	0																					
linea1	24																					
linea2	48																					
linea3	72																					
linea4	96																					
linea5	120																					
linea6	144																					
linea7	168																					
linea8	192																					
linea9	216																					
	240																					

Tabla 11: Esquema Pantalla Táctil 1 (tomado de [13]).

Una vez comprendidas las variables que se utilizan para calcular dónde se presiona la pantalla se puede proceder a explicar las funciones mencionadas anteriormente.

6.4.2 Descripción funcional

6.4.2.1 Función de Interfaz

Esta función, llamada Comprobar_Pantalla, fue introducida en el apartado 6.2 "**¡Error! No se encuentra el origen de la referencia.**". Sirve para diseñar un enú que según la variable opción (estado en el que se encuentre) y la posición de pantalla con la que se interactúe, el sistema cambiará de estado o seguirá en el mismo.

Para simplificar su explicación se muestra la tabla anterior para cada una de los estados del programa. Es necesario recordar que para consultar cualquiera de las funciones que presente datos por pantalla, ya sea mensajes o dibujos, se tendrá que recurrir al anexo con el código completo. Aunque, en el Capítulo 7 "*PRUEBAS Y RESULTADOS*" se incluyen una serie de "pantallazos" que muestran el resultado final del programa.

Para comenzar con la explicación de esta función se definen hasta tres posibilidades dónde se estudiarán los puntos de presión sobre la pantalla[15].

- ✓ Menú inicial: Opción = 1.
- ✓ Escribir clave: Opción = 3 ó 5.
- ✓ Escribir código PIN: Opción = 30 y 31.

Opción=1:

La primera posibilidad corresponde con la siguiente pantalla:

		horizontal																				
líneas	vertical	0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240	256	272	288	304	320
linea0	0																					
linea1	24																					
linea2	48				S	I		H	A		O	L	V	I	D	A	D	O				
linea3	72						L	A		T	A	R	J	E	T	A						
linea4	96																					
linea5	120																					
linea6	144						P	U	L	S	E		A	Q	U	I						
linea7	168																					
linea8	192																					
linea9	216																					
	240																					

Tabla 12: Esquema Pantalla Táctil 2 (tomado de [13]).

En este caso, solo se pasará a la siguiente opción si se pulsa dentro del rectángulo blanco con mensaje "Pulse Aquí". Esto corresponde con las coordenadas:

[15] *Nota: Los valores intermedios para la variable "opción" no se estudian porque o bien son solo pasos intermedios para representar por pantalla o corresponden a las funciones de definir puerta y contraseña de las que hablamos al principio.*

✓ (20<horizontal<300) & (130<vertical<180)

Si la pulsación no está entre esos valores, el no cambiará a no ser que se introduzca la tarjeta inteligente tal y como se mostraba en el diagrama de flujo del main. Si por el contrario sí está en ese intervalo, el estado cambiará a la "opción=3".

Opción = 3 ó 5:

Para la "opción = 3" se estudia esta pantalla:

		horizontal																				
líneas	vertical	0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240	256	272	288	304	320
linea0	0																					
linea1	24																					
linea2	48	P	A	S	S	W	O	R	D	:												
linea3	72																					
linea4	96																					
linea5	120																					
linea6	144							A	C	E	P	T	A	R								
linea7	168																					
linea8	192																					
linea9	216																					
	240																					

Tabla 13: Esquema Pantalla Táctil 3 (tomado de [13]).

Si el usuario se encuentra en este estado significa que ha olvidado la tarjeta y va a proceder a escribir la contraseña. En este caso, se comprueban varios focos de presión.

El primero de ellos es la línea 3 completa. Si se pulsa en cualquier punto de la línea 3 (72<vertical<96 & 0<horizontal<320) el sistema pasará a la opción 5 donde aparecerá un teclado para escribir.

Si por el contrario pulsa sobre la línea 6 (144<vertical<168 & 0<horizontal<320) el sistema pasará a la opción 6 dónde se comprobará la contraseña para permitir o no el acceso.

Para estudiar la pantalla táctil sólo interesa cuando la opción pasa al valor 5, en el cual se muestra el teclado como se dijo anteriormente.

		horizontal																			
líneas	vertical	0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240	256	272	288	304	320
linea0	0																					
linea1	24																					
linea2	48	P	A	S	S	W	O	R	D	:												
linea3	72																					
linea4	96																					
linea5	120																					
linea6	144	1		2		3		4		5		6		7		8		9		0		
linea7	168	q		w		e		r		t		y		u		i		o		p		
linea8	192	a		s		d		f		g		h		j		k		l		<<		
linea9	216			z		x		c		v		b		n		m				ok		
	240																					

Tabla 14: Esquema Pantalla Táctil 4 (tomado de [13]).

La función de escribir se explicará más adelante, aquí nos centraremos en decir que sólo se saldrá de esta opción si se pulsa fuera del teclado o se da a la tecla de "ok" situada en la esquina inferior derecha del teclado.

Por lo tanto, cuando se detecte una pulsación fuera del teclado, valor (vertical<144), o bien se pulse la tecla "ok" (288<horizontal<320 & 216<vertical<240), se volverá a la "opción=3".

En cualquier otro caso significa que se están pulsando las letras del teclado y por tanto, se activa la función de escribir.

Opción = 30 ó 31:

Este estado es alcanzado si se ha introducido la tarjeta inteligente y nos pide el PIN de seguridad. Las pantallas que se estudian son análogas a los casos para escribir la clave sin tarjeta, con la única diferencia que en este caso no se necesita un teclado alfanumérico.

Cuando el sistema se encuentra en la "opción=30" se estudian los mismos valores que para "opción=3".

		horizontal																				
líneas	vertical	0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240	256	272	288	304	320
linea0	0																					
linea1	24																					
linea2	48	I	N	T	R	O	D	U	C	E		P	I	N	:							
linea3	72																					
linea4	96																					
linea5	120																					
linea6	144							A	C	E	P	T	A	R								
linea7	168																					
linea8	192																					
linea9	216																					
	240																					

Tabla 15: Esquema Pantalla Táctil 5 (tomado de [13]).

En el caso de que la pulsación se corresponda con la línea 3, se pasará a la "opción=31" mostrando el teclado numérico para introducir el PIN. Si por el contrario se pulsa la línea 6, se pasará a la opción=32 donde se comprobará la validez del PIN dando o no acceso al usuario.

La pantalla que se estudia para introducir el código PIN es la siguiente, siendo muy similar a la de introducir la clave si se te ha olvidado la tarjeta.

		horizontal																				
líneas	vertical	0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240	256	272	288	304	320
linea0	0																					
linea1	24																					
linea2	48	I	N	T	R	O	D	U	C	E		P	I	N	:							
linea3	72																					
linea4	96																					
linea5	120																					
linea6	144			1				2				3										
linea7	168			4				5				6						OK				
linea8	192			7				8				9										
linea9	216							0										Borrar				
	240																					

Tabla 16: Esquema Pantalla Táctil 6 (tomado de [13]).

Al igual que para su análogo (opción=5) si se pulsa fuera de los límites de teclado o en la tecla "ok" (192<horizontal<320 & 144<vertical<192) se volverá a "opción=30". De lo contrario, se llama a la función para escribir el código PIN.

Por último añadir que por seguridad, el sistema borra los datos anteriormente introducidos al pasar de las opciones 3 y 30 a las 5 y 31 respectivamente.

Una vez conocidas estas funciones, es posible manejar el sistema de forma completa, ya sea introduciendo la tarjeta o navegando por las diferentes pantallas del menú.

6.4.2.2 Funciones de escritura

Para finalizar con el apartado 6.4 "Pantalla Táctil" sólo queda definir la forma de escribir y guardar datos. La forma de escribir ha pasado a ser bastante intuitiva después de conocer el funcionamiento de la pantalla táctil, así como sus dimensiones y parámetros.

Tanto para el teclado alfanumérico, como para el numérico, las pantallas para estudiar las posiciones serán las tablas 14 y 16 respectivamente del apartado anterior, donde habrá que estudiar las coordenadas horizontal y vertical de la pulsación para conocer el carácter que se está pulsando.

Una vez que se ha realizado eso, de igual forma que se estudiaba en "*Función de Interfaz",* queda definir cómo se guardan los datos y como se representan.

En ambos casos por seguridad, no se mostrará por pantalla el carácter introducido, sino un "*". Sin embargo, la letra/número pulsado se grabará internamente en la posición de un array según el número de caracteres introducidos.

Para resumir todo lo explicado anteriormente se incluye un diagrama de flujo mostrando el proceso según la variable opción:

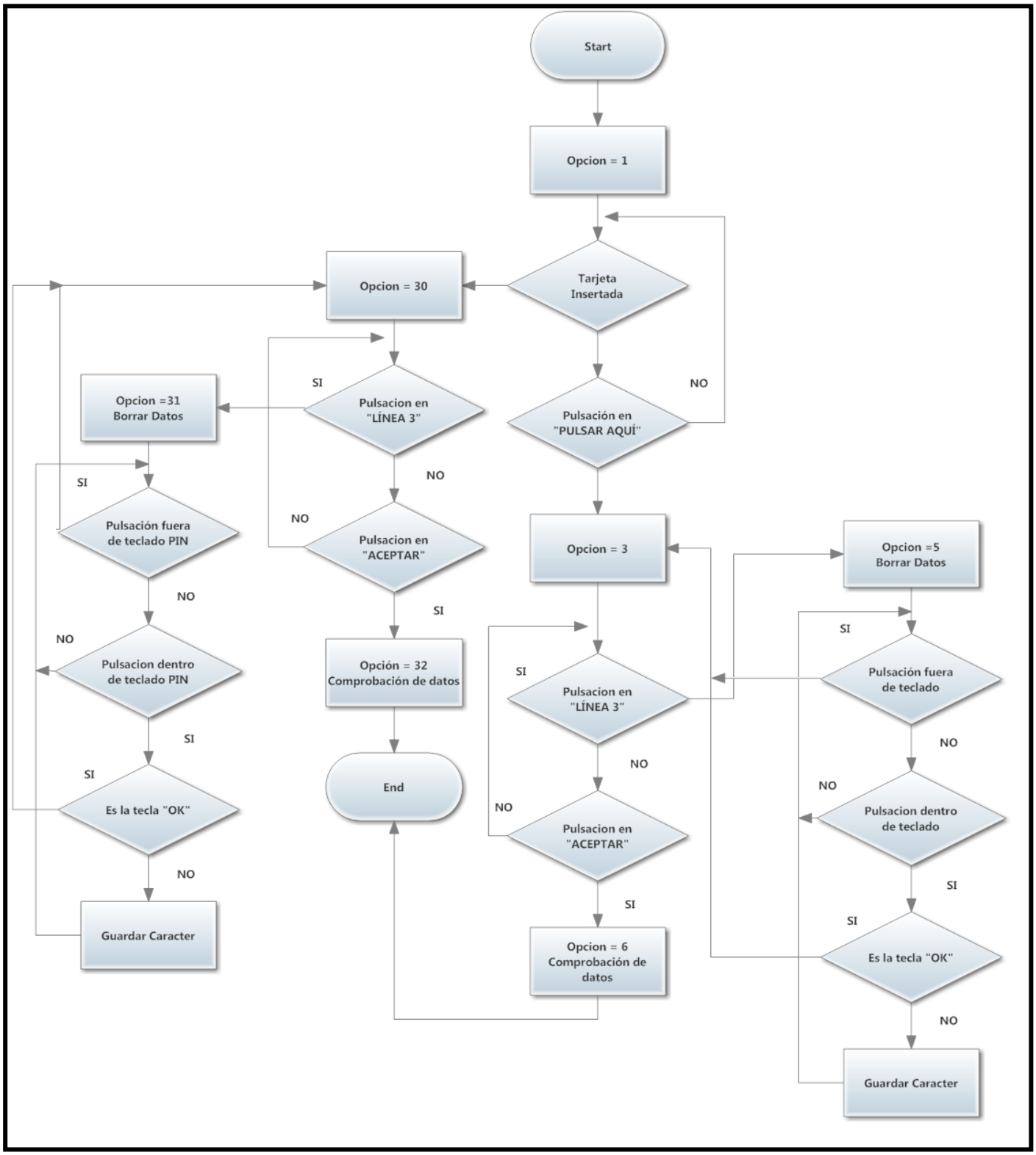


Figura 25: Diagrama de Flujo para Insertar datos por teclado (tomado de [13]).

Cabe destacar que se ha incluido una mejora para la introducción de datos en el teclado alfanumérico. Tal y como se observa en la tabla 14, el teclado definido es relativamente pequeño para saber con certeza la tecla que se

está pulsando. Por lo que se decidió incluir en la línea 5 un array que presentase la letra que se está pulsando. Un ejemplo se muestra en la siguiente figura:

S	S	W	O	R	D	:
		e				
2		3		4		
w		e		r		

Tabla 17: Detalle Teclado alfanumérico (tomado de [13]).

6.5 Interrupciones

6.5.1 Introducción

En este apartado se resumen las actuaciones que se realizan por interrupción fuera del programa principal. Estas posibles interrupciones son las siguientes:

- ✓ RTC
- ✓ Botón TAMPER
- ✓ Botón WAKEUP

Como se comentó anteriormente, el contador del RTC funciona de forma totalmente independiente al sistema y por tanto no hay nada que influya en el periférico. Sin embargo, se decidió crear una rutina de interrupción para este periférico con el fin de ser capaces de representar la hora y fecha de forma prioritaria.

Por lo tanto, la interrupción del RTC sirve para actualizar el valor que se presenta por pantalla del tiempo. Si no se hubiese establecido así, aunque el valor del contador del RTC fuera correcto, se vería desfasado.

La siguiente interrupción es la del botón TAMPER. Al estar definida como interrupción externa, permite despertar al sistema del modo de bajo consumo. Pero su principal función reside en permitir el cambio de hora y fecha del sistema. Las funciones que permiten dicho cambio se explicarán en el apartado 6.6 "*Funciones de carácter general*".

La última interrupción es la del botón WAKEUP. Este es el botón accesible para el usuario y su única misión es despertar al sistema del bajo consumo.

6.5.2 Descripción funcional

6.5.2.1 Interrupción del RTC

Esta interrupción se ejecuta cada vez que el contador del RTC cuenta 1 segundo. Cada vez que esto se produce se transforma tanto el valor del contador para representar la hora[16], como los valores de día, año y mes para representar la fecha.

El cálculo de la hora se divide en unidades y decenas de cada medida ya que cada cifra se representa en una posición distinta del array[17]:

- $unidades\ segundo = \big((valorRTC\ \%\ 3600)\ \%\ 60\big)\ \%\ 10$
- $decenas\ segundo = \frac{\big((valorRTC\ \%\ 3600)\ \%\ 60\big)}{10}$
- $unidades\ minuto = \frac{\big((valorRTC\ \%\ 3600)\ \%\ 600\big)}{60}$
- $decenas\ minuto = \left(\frac{(valorRTC\ \%\ 3600)}{600}\right)$
- $unidades\ hora = \left(\frac{valorRTC\ \%\ 36000}{3600}\right)$

[16] *Nota: Los cambios de segundos a minutos y de minutos a hora se hacen automáticamente.*
[17] *Nota: La operación con "%" calcula el resto de la división de esos números.*

- $decenas\ hora = \frac{valorRTC}{36000}$

En relación a la hora, ya sólo queda indicar que si se llega a un valor igual o superior a las 24 horas, el contador se reinicia y se suma un día.

La fecha también se divide en unidades y decenas para representarse en el mismo array que la hora. En este caso los cálculos son los siguientes[18]:

- $unidades\ día = dia\ \%\ 10$
- $decenas\ día = \frac{dia\%100}{10}$
- $unidades\ mes = month\ \%\ 10$
- $decenas\ mes = \frac{month\ \%\ 100}{10}$
- $unidades\ año = (year\ \%\ 10)$
- $decenas\ año = \frac{year\ \%\ 100}{10}$

Una vez que se han calculado los valores de cada carácter a representar, tan solo queda añadir que para presentarlo por pantalla se deberán pasar de hexadecimal a código ASCII. Para ello, basta con sumar a cada uno el carácter '0' en ASCII.

Las dos últimas actuaciones que se realizan en esta interrupción son: escribir en los registros de backup el año, el mes y el día y definir el día de la semana correspondiente.

Para definir cuál es el día de la semana correspondiente a esa fecha en particular, se utiliza una ecuación denominada como "Fórmula de Zeller". La función utilizada se explica en el apartado 6.6 "*Funciones de carácter general*".

Esta interrupción se centra en definir el carácter a representar según el resultado de dicha fórmula (L: Lunes, M: Martes, X: Miércoles, J: Jueves,

[18] *Nota: El día de la semana y los cambios de día según el mes se realizan funciones que se explicarán en el apartado 6.6 "Funciones de carácter general".*

V: Viernes, S: Sábado y D: Domingo) y simultáneamente de definir la máscara que se utilizaba para comprobar los permisos de la tarjeta.

6.5.2.2 Interrupción del botón TAMPER y WAKEUP

La interrupción del botón WAKEUP sólo sirve para sacar al sistema del modo bajo consumo y reiniciar el contador que hace que este modo vuelva a activarse.

Estas mismas acciones se llevan a cabo en la interrupción del botón TAMPER, pero adicionalmente, según el número de veces que se pulse, se llama a la función de cambiar hora o a la función de cambiar fecha.

Se incluye un pequeño diagrama de flujo para representar estas operaciones:

Si se utiliza el botón TAMPER para salir del modo de bajo consumo, no debe saltar ninguna función de editar, por lo que se añade la comprobación de "Modo Bajo Consumo ON"

Según se pulse el botón fuera del bajo consumo se irán alternando los modos de editar fecha y hora sin necesidad de pasar por el programa principal.

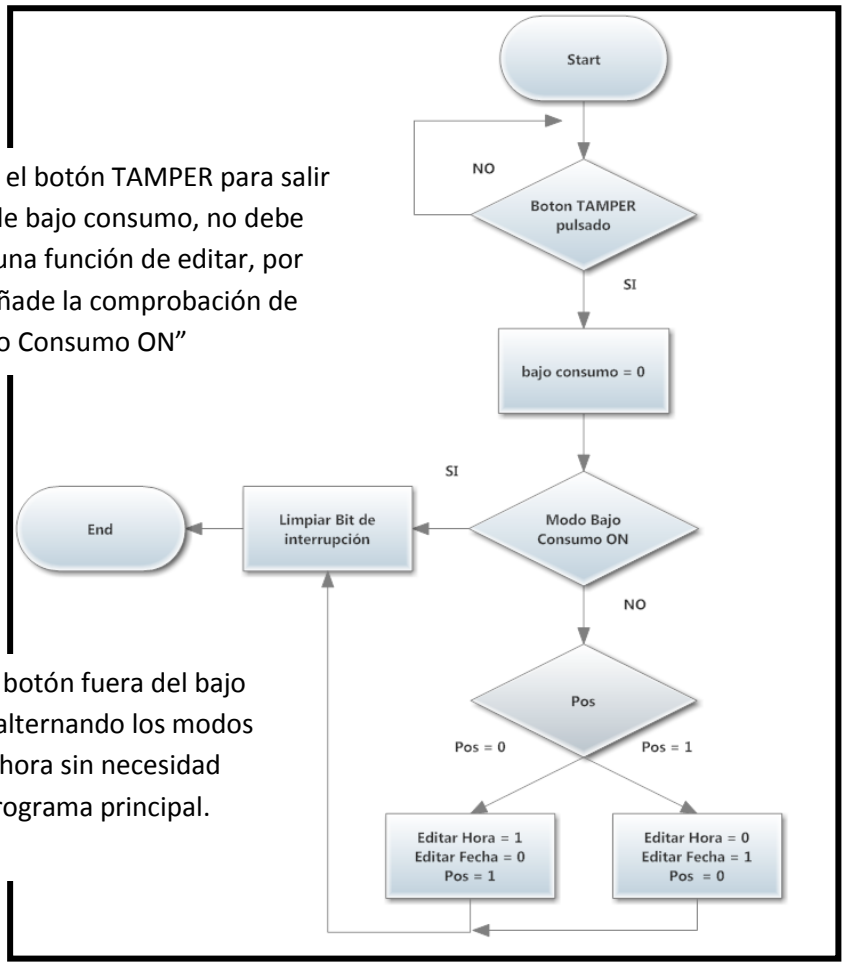


Figura 26: Diagrama de Flujo de Int. Wakeup y Tamper (tomado de [13]).

6.6 Funciones de carácter general

6.6.1 Introducción

Llegados a este punto ya se conoce el funcionamiento completo del sistema y lo único pendiente es explicar las funciones que utilizadas y que todavía no se han explicado:

- ✓ Introducción inicial de puerta y clave: Tal y como se comentó en el apartado 6.2 "**¡Error! No se encuentra el origen de la referencia.**" en la nicialización se añadieron dos operaciones para introducir la clave y la puerta del sistema.
- ✓ Comprobación de clave y horario: Estas funciones se encargan de comprobar carácter a carácter la cadena introducida por el usuario y la guardada en el sistema, así como los permisos de horario.
- ✓ Reinicio datos: Existen dos funciones de reinicio de datos. La primera referente a los datos de la tarjeta, la cual ya se explicó en el apartado correspondiente, y ésta que realiza operaciones similares pero está más relacionada con los datos introducidos por pantalla.
- ✓ Calendario: Función que define el número de días de cada mes y si el año es bisiesto.
- ✓ Día de la semana: Aquí es donde implementamos la fórmula de Zeller para calcular el día de la semana.
- ✓ Cambio de fecha y hora: Estas funciones, cuyo uso se activa con la interrupción del botón TAMPER se encargan de cambiar la hora y la fecha del sistema.

6.6.2 Descripción funcional

6.6.2.1 Introducción inicial de puerta y clave

La seguridad de este sistema al introducir este tipo de funciones se reduce drásticamente. Sin embargo, ha permanecido en el desarrollo para facilitar enormemente las pruebas del resultado final.

Por otra parte, la inclusión de estas funciones no sería tan disparatada si se incluyese un algoritmo adicional de seguridad para ejecutarlas, lo cual se comentará más adelante en el capítulo 8 “CONCULSIONES Y LÍNEAS FUTURAS”.

Al no tratarse de un objetivo de un sistema de control de accesos, su desarrollo se ha implementado a partir de funciones ya creadas modificando ligeramente su presentación.

Para introducir el número de puerta, se han utilizado las funciones usadas en la introducción del código PIN de seguridad con algunas modificaciones. Dichas modificaciones son la obligatoriedad de introducir dos cifras para indicar el número de puerta, es decir, para definir el sistema como puerta 6 se debe introducir por pantalla “06”. Además, no se contemplan más de 99 puertas, por lo que no se tendrá en cuenta más de dos cifras introducidas.

Otra modificación es ver los datos que se están introduciendo (no se sustituyen por “*”). Y por último, al pulsar el botón “Aceptar” se graban los datos.

Por otra parte, para introducir la clave del sistema, se han utilizado las funciones para escribir la contraseña cuando no se dispone de tarjeta inteligente.

La única modificación es que también se permite ver los caracteres que se están introduciendo. Y de nuevo al pulsar el botón de “Aceptar” se grabarán los datos.

6.6.2.2 Comprobación de clave y horario

La comprobación de clave se basa en un bucle de 20 repeticiones (número de caracteres máximos que se pueden escribir" que compara la cadena introducida por el usuario y la guarda en el sistema como clave.

En el momento en el que exista un carácter distinto saltará una alarma para indicar que las cadenas no son iguales y se incrementará un contador. Si dicho contador llega a ser 3 (cuando se ha introducido hasta 3 veces mal la clave) se activará otra alarma indicando que debe haber una penalización de con un tiempo de espera de 10 segundos. Durante este tiempo el usuario no puede hacer nada.

Cuando la cadena sea correcta se pasará a comprobar el horario. Dicha comprobación se hace de forma análoga a la comprobación con los datos de la tarjeta, con la diferencia de que los valores de entrada y salida están definidos en el sistema.

Al igual que en el caso anterior, si los intentos fallidos llegan hasta 3, se ejecutará la penalización. Cuando el horario sea el correcto, se abrirá la puerta.

6.6.2.3 Reinicio datos

Como ya se adelantó, esta función reinicia los datos introducidos por el usuario. Estos datos corresponden a:

- ✓ PIN: El código de seguridad introducido para la tarjeta
- ✓ Clave introducida: Clave de seguridad introducida por el usuario en caso de no disponer de la tarjeta.

Además, se reinician también las cadenas que mostraban la clave y PIN ocultos por asteriscos y el array que mostraba la letra que se estaba pulsando en el teclado alfanumérico.

6.6.2.4 Calendario

Con esta función se definen los meses con 31 días, con 30 días y se estudia si el año es bisiesto para definir los días de febrero. Para simplificar la explicación se añade un diagrama que muestra las comparaciones y operaciones que se realizan.

El funcionamiento varía ligeramente cuando el usuario se encuentra modificando la fecha. En ese caso, ni los meses cambian por los días, ni los años por los meses.

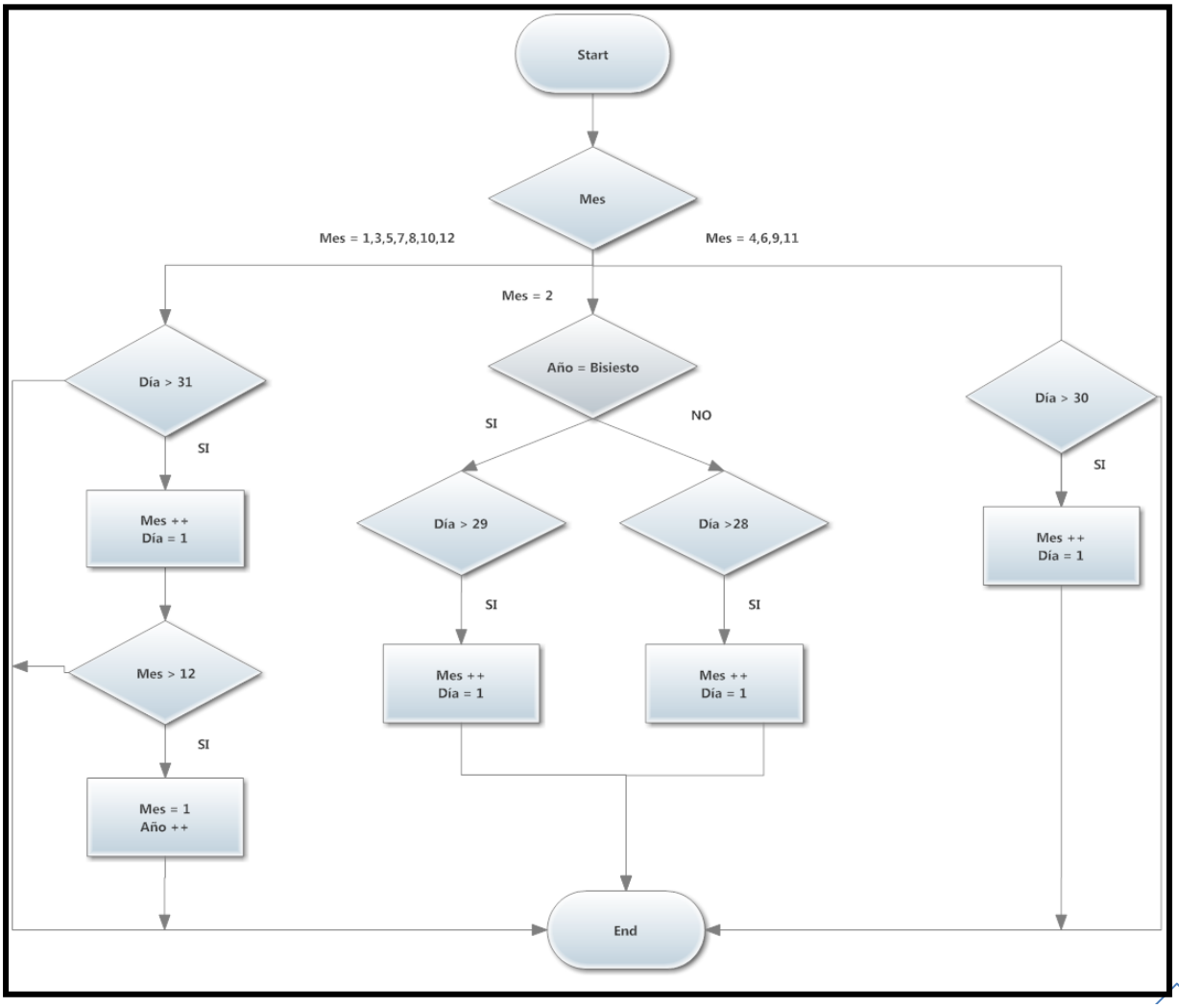


Figura 27: Diagrama de Flujo de la función Calendario (tomado de [13]).

6.6.2.5 Día de la semana

La fórmula de Zeller es capaz de calcular el día de la semana teniendo como datos día mes y año. Sin embargo, los meses cambian su numeración para que la fórmula funcione.

Marzo pasa a ser el mes 1, y se sigue contando hasta diciembre que llega a ser el mes 10. Enero y febrero son los meses 11 y 12 respectivamente pero si nos encontramos en uno de ellos se toma el año anterior y no el actual.

Pongamos dos ejemplos: el 16/4/2012 y el 21/2/2012. La fórmula de Zeller se define como la siguiente:

$$Día\ Semana = \left(día + \left(\frac{13 \cdot mes - 1}{5}\right) + y + \frac{y}{4} + \frac{c}{4} - 2 \cdot c\right) \% 7$$

- día = día del mes actual
- mes = mes según la numeración anterior: Mar = 1, Abr = 2…, Feb = 12
- y = dos últimas cifras del año
- c = dos primeras cifras del año

El resto de la división entre 7 dará un número del 0 al 6 donde el 0 se corresponde con el domingo, el 1 con el lunes y así sucesivamente hasta el 6 que es sábado.

Para el 16/4/2012 las variables tomarían los siguientes valores:

- día = 16 / mes = 2 / y = 12 / c = 20 → Dando como resultado Día Semana = 1 , es decir, lunes.

Para el 21/2/2012 sería de la siguiente forma:

- día = 21 / mes = 12 / y = 11 / c = 20 → Y el resultado sería 2, martes.

Esta ecuación es la misma que se ha implementado en esta función. Por lo que para definir la máscara del día de la semana se comprueba el valor del resultado y se asigna el valor correspondiente.

6.6.2.6 Cambio de fecha y hora

Los cambios de fecha y hora son dos funciones distintas, aunque ambas se activan con la pulsación del botón TAMPER

Las dos utilizan un algoritmo muy parecido en función de la interactuación con el Joystick. En caso de que se pulse en las direcciones de arriba o abajo, el sistema sumará o restará la cifra correspondiente a la unidad que se esté cambiando. Mientras que si se pulsan las direcciones de derecha o izquierda, se cambiará la unidad que se está modificando. Por ejemplo, si se está en minutos, cada pulsación hacia arriba sumará un minuto al reloj, mientras que si se pulsa hacia abajo restará. Al pulsar sobre la izquierda, se pasará a modificar las horas, mientras que en la derecha no surgirá efecto ya que el cambio de segundos es innecesario.

Al pulsar el botón central en cualquiera de las funciones de saldrá del modo edición para volver al bucle principal.

Por último añadir ciertas características de esta función:

- ✓ Se ilumina en rojo la unidad que se esté cambiando.
- ✓ Al estar este modo activado, se deja de tomar el valor del contador del RTC, aunque éste se vaya actualizando.
- ✓ Al salir de este modo, el valor del RTC se cambia al valor resultante en la modificación.
- ✓ Durante la modificación si se sobrepasan las 24 horas, todos los valores se inicializan en 0.
- ✓ En el cambio de fecha es imposible colocar un número de días no apto para el mes en cuestión.

7 PRUEBAS Y RESULTADOS

Una vez que el desarrollo del sistema ha concluido, nos centraremos en explicar cómo se realizaron las pruebas y cuál ha sido el resultado obtenido.

Para ello, este capítulo se dividirá en dos apartados. El primero de ellos comentará los métodos que se han seguido para realizar correctamente todas las pruebas necesarias. Y el segundo mostrará por imágenes el funcionamiento final del sistema con la última prueba vez que se llevó a cabo.

7.1 Métodos de pruebas

Principalmente se han utilizado dos métodos para comprobar que el sistema funciona correctamente. El primero, una herramienta específica del sistema de desarrollo, y el segundo una combinación de pruebas personales y familiares.

En el entorno de desarrollo existe la posibilidad de contemplar cómo van cambiando los registros de los periféricos. Al utilizar el modo debug e incluir puntos de ruptura, dichos registros se van actualizando mostrando el estado de todos sus bits.

Esta herramienta no se descubrió hasta bien entrado en el desarrollo, por lo que anteriormente se perdió mucho tiempo en repetir de diferentes formas una misma instrucción, que a lo mejor no funcionaba por otro motivo. Sin embargo, una vez descubierta, los errores y cambios necesarios se detectaban rápidamente.

Por otra parte, la posibilidad de incluir parámetros como variables tipo “watch” ayudó a estudiar la evolución de las variables y descubrir rápidamente dónde estaban los fallos.

Este primer método se utilizó principalmente mientras se desarrollaba el sistema. Una vez que el desarrollo alcanzó la fase final se utilizó el segundo método.

Hay cierto tipo de errores que no se pueden detectar con ninguna herramienta en concreto. Estos errores son provocados por un mal uso o el desconocimiento total del sistema. Para detectar ese tipo de fallos es conveniente realizar pruebas “a malas”, es decir, intentar “volver loco al sistema”.

Este tipo de pruebas no resultan tan eficientes si es quién ha diseñado el sistema quien las hace. Por lo tanto, cada vez que se creía que el diseño era completo, se pedía a familiares y amigos que probasen la aplicación, intentando descubrir fallos.

Esto descubrió múltiples fallos ya que realizaban acciones que “no estaban contempladas” en el sistema. Como por ejemplo, sacar la tarjeta inteligente cuando se está en pleno proceso de verificación o pulsar varios botones a la vez.

Sin embargo, todas esas acciones consiguieron revelar errores desconocidos, permitiendo lograr un diseño del sistema más sólido.

7.2 Resultado Final

A continuación se va a mostrar la última prueba realizada mostrando algunas fotografías del sistema.

Al conectar la placa de desarrollo a la fuente de alimentación, el sistema se inicia pidiendo que se introduzca la puerta y la contraseña.

Fotografía 1: Definición de puerta

Fotografía 2: Definición de contraseña

Una vez que introducidas la puerta y contraseña, el sistema muestra una breve presentación de bienvenida.

Fotografía 3: Mensaje de bienvenida

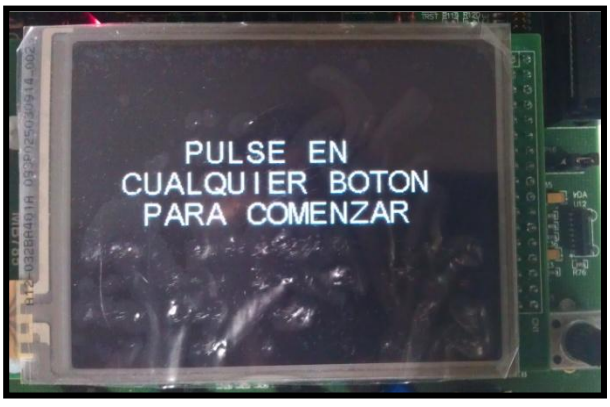
Fotografía 4: Mensaje de "Universidad Carlos III de Madrid"

A continuación ya se puede interactuar con el sistema. La pantalla por defecto permite indicar al sistema que no disponemos de la tarjeta.

Existen tres posibilidades de actuación. La primera es pulsar en la pantalla para indicar que se ha olvidado la tarjeta. La segunda es introducir la tarjeta y la tercera no hacer nada esperando que el sistema se ponga en bajo consumo.

Fotografía 5: Pantalla Inicial

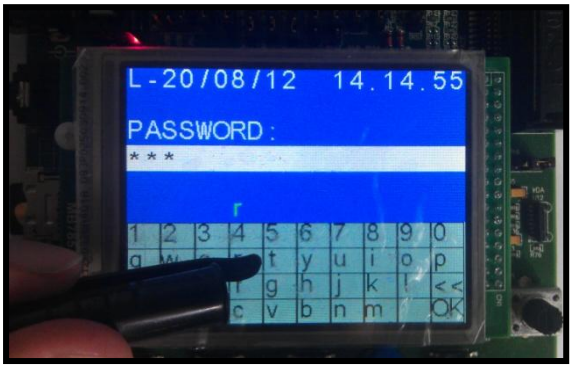
Si dejamos el sistema 15 segundos, mostrará la siguiente pantalla indicando que se encuentra en modo bajo consumo y para salir se debe pulsar o bien el botón TAMPER, o bien el WAKEUP. Al salir se vuelve a la pantalla anterior.



Fotografía 6: Modo bajo consumo activado

En el caso de pulsar indicando que hemos olvidado la tarjeta, el sistema nos permitirá introducir una clave que comparará con la introducida anteriormente (“default”).

Para la primera prueba se introduce otra contraseña y se comprueba el mensaje que sale al darle a “Aceptar”.



Fotografía 7: Introducción de contraseña

Fotografía 8: Datos no correctos

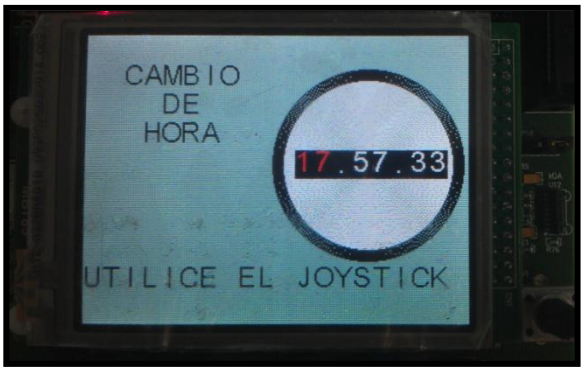
Obviamente sale que la clave no es la correcta y nos devuelve de nuevo a la pantalla inicial. Ahora introduciremos correctamente la contraseña y para comprobar los mensajes que salen.

Fotografía 9: Datos correctos

Fotografía 10: Horario no admitido

En este caso nos sale que los datos son correctos pero que el horario no está admitido al ser las 14:15 horas. Se recuerda que se diseñó el sistema de tal forma que por teclado solo se pudiera pasar en el horario desde las 16:00 hasta a las 19:00 horas.

Sin embargo, se tiene la posibilidad de cambiar la hora para permitir el acceso sin tarjeta. Por lo tanto cambiamos la hora a una comprendida entre las 16 y las 19 horas y volvemos a repetir el procedimiento anterior.



Fotografía 11: Modo cambio de hora

Ahora se cumplen todos los requisitos para acceder a la zona, por lo los mensajes son los siguientes:

Fotografía 12: Contraseña correcta

Fotografía 13: Puerta abierta por contraseña

A continuación introduciremos la tarjeta electrónica de Raúl. Al introducir la tarjeta se pide el código PIN. Entonces introduciremos el código PIN correcto, en este caso es el 56785678 y pulsaremos en OK.

Cuando se oculte el teclado numérico pulsaremos el botón de aceptar y el sistema comprobará si ese pin introducido es el correcto.

Al ser correcto seguirá comprobando los permisos de la tarjeta. Verificará si tiene acceso en la hora y día de la semana en la que nos encontramos.

Fotografía 14: Introducción PIN. Usuario Raúl

Raúl tiene acceso las 24 horas y todos los días de la semana por lo que no debería de salir ningún mensaje de error, abriendo la puerta sin problemas.

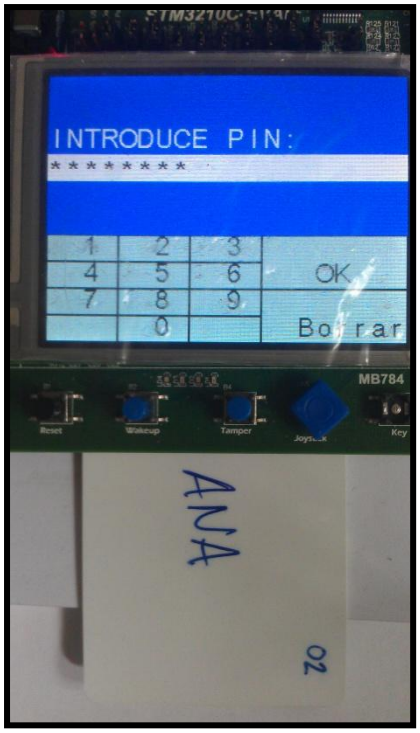
Los siguientes mensajes, muestran que los datos son correctos y que el sistema abre la puerta.

Fotografía 15: Usuario Raúl

Fotografía 16: Puerta abierta por el usuario Raúl

Ahora se prueba con la tarjeta de Ana. Esta usuaria únicamente puede acceder los lunes, miércoles y viernes por la tarde.

En este caso se cumplen los requisitos de horario pero no tiene acceso a la puerta 6 que se definió en un principio. Por lo que al introducir el PIN 90129012 nos debería de aceptar el código de seguridad pero indicarnos que no podemos acceder a esta puerta.



Fotografía 17: Introducción PIN. Usuaria Ana

Fotografía 18: Usuaria Ana

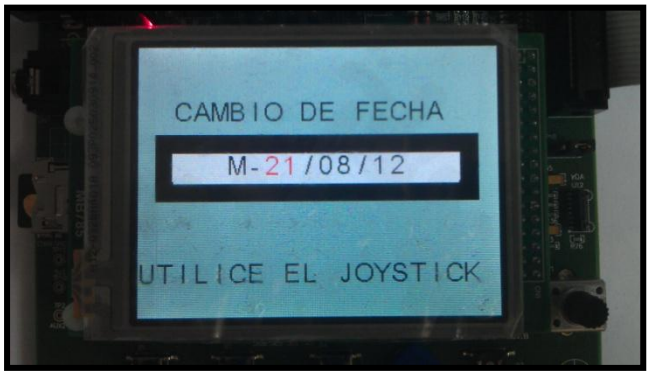
Fotografía 19: Puerta denegada a la usuaria Ana

Gracias a la opción de cambiar la puerta al principio, con simplemente realizar un Reset, el sistema dará la opción de cambiar la puerta y comprobar que si se modifica a la 10 por ejemplo, repitiendo el proceso anterior Ana tiene acceso.

Fotografía 20: Puerta abierta por la usuaria Ana

Comprobaremos ahora que el sistema detecta los permisos de fecha y hora y si éstos son incorrectos, no permite el acceso.

Primero modificaremos la fecha sin variar la hora. Sumamos un día más a la fecha que teníamos antes (lunes 20 de agosto) para que sea martes.



Fotografía 21: Modo cambio de fecha

Recordamos que Ana no tiene acceso los martes por lo que el sistema no debería abrirle la puerta.

En efecto, nos indica que Ana no tiene permiso para acceder a la zona el día que lo estamos intentando.

Si volvemos a poner el lunes como fecha y devolvemos la hora a la que en un principio teníamos, las 14:15 horas. El sistema comprobará que Ana no tiene permisos para acceder en este horario y avisará con el mensaje de la Fotografía 23.

Al realizar estas comprobaciones incluyendo el Reset de la placa comprobamos que el sistema escribe y lee correctamente los registros de Backup que se utilizaban para mantener la fecha.

Fotografía 22: Día no permitido a la usuaria Ana

Por último sólo queda verificar que el sistema detecta si se introduce mal el código PIN de la tarjeta. En ese caso, no dejará continuar hasta que se introduzca bien, o en el caso de sobrepasar el número de intentos posibles (3) nos obligará a retirar la tarjeta.

Para estas últimas comprobaciones se utilizará el usuario que restante, Mario, cuyo código PIN es 12341234.

Al introducir la tarjeta y teclear mal el código PIN salta el siguiente mensaje. Si continuamos introduciéndolo mal, al llegar a 3 intentos erróneos, el sistema nos obliga a sacar la tarjeta.

Fotografía 23: Horario no permitido a la usuaria Ana

Fotografía 24: Código PIN no válido

Fotografía 25: Retire la tarjeta

Finalizamos este capítulo indicando que gracias a los métodos de pruebas se ha conseguido un resultado final sólido y satisfactorio.

8 CONCULSIONES Y LÍNEAS FUTURAS

Para finalizar con el presente proyecto fin de carrera, se incluirán las conclusiones que se han obtenido al trabajar con este tipo de tecnología, así como el análisis de si se han cumplido los objetivos que se habían marcado y dónde han surgido más problemas.

También se incluirá un apartado sobre líneas futuras para demostrar la potencialidad de esta herramienta en este tipo de sistemas.

8.1 Conclusiones

En este trabajo se ha creado un prototipo de sistema de control de accesos donde los métodos de identificación han sido: en primer lugar y como método principal, la tarjeta inteligente y como secundario una clave introducida por el usuario.

Todos los objetivos que se marcaron en la adjudicación del proyecto se han cumplido satisfactoriamente, a excepción de la comunicación por puerto Ethernet por la imposibilidad física que presenta la placa para configurarse en modo Smartcard y modo Ethernet simultáneamente.

Adicionalmente, se han incluido ciertas mejoras en el diseño del software con el fin de demonstrar las grandes capacidades que presenta un sistema de control de accesos basado en microprocesadores.

Han existido dos etapas en el desarrollo de este prototipo que han sido con diferencia las partes donde se han encontrado mayores dificultades. La primera fue en el momento de adquirir la STM3210C-Eval por el poco conocimiento que se tenía sobre Microprocesadores.

Este aspecto obligó al alumno a esforzarse al máximo en estudiar y entender esta tecnología desarrollando programas independientes al objetivo final del proyecto.

La segunda etapa más costosa fue trabajar con las tarjetas inteligentes por el completo desconocimiento que se tenía sobre ellas. Al igual que en el caso anterior, el autor se vio obligado a encontrar, entender y aprenderse una gran cantidad de información para aprender en qué consiste esta tecnología y cómo se puede trabajar con ella.

A pesar de todo, podemos concluir que se ha logrado un buen prototipo que demuestra a su vez, las grandes posibilidades de esta herramienta, como el esfuerzo y trabajo dedicado en desarrollarlo.

8.2 Líneas Futuras

Se ha hablado constantemente de las grandes posibilidades que tiene este sistema y es en este apartado donde se especificarán algunas de ellas.

Inicialmente se propuso la comunicación por Ethernet para comunicarse con una base de datos y poder incluir como método secundario la introducción personalizada de usuario y contraseña.

Esta sería la primera mejora evidente que se podría desarrollar en este prototipo. La placa de desarrollo dispone de antenas de Bluetooth y Wifi para una comunicación inalámbrica, pudiendo establecer conexión con una base de datos.

Otro punto a mejorar sería la comunicación de todos los dispositivos con un sistema central de control. Esa comunicación también se podría hacer vía inalámbrica y permitiría el control de los dispositivos desde un único puesto. En caso de incendio, emergencia sanitaria, o cualquier otro aspecto importante se podría permitir el paso a cualquier zona en cuestión de segundos.

Un aspecto susceptible a la mejora es la solidez del diseño. En este proyecto se han incluido comprobaciones para que el sistema no se quede bloqueado en caso de que la comunicación con la tarjeta inteligente falle. Sin embargo, no se han estudiado posibles errores dentro de las tramas de la comunicación. Añadir determinados algoritmos de comprobación (comprobación del ATR, comprobación de todas las

posibilidades de los bytes de estado, etc.) conseguiría una mayor solidez en el diseño y una mayor seguridad en caso de falsificación de tarjetas.

La placa de desarrollo también dispone de otros periféricos muy útiles para sistemas de este tipo. El lector de tarjetas microSD ofrece un almacenamiento de gran capacidad, portable y fiable donde guardar registros, información o con una función más embellecedora, introducir imágenes en la interfaz del sistema.

Contiene además varios tipos de puertos con los que se puede incluir nuevos periféricos para por ejemplo, combinar más métodos de identificación.

Por último añadir que también dispone de salida de audio para conseguir un aspecto muy importante, mayor accesibilidad en personas discapacitadas.

9 ANEXOS

9.1 Anexo 1: Presupuesto

A continuación se va a realizar un estudio aproximado sobre el coste del proyecto incluyendo factores como:

- ✓ Horas de trabajo
- ✓ Licencia del entorno de desarrollo y placa de desarrollo STM3210C-EVAL
- ✓ Coste de amortización del equipo de desarrollo (por ejemplo, el ordenador de programación y el de edición de la documentación)
- ✓ Costes Personales

Las horas de trabajo se pueden dividir según la metodología explicada en el capítulo 4:

	HORAS
Fase 1 : Aprendizaje de Conceptos Básicos de Microprocesadores	40
Fase 2: Localización de Documentación	20
Fase 3: Comienzo del Proyecto sin Bibliotecas	80
Fase 4: Disponibilidad de Bibliotecas y Ejemplos	60
Fase 5: Tarjetas Inteligentes	40
Fase 6: Finalización del proyecto y pruebas	30
Fase 7: Redacción de la memoria	80
Total	350

Tabla 18: Reparto de horas de trabajo

La licencia y la placa de desarrollo STM3210C-EVAL cuestan un total de aproximadamente 250,00 €

Los **costes personales** se dividen según la siguiente tabla:

PUESTO	HORAS	Precio/Hora	Precio
Ingeniero			
Investigación	200	32	6.400,00
Desarrollo	150	25	3.750,00

Tabla 19: Costes Personales

Los **costes totales** ascienden a un total de: ***veintiún mil quinientos sesenta y ocho con veinticinco*** *€*

CONCEPTO	PRECIO (€)
Materiales	250,00
Costes Amortización de Equipos	20,25
Costes Personales	10.150,00
Costes Indirectos (15%)	1.522,50
Subtotal	11.942,75
IVA (21%)	2.507,98
TOTAL	14.450,73

Tabla 20: Costes totales del proyecto

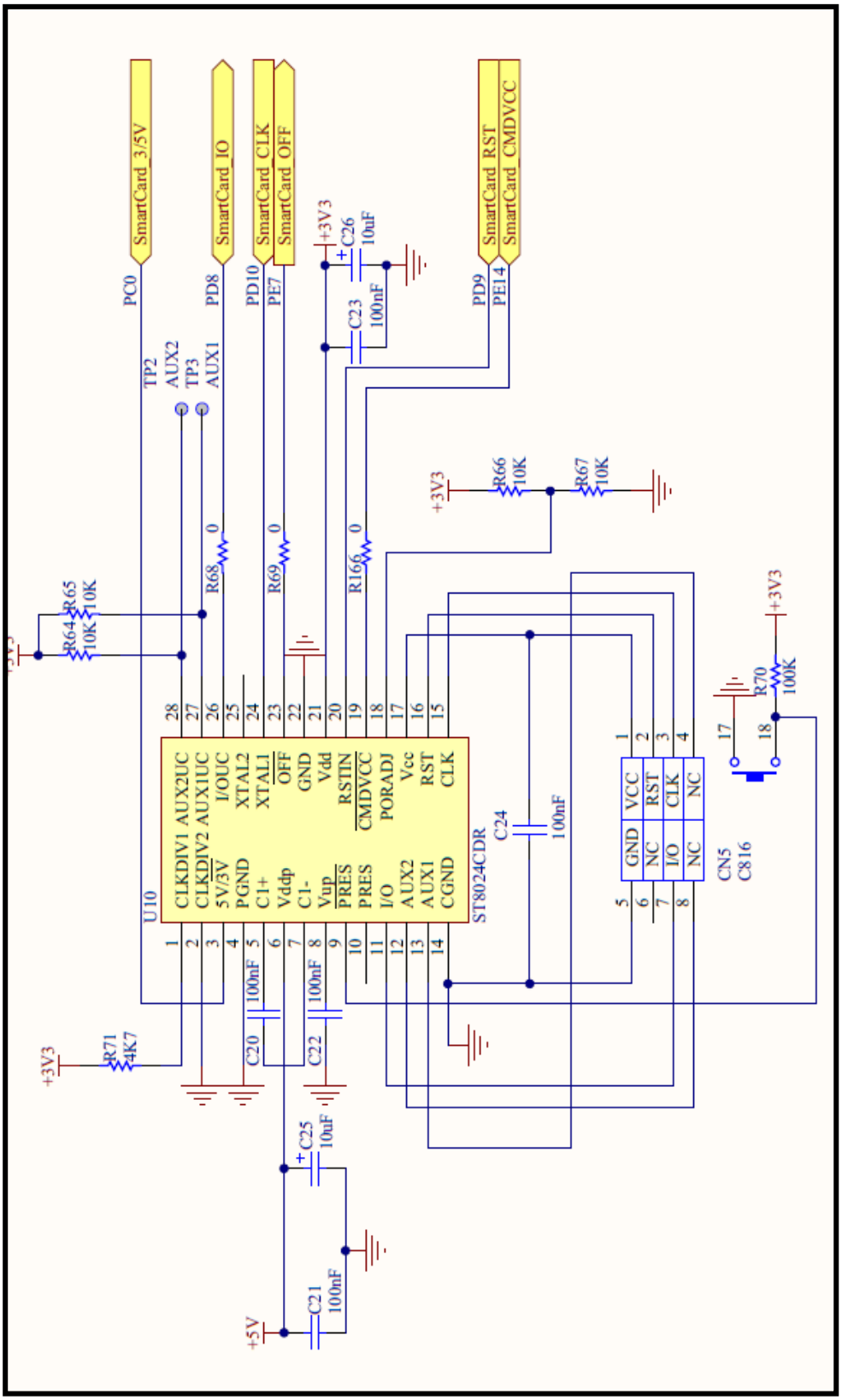
Leganés, 1 de septiembre del 2012

Fdo:

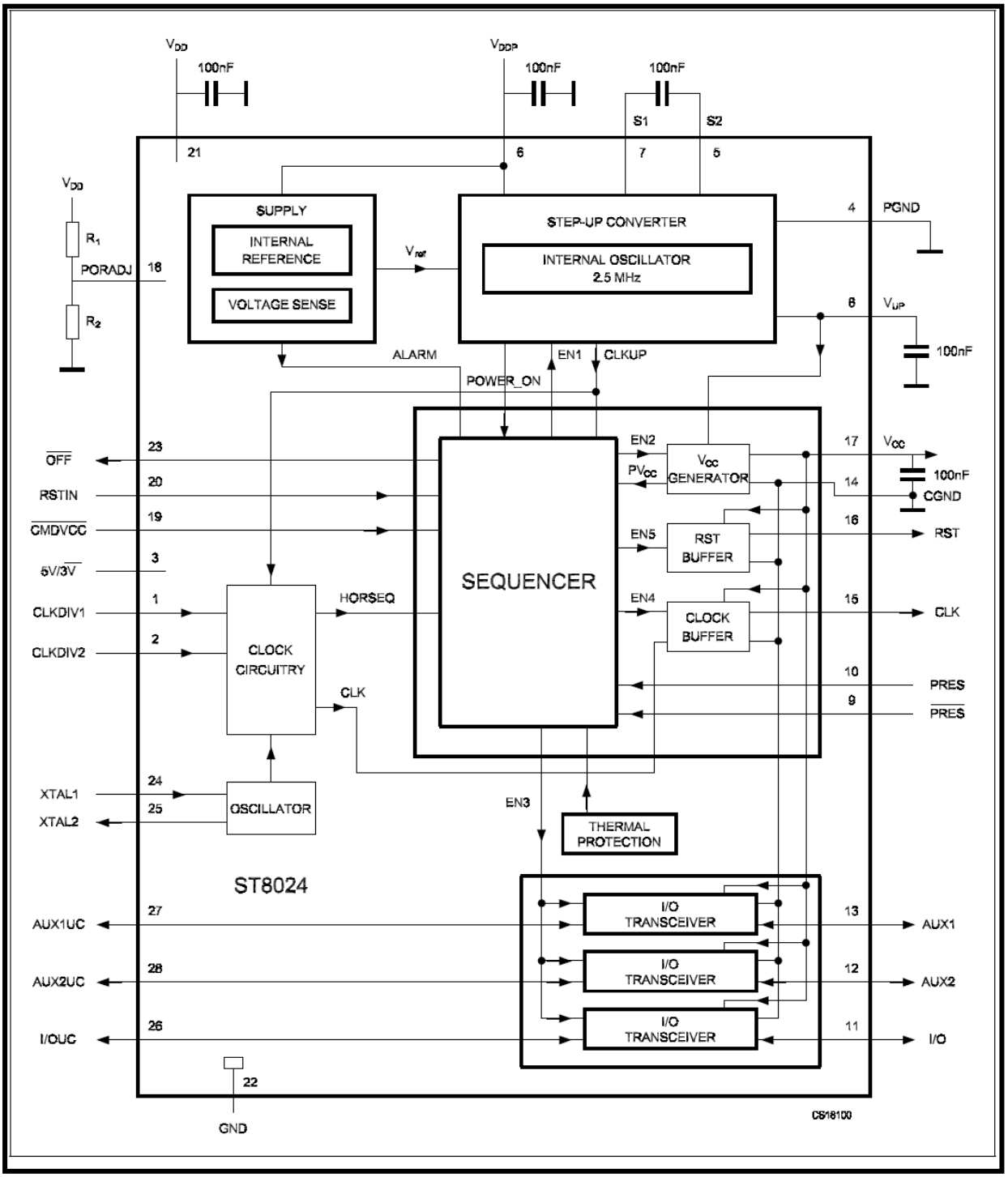
El ingeniero

9.2 Anexo 2: Esquemáticos importantes

9.2.1 SmartCard



9.2.2 ST8024


VDD
100nF
VDDP
100nF
100nF
S1
S2
21
6
7
5
VDD
R1
PORADJ
18
R2
SUPPLY
INTERNAL REFERENCE
VOLTAGE SENSE
Vref
STEP-UP CONVERTER
INTERNAL OSCILLATOR 2.5 MHz
4
PGND
8
VUP
100nF
ALARM
EN1
CLKUP
POWER_ON
OFF
23
RSTIN
20
CMDVCC
19
5V/3V
3
CLKDIV1
1
CLKDIV2
2
CLOCK CIRCUITRY
HORSEQ
CLK
SEQUENCER
EN2
PVCC
VCC GENERATOR
17
VCC
100nF
14
CGND
EN5
RST BUFFER
16
RST
EN4
CLOCK BUFFER
15
CLK
10
PRES
9
PRES
XTAL1
24
XTAL2
25
OSCILLATOR
EN3
THERMAL PROTECTION
ST8024
AUX1UC
27
AUX2UC
28
I/OUC
26
I/O TRANSCEIVER
I/O TRANSCEIVER
I/O TRANSCEIVER
13
AUX1
12
AUX2
11
I/O
22
GND
CS16100

9.3 Anexo 3: Código Total del diseño

9.3.1 Configuracion.c

9.3.2 Configuracion.h

9.3.3 Main.c

9.3.4 Main.h

9.3.5 Funciones.c

9.3.6 Funciones.h

9.3.7 Tarjeta.c

9.3.8 Tarjeta.h

10 BIBLIOGRAFÍA

[1] Honeywell. Congreso Seguridad Hospitalaria. Abril 2009 **www.seguridadhospitalaria.org/carlos_camacho.ppt .** Último acceso (19/07/2012)

[2] Wikiversity. **http://es.wikiversity.org/wiki/Ingenier%C3%ADa_de_microcontroladores .** Último acceso (21/07/2012)

[3] Raúl Sánchez Reíllo, José Carlos Acedo Jiménez, David Cerezo Quesada, Xoan R. Rodríguez Lorenzo. "La Tecnología de las Tarjetas Inteligentes". (1999)

[4] Fábrica Nacional de Moneda y Timbre. "Manual de Uso Tarjeta FNMT WG10 Versión 2.0". (2006)

[5] STMicroelectronics. "**UM0600:** User manual STM3210C-EVAL evaluation board". (2011). Disponible en: **http://www.st.com/internet/evalboard/product/217965.jsp**

[6] Página oficial de STMicroelectronics. **http://www.st.com/internet/evalboard/product/217965.jsp** . Último acceso (6/08/2012)

[7] Wikipedia. **http://en.wikipedia.org/wiki/ARM_architecture** . Último acceso (20/07/2012)

[8] Apuntes Universidad Carlos III de Madrid: Mario Acevedo Aguilar "Apuntes Microprocesadores".

[9]STMicroelectronics. "**RM0008:** Reference manual STM32F107VC". (2011) Disponible en: **http://www.st.com/internet/mcu/product/221020.jsp**

[10] Paul Ockenden "Capacitive or resistive: what's the best type of touchscreen?"**.** **http://www.pcpro.co.uk/realworld/357325/capacitive-or-resistive-whats-the-best-type-of-touchscreen** . Último acceso (23/07/2012)

[11] Elaboración propia a partir de**: http://en.wikipedia.org/wiki/TFT_LCD** y **http://www.ercservice.com/learning/what-is-tft-lcd.html** (Electronics Repair Center). Último acceso (23/07/2012)

[12] Wikipedia. **http://es.wikipedia.org/wiki/Serial_Peripheral_Interface**. Último acceso (27/07/2012)

[13] Elaboración propia. Autor Mario Acevedo Aguilar.

[14] STMicroelectronics **. "AN2598**: Smartcard interface with the STM32F10x microcontrollers". (2012). Disponible en: **http://www.st.com/internet/evalboard/product/217965.jsp**

```
/**
  ******************************************************************************
  * @file    IOExpander/Configuracion.c
  * @author  Mario Acevedo Aguilar
  * @version V 2.0
  * @date    15/08/2010
  * @brief   Configuracion
  ****************************************************************************** **/

/* Includes ------------------------------------------------------------------*/

#include "main.h"
#include "Configuracion.h"
#include "Funciones.h"

/* Variables -----------------------------------------------------------------*/

__IO uint32_t TimingDelay;

unsigned int valorRTC = 0;
unsigned char editar_fecha=0;
unsigned char editar_hora=0;
unsigned char tiempo[20]="                    ";

unsigned int usegundo=0,dsegundo=0,uminuto=0,    // Variables para representar el RTC
             dminuto=0,uhora=0,dhora=0;

unsigned int uano=0,dano=0,umes=0,               // Variables para representar el RTC
             dmes=0,udia=0,ddia=0;

unsigned int day=1,month=9,year=2012;
unsigned char mascara=0x00;

USART_InitTypeDef USART_InitStructure;
USART_ClockInitTypeDef USART_ClockInitStructure;
ErrorStatus HSEStartUpStatus;


/* Funciones -----------------------------------------------------------------*/

/* Funciones para que el sistema espera un determinado tiempo 100 = 1s ----*/
void Delay(uint32_t nTime)
{
  TimingDelay = nTime;

  while(TimingDelay != 0);
}
void TimingDelay_Decrement(void)
{
  if (TimingDelay != 0x00)
  {
    TimingDelay--;
  }
}

/* Configuración del RCC  ------------------------------------------------------
--------*/
void RCC_Configuration(void)
{
    /* Reiniciamos los registros del RCC */
    RCC_DeInit();

    /* Habilitamos el HSE */
    RCC_HSEConfig(RCC_HSE_ON);

    /* Esperamos hasta que el HSE este listo */
    HSEStartUpStatus = RCC_WaitForHSEStartUp();
```

```
    if(HSEStartUpStatus == SUCCESS)
    {
        /* HCLK = SYSCLK */
        RCC_HCLKConfig(RCC_SYSCLK_Div1);

        /* PCLK2 = HCLK */
        RCC_PCLK2Config(RCC_HCLK_Div1);

        /* PCLK1 = HCLK/2 */
        RCC_PCLK1Config(RCC_HCLK_Div2);

        /* PLLCLK = 8MHz * 9 = 72 MHz */
        RCC_PLLConfig(RCC_PLLSource_PREDIV1, RCC_PLLMul_9);

        /* Habilitamos el PLL */
        RCC_PLLCmd(ENABLE);

        /* Esperamos hasta que el PLL este listo */
        while(RCC_GetFlagStatus(RCC_FLAG_PLLRDY) == RESET)
        {
        }

        /* Seleccionamos el PLL para el SYSCLK */
        RCC_SYSCLKConfig(RCC_SYSCLKSource_PLLCLK);

        /* Esperamos hasta que este listo */
        while(RCC_GetSYSCLKSource() != 0x08)
        {
        }
    }

    /* Habilitamos GPIO_3_5V, GPIOB, GPIO_CMDVCC, GPIO_RESET, GPIO_OFF and AFIO clocks
*/
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_3_5V | RCC_APB2Periph_GPIOD | RCC_APB2Periph_
RESET |
                           RCC_APB2Periph_CMDVCC | RCC_APB2Periph_OFF | RCC_APB2Periph_AF
IO, ENABLE);

    /* Enable USART3 clocks */
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_USART3, ENABLE);
}

/* Configuración de la USART3  ------------------------------------------------------
--------*/
void USART3_Configuration(void)
{

    /* USART Clock igual a 4.5MHz (PCLK1 = 36 MHZ / 8) */
    USART_SetPrescaler(USART3, 0x10);

    USART_ClockInitStructure.USART_Clock = USART_Clock_Enable;
    USART_ClockInit(USART3, &USART_ClockInitStructure);
    /* USART:
        - Baud Rate 9600
        - Long de palabra = 9 bits
        - Stop bits = 1.5
        - Paridad PAR
        - Habilitado Transmision y Recepcion
    */
    USART_InitStructure.USART_BaudRate = 9600;
    USART_InitStructure.USART_WordLength = USART_WordLength_9b;
    USART_InitStructure.USART_StopBits = USART_StopBits_1_5;
    USART_InitStructure.USART_Parity = USART_Parity_Even;
    USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;
    USART_Init(USART3, &USART_InitStructure);

    /* Habilitamos USART3 */
    USART_Cmd(USART3, ENABLE);
```

```
    /* Habilitamos la USART3 en modo Tarjeta Inteligente */
    USART_SmartCardCmd(USART3, ENABLE);

}

/* Configuración del RTC  ------------------------------------------------------------
--------*/
void RTC_Configuration(void)
    {

    /* Habilitamos los relojes del PWR y el BKP */
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_PWR | RCC_APB1Periph_BKP, ENABLE);

    /* Permitimos el acceso a los registros de Backup */
    PWR_BackupAccessCmd(ENABLE);

    /* Habilitamos el LSE */
    RCC_LSEConfig(RCC_LSE_ON);
    /* Esperamos hasta que el LSE este listo */
    while (RCC_GetFlagStatus(RCC_FLAG_LSERDY) == RESET)
    {}

    /* Seleccionamos el LSE para el RTCCLK */
    RCC_RTCCLKConfig(RCC_RTCCLKSource_LSE);

    /* Habilitamos el RTC Clock */
    RCC_RTCCLKCmd(ENABLE);

    /* Esperamos que los Registros de sincronicen */
    RTC_WaitForSynchro();

    /* Esperamos hasta que se haya escrito la última operación */
    RTC_WaitForLastTask();

    /* Habilitamos la interrupción por segundos */
    RTC_ITConfig(RTC_IT_SEC, ENABLE);

    /* Esperamos hasta que se haya escrito la última operación */
    RTC_WaitForLastTask();

    /* Seleccionamos el prescaler para obtener el periodo de 1 segundo */
    RTC_SetPrescaler(32767); /* Periodo RTC = RTCCLK/RTC_PR = (32.768 KHz)/(32767+1) */

    /* Esperamos hasta que se haya escrito la última operación */
    RTC_WaitForLastTask();
}

/* Configuración del GPIO  ------------------------------------------------------------
--------*/
void GPIO_Configuration(void)
{
    /* Configuro el remapeo para la USART3:
        PD8: TX
        PD9: RX
        PD10: CK
    */

    AFIO->MAPR |= (0x3<<4);

    /* Configuro los siguientes GPIO de la siguiente forma:

        - PD8: Modo Alternate Function Push-Pull. Remap a USART3_TX
        - PD9: Modo Output Push-Pull. Remap a RSTIN
        - PD10: Modo Output Push-Pull. Remap a USART3_CK
        - PC0: Modo Output Push-Pull. Remap a 5V SmartCard
        - PE7: Modo Input Floating. Remap a OFF (Detectar SmartCard)
        - PE14: Modo Output Push-Pull. Remap a CMDVCC (Secuencia activacion)
```

```
            * Activo a nivel bajo   */

    /*- PD8: Modo Alternate Function Push-Pull. Remap a USART3_TX */

    GPIOD->CRH |=(0xD<<0);

    /*- PD9: Modo   Output Push-Pull. Remap a RSTIN */

    GPIOD->CRH &= ~(0xF<<4);
    GPIOD->CRH |=(0x1<<4);
    GPIOD->BSRR =(1<<25);

    /*- PD10: Modo Output Push-Pull. Remap a USART3_CK */

    GPIOD->CRH &= ~(0x0F<<8);
    GPIOD->CRH |= (0x0B<<8);

    /*- PC0: Modo Output Push-Pull. Remap a 5V SmartCard */

    GPIOC->CRL &= ~(0xF<<0);
    GPIOC->CRL |=(0x1<<0);
    GPIOC->BSRR =(1<<0);

    /*- PE7: Modo Input Floating. Remap a OFF (Detectar SmartCard) */

    GPIOE->CRL &= ~(0x0F<<28);
    GPIOE->CRL |=(0x4<<28);

    /*- PE14: Modo Output Push-Pull. Remap a CMDVCC (Secuencia activacion)
            * Activo a nivel bajo */

    GPIOE->CRH &= ~(0xF<<24);
    GPIOE->CRH |=(0x1<<24);
    GPIOE->BSRR =(1<<14);

    /*- Activaciones para asegurar que el inicio de la comunicacion con
        la tarjeta se hace en el momento requerido

            - PE14 = 1 Desactivación de el comienzo de la recepción de datos
            - PC0 = 1  Activación de la alimentación a 5V

    */

}

/* Configuración del NVIC  ----------------------------------------------------
--------*/
void NVIC_Configuration(void)
{
  NVIC_InitTypeDef NVIC_InitStructure;


  NVIC_PriorityGroupConfig(NVIC_PriorityGroup_1);

  /* Habilitamos la interrupción del RTC */
  NVIC_InitStructure.NVIC_IRQChannel = RTC_IRQn;
  NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 1;
  NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
  NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
  NVIC_Init(&NVIC_InitStructure);

}

/* Interrupción del RTC -------------------------------------------------------
--------*/
void RTC_IRQHandler(void)
{
    if (RTC_GetITStatus(RTC_IT_SEC) != RESET)
```

```
    {
        /* Limpiamos el Bit de interrupción */
        RTC_ClearITPendingBit(RTC_IT_SEC);

        /* Esperamos hasta que se haya escrito la última operación */
        RTC_WaitForLastTask();

        /* Si no estamos en el modo EDICION tomamos el valor del RTC, al salir, lo actu
alizamos */
        if((editar_hora==0)&&(editar_fecha==0))valorRTC=RTC_GetCounter();
        else if (editar_hora==1)RTC_SetCounter(valorRTC);

        /* Función Calendario para conocer la relación día,mes */
        Calendario();

        /* Representamos tanto la hora como la fecha */
        usegundo = ((valorRTC % 3600) % 60) % 10;
        dsegundo = ((valorRTC % 3600) % 60) / 10;

        uminuto = (((valorRTC % 3600) % 600) / 60);
        dminuto = ((valorRTC % 3600) / 600);

        uhora = (valorRTC % 36000) / 3600;
        dhora = (valorRTC / 36000);

        uano = (year%10);
        dano = (year%100)/10;

        umes = month%10;
        dmes =(month%100)/10;

        udia = day%10;
        ddia =(day%100)/10;


        tiempo[19]=('0' + usegundo);
        tiempo[18]=('0' + dsegundo);
        tiempo[17]='.';
        tiempo[16]=('0' + uminuto);
        tiempo[15]=('0' + dminuto);
        tiempo[14]='.';
        tiempo[13]=('0' + uhora);
        tiempo[12]=('0' + dhora);
        tiempo[11]=' ';
        tiempo[10]=' ';
        tiempo[9]=('0' + uano);
        tiempo[8]=('0' + dano);
        tiempo[7]='/';
        tiempo[6]=('0' + umes);
        tiempo[5]=('0' + dmes);
        tiempo[4]='/';
        tiempo[3]=('0' + udia);
        tiempo[2]=('0' + ddia);
        tiempo[1]='-';

        /* Según el día de la semana que sea, representamos el día y configuramos la má
scara */
        switch (Dia_Semana(day,month,year)){
            case 0: tiempo[0]='D';
                    mascara=0x40;
                    break;
            case 1: tiempo[0]='L';
                    mascara=0x01;
                    break;
            case 2: tiempo[0]='M';
                    mascara=0x02;
                    break;
            case 3: tiempo[0]='X';
```

```
                    mascara=0x04;
                    break;
            case 4: tiempo[0]='J';
                    mascara=0x08;
                    break;
            case 5: tiempo[0]='V';
                    mascara=0x10;
                    break;
            case 6: tiempo[0]='S';
                    mascara=0x20;
                    break;
        }

        /* Escribirmos en los registros de BackUp para sostener la fecha */
        BKP_WriteBackupRegister(BKP_DR1, year);
        BKP_WriteBackupRegister(BKP_DR2, month);
        BKP_WriteBackupRegister(BKP_DR3, day);

        /* Reset del RTC Counter si se ha llegado a 23:59:59 */
        if (RTC_GetCounter() >= 0x00015180)
        {
          RTC_SetCounter(0x0);
          day++;
        /* Esperamos hasta que se haya escrito la última operación */
            RTC_WaitForLastTask();
        }
    }
}

/* Interrupción de la EXTI WAKEUP ----------------------------------------------------
--------*/
void EXTI0_IRQHandler(void)
{
  if(EXTI_GetITStatus(WAKEUP_BUTTON_EXTI_LINE) != RESET)
  {
    bajoconsumo=0;
    EXTI_ClearITPendingBit(WAKEUP_BUTTON_EXTI_LINE);

  }
}

/* Interrupción de la EXTI TAMPER ----------------------------------------------------
--------*/
void EXTI15_10_IRQHandler(void)
{
    static int pos = 0;

    if(EXTI_GetITStatus(TAMPER_BUTTON_EXTI_LINE) != RESET)
    {
        bajoconsumo=0;
        if(modobajoconsumo==0){

        /* Editamos HORA */
            if (pos==0){
                editar_hora=1;
                editar_fecha=0;
                pos=1;
            }
        /* Editamos FECHA */
            else if (pos==1){
            editar_hora=0;
            editar_fecha=1;
            pos=0;
            }

        }
    EXTI_ClearITPendingBit(TAMPER_BUTTON_EXTI_LINE);
    }
```

```
}
/* Calculamos el dia de la semana segun la formula de Zeller ------------------------
--------*/
int Dia_Semana(unsigned int day, unsigned int month, unsigned int year)
{
    unsigned int mz,yz,cz;
    unsigned int ds;

    /* Lo primero es preparar los datos para aplicar la fórmula de Zeller */

    if (month<3) mz=month+10;
    else mz=month-2;

    if (mz>10) year--;

    yz=year%100;
    cz=year/100;

    /* Lo último, aplicamos la fórumla de Zeller para calcular el dia de la semana

    ds = day + [(13*mz - 1)/5] + dz + [dz/4] + [cz/4] - 2*cz

    Siendo:

        day = dia del mes
        mz = numero del mes, tomando la siguiente referencia: Mar=1, ..., Dic=10, Ene=1
1, Feb=12
        dz = Las ultimas dos cifras del año. Para Enero y Febrero se utiliza el año ant
erior
        cz = Las dos primeras cifras del año
    */

    ds = ((day + ((13*mz - 1)/5) + yz + (yz/4) + (cz/4) - 2*cz)%7);
    return ds;
}

/* Funcion para definir los dias que tiene cada mes ----------------------------------
--------*/
void Calendario(void)
{
    /* Lo primero que hacemos es comprobar si el año es bisiesto */

    unsigned int bisiesto;

    if (year%100==0){
        if (year%400==0) bisiesto=1;
        else bisiesto=0;
    }
    else if (year%4==0) bisiesto=1;
    else bisiesto=0;

    if ((month>=1)&&(month<=12)){

        switch (month){

            /* Enero, Marzo, Mayo, Julio, Agosto, Octube y Diciembre tienen 31 días */
            case 1:
            case 3:
            case 5:
            case 7:
            case 8:
            case 10:
            case 12:
                if (day<1)day=31;
                else if (day>31){
                    day=1;
                    if ((editar_hora==0)&&(editar_fecha==0)){
                        month ++;
```

```
                        if (month>12){
                        month=1;
                        year++;
                        }
                    }
                }
                break;
            /* Abril, Junio, Septiembre y Noviembre tienen 30 días */
            case 4:
            case 6:
            case 9:
            case 11:
                if (day<1)day=30;
                else if (day>30){
                    day=1;
                    if ((editar_hora==0)&&(editar_fecha==0)) month ++;
                }
                break;
            /* Febrero tendrá 29 o 28 según el año sea o no bisiesto */
            case 2:
                if (bisiesto==0){
                    if (day<1)day=28;
                    else if (day>28){
                    day=1;
                    if ((editar_hora==0)&&(editar_fecha==0)) month ++;
                    }
                }
                else if (bisiesto==1){
                    if (day<1)day=29;
                    else if (day>29){
                    day=1;
                    if ((editar_hora==0)&&(editar_fecha==0)) month ++;
                    }
                }
                break;
            default:
                break;

        }
    }
    else if (month<1)month=12;
    else if (month>12)month=1;
}
```

```
/**
  ******************************************************************************
  * @file    IOExpander/Configuracion.h
  * @author  Mario Acevedo Aguilar
  * @version V 2.0
  * @date    15/08/2010
  * @brief   Header Configuracion
  ****************************************************************************** **/

/* Definiciones Privadas ------------------------------------------------------*/

#define GPIO_3_5V                   GPIOC
#define SC_3_5V                     GPIO_Pin_0
#define GPIO_RESET                  GPIOD
#define SC_RESET                    GPIO_Pin_9
#define GPIO_CMDVCC                 GPIOE
#define SC_CMDVCC                   GPIO_Pin_14
#define GPIO_OFF                    GPIOE
#define SC_OFF                      GPIO_Pin_7
#define GPIO_MCU_IO                 GPIOD
#define SC_MCU_IO                   GPIO_Pin_8

#define SC_EXTI                      EXTI_Line7
#define SC_EXTI_IRQ                  EXTI9_5_IRQn

#define RCC_APB2Periph_3_5V      RCC_APB2Periph_GPIOD
#define RCC_APB2Periph_RESET     RCC_APB2Periph_GPIOB
#define RCC_APB2Periph_CMDVCC    RCC_APB2Periph_GPIOE
#define RCC_APB2Periph_OFF     RCC_APB2Periph_GPIOE
#define SC_PortSource            GPIO_PortSourceGPIOE
#define SC_PinSource               GPIO_PinSource7

/* Variables ------------------------------------------------------------------*/

extern __IO uint32_t TimingDelay;
extern unsigned int valorRTC;
extern unsigned char editar_fecha;   // Indicadores del MODO EDICION FECHA/HORA
extern unsigned char editar_hora;
extern unsigned char tiempo[20];     // Array para mostrar por pantalla el tiempo

extern unsigned int day,month,year;

extern unsigned char mascara;        // Variable para COMPROBAR permisos de DIA


/* Constantes -----------------------------------------------------------------*/

/* El horario sin tarjeta es de 16:00 a 19:00 ---------------------------*/

static const unsigned int horario_entrada=57600;
static const unsigned int horario_salida=68400;

/* Funciones ------------------------------------------------------------------*/

void Delay(uint32_t nTime);
void TimingDelay_Decrement(void);
void RCC_Configuration(void);
void RTC_Configuration(void);
void USART3_Configuration(void);
void GPIO_Configuration(void);
void NVIC_Configuration(void);
void RTC_IRQHandler(void);
void EXTI0_IRQHandler(void);
void EXTI15_10_IRQHandler(void);
int Dia_Semana(unsigned int day, unsigned int month, unsigned int year);
void Calendario(void);
```

```
/**
  ******************************************************************************
  * @file    IOExpander/main.c
  * @author  Mario Acevedo Aguilar
  * @version V 2.0
  * @date    15/08/2010
  * @brief   Main Program
  ****************************************************************************** **/


/* Includes ------------------------------------------------------------------*/

#include "main.h"
#include "Configuracion.h"
#include "Tarjeta.h"
#include "Funciones.h"

/* Private typedef -----------------------------------------------------------*/

unsigned int opcion=20;          // Indica el estado actual y siguiente del sistema
unsigned int bajoconsumo=0;      // Contador para entrar en bajo consumo
unsigned int modobajoconsumo=0; // Modo de bajo consumo (Activado/Desactivado)
unsigned long timeout=0;         // Evita bloqueos

/* Private function prototypes -----------------------------------------------*/
/* Private functions ---------------------------------------------------------*/


int main(void)
{
    /* Variables de uso general ------------------------------------------------*/

    unsigned int CardInserted=0; // Definir si hay o no tarjeta insertada
    unsigned int i=0;             // Variable de uso general
    unsigned int pin_correcto=0; // Indicador de PIN Correcto/Incorrecto
    unsigned int columna=0;      // Indicador de columnas para presentar un carácter
    unsigned int intentos=0;     // Veces que se permite introducir mal el código PIN

    /* Variables para realizar el ciclo de penalización ------------------------*/

    unsigned int wait=0;             // Contador
    unsigned char temporizador[1];  // Para presentar cuenta por pantalla

    /* Inicialización del microcontrolador: */

    SystemInit();

    /* Configuramos el SysTick con un pulso cada 10 ms con el reloj igual a 9 MHz(HCLK/
8, default) */

    SysTick_Config(SystemFrequency / 100);


    /* Inicializa los LEDs de la placa de desarrollo */

    STM_EVAL_LEDInit(LED1);
    STM_EVAL_LEDInit(LED2);
    STM_EVAL_LEDInit(LED3);
    STM_EVAL_LEDInit(LED4);

    /* Configuramos los botones TAMPER y WAKEUP en modo Interrupción */

    STM_EVAL_PBInit(Button_TAMPER, Mode_EXTI);
    STM_EVAL_PBInit(Button_WAKEUP, Mode_EXTI);

    /* Inicializamos la pantalla LCD */

    STM3210C_LCD_Init();
```

```
    /* Borramos la pantalla LCD */

    LCD_Clear(White);

    /* Configuramos el IO Expander y comprobamos que lo ha hecho correctamente */

    if (IOE_Config() == IOE_OK);
    else
    {
        LCD_DisplayStringLine(Line3, " Configuracion      ");
        LCD_DisplayStringLine(Line4, "    erronea del     ");
        LCD_DisplayStringLine(Line5, "      IO Expander   ");
        LCD_DisplayStringLine(Line7, "  Reinicia la placa ");
        while(1);
    }

    /* Leemos los registros de Backup para definir el valor del día, mes y año */

    year=BKP_ReadBackupRegister(BKP_DR1);
    month=BKP_ReadBackupRegister(BKP_DR2);
    day=BKP_ReadBackupRegister(BKP_DR3);

    /**
    ******************************************************************************
    Una vez comprobado que todo se ha inicializado correctamente, procedemos a
    configurar el microcontrolador para que el Sistema de Control de Accesos
    funcione correctamente.
    ******************************************************************************
    **/

    /* Configuracion RCC ------------------------------------------------------*/
    /* Las caracteristicas de la configuracion son las siguientes:
        - HCLK = SYSCLK
        - PCLK2 = HCLK
        - PCLK1 = HCLK/2
        - PLLCLK = 8MHz * 9 = 72 MHz
        - Selección del PLL como reloj del sistema
        - Habilitación de los relojes de GPIO, AFIO y USART
    */
    RCC_Configuration();

    /* Configuración de la USART3 ---------------------------------------------*/
    /* Las caracteristicas de la configuracion son las siguientes:
        - Reloj USART: 4,5 MHz
        - Word Length = 9 Bits
        - 1.5 Stop Bit
        - Even parity
        - BaudRate = 9600 baud
        - Tx and Rx enabled
    */
    USART3_Configuration();

    /* Configuracion RTC ------------------------------------------------------*/
    /* Las caracteristicas de la configuracion son las siguientes:
        - Configurado a partir del LSE
        - Habilitada la interrupción por segundos: RTC Second IT
        - Periodo del RTC = 1 seg: RTC period = RTCCLK/RTC_PR = (32.768 KHz)/(Prescaler
+1)
        - Prescaler = 32767
    */
    RTC_Configuration();

    /* Configuracion GPIO -----------------------------------------------------*/
    /* Las caracteristicas de la configuracion son las siguientes:
        - Para USART3:
            - PD8: Modo Alternate Function Push-Pull. Remap a USART3_TX
            - PD9: Modo Output Push-Pull. Remap a RSTIN
```

```
            - PD10: Modo Output Push-Pull. Remap a USART3_CK
            - PC0: Modo Output Push-Pull. Remap a 5V SmartCard
            - PE7: Modo Input Floating. Remap a OFF (Detectar SmartCard)
            - PE14: Modo Output Push-Pull. Remap a CMDVCC (Secuencia activacion)
                * Activo a nivel bajo *
    */
    GPIO_Configuration();


    /* Configuracion NVIC ----------------------------------------------------*/
    /* Las caracteristicas de la configuracion son las siguientes:
        - Interrupción RTC: Prioridad ALTA
    */
    NVIC_Configuration();

    /**
    ******************************************************************************
    Añadimos las funciones para poder definir la puerta a la que se tendrá acceso
    y la contraseña que hay que introducir si no se dispone de la tarjeta.
    ******************************************************************************
    **/

    /* Primero mostramos la pantalla para indicar la puerta */

    Pantalla(Blue,White);
    Menu();
    LCD_DisplayStringLine(Line2,"PUERTA:             ");
    opcion=10;

    /* Hasta que no incluyamos una puerta no avanzamos */

    while(opcion<=12){
        Comprobar_Pantalla();
        LCD_SetBackColor(White);
        LCD_SetTextColor(Black);
        LCD_DisplayStringLine(Line3,user);
        LCD_SetBackColor(Blue);
        LCD_SetTextColor(White);
    }

    /* Una vez que se han guardado los datos de la puerta, reiniciamos los valores
    y mostramos la pantalla para introducir la contraseña (password) */

    Reinicio_Datos();
    Pantalla(Blue,White);
    Menu();

    /* Hasta que no incluyamos una contraseña no avanzamos */

    while(opcion>=20){
        Comprobar_Pantalla();
        LCD_SetBackColor(White);
        LCD_SetTextColor(Black);
        LCD_DisplayStringLine(Line3,user);
        LCD_SetBackColor(Blue);
        LCD_SetTextColor(White);
    }

    /* Una vez que se han guardado los datos de contraseña, reiniciamos los valores
    y mostramos la presentación de entrada */

    Reinicio_Datos();
    opcion=0;
    Pantalla(Blue,White);

    Mostrar_Mensaje_Presentacion();

 /**
```

```
  ********************************************************************************
  MAIN: Este será el bucle principal del sistema. Se divide en dos partes:
    - Identificación SIN TARJETA. La interfaz de comunicación es la PANTALLA
    - Identificación CON TARJETA. La interfaz de comunicación es la TARJETA
  ********************************************************************************
 **/



while(1){

    /* Primero comprobamos si se activa el modo de bajo consumo ------------------*/

    bajoconsumo++;
    if (bajoconsumo >= 100){ // Equivale a 15 segundos (APROX.) de inactividad

        bajoconsumo=0;
        modobajoconsumo=1;
    /* Mostramos la pantalla del modo de bajo consumo */
        Pantalla(Black,White);
        LCD_DisplayStringLine(Line3, "     PULSE EN      ");
        LCD_DisplayStringLine(Line4, "  CUALQUIER BOTON  ");
        LCD_DisplayStringLine(Line5, "   PARA COMENZAR   ");
    /* Entramos en STOP MODE a la espera de que se de una INT por EXTI */
        PWR_EnterSTOPMode(PWR_Regulator_LowPower,PWR_STOPEntry_WFI);
    /* Al salir reconfiguramos el RCC porque varios registros han cambiado */
        RCC_Configuration();
        modobajoconsumo=0;
        Pantalla(Blue,White);
        opcion=0;

    }

    /* Tarjeta NO insertada -------------------------------------------------------*/

    if (CardInserted == 0){

    /* Si se introduce la tarjeta, saldremos de esta opción */
        if ((GPIOE->IDR&0x80)!=0) CardInserted=1;
    /* Bucle principal. Mientras que no estemos en modificar fecha/hora */
        if ((editar_hora==0)&&(editar_fecha==0)){
            LCD_DisplayStringLine(Line0,tiempo);
            Comprobar_Pantalla();
            switch(opcion){

                /* Pantalla Inicio*/
                case 0:
                    Inicio();
                    Reinicio_Datos();
                    opcion=1;
                    break;
                /* Pantalla Contraseña*/
                case 2:
                    Menu();
                    opcion=3;
                    break;
                /* Mostrar Teclado*/
                case 4:
                    Teclado();
                    opcion=5;
                    break;
                /* Representar datos introducidos*/
                case 5:
                    LCD_DisplayStringLine(Line0,tiempo);
                    LCD_SetBackColor(White);
                    LCD_SetTextColor(Black);
                    LCD_DisplayStringLine(Line3,mpass);
                    LCD_SetBackColor(Blue);
```

```
                    LCD_SetTextColor(Green);
                    LCD_DisplayStringLine(Line5,letra);
                    LCD_SetTextColor(White);
                    for (i=0;i<20;i++){
                        letra[i]=' ';
                    }
                    break;
                /* Comprobar datos introducidos*/
                case 6:
                    opcion=Comprobar_Clave();
                    Delay(100);
                    Pantalla(Black,White);
                    break;
                /* Penalización por nº de intentos */
                case 7:
                    if (wait<9){
                        Delay(100);
                        wait++;
                        temporizador[0]=wait +'0';
                        LCD_DisplayStringLine(Line9,temporizador);
                    }
                    else{
                        opcion=0;
                        wait=0;
                        Pantalla(Blue,White);
                    }
                    break;
                /* Comprobar horario actual con permiso sin tarjeta */
                case 8:
                    opcion=Comprobar_Horario();
                    Delay(200);
                    Pantalla(Black,White);
                    break;
                /* Cualquier otra opcion no hace nada*/
                default:
                    break;

            }
        }

    /* Se ha activado la edición de Fecha/Hora */

        else if (editar_hora==1){
            Pantalla_Hora();
            while(editar_hora==1)Editar_Hora();
        }
        else if (editar_fecha==1){
            Pantalla_Fecha();
            while(editar_fecha==1)Editar_Fecha();
        }

    }

 /**
  ********************************************************************************
  TARJETA DETECTADA: Recibimos la informacion del usuario UNA VEZ y la comparamos
  con los datos que tenemos en el archivo.

  Para vovler a leer datos sera necesario sacar y meter la misma u otra tarjeta.
  ********************************************************************************
 **/

    else if (CardInserted == 1){

        bajoconsumo=0;
        intentos=0;
        pin_correcto=0;
        Reinicio_Datos();
```

```
        Pantalla(Blue,White);
        LCD_DisplayStringLine(Line3, " TARJETA DETECTADA ");
        LCD_DisplayStringLine(Line4, "     POR FAVOR     ");
        LCD_DisplayStringLine(Line5, "       ESPERE      ");

     /* ACTIVACIÓN de la TARJETA mediante RESET FRÍO ----------------------------*/

        GPIO_ResetBits(GPIO_CMDVCC, SC_CMDVCC);
        GPIO_ResetBits(GPIO_RESET, SC_RESET);
        GPIO_SetBits(GPIO_RESET, SC_RESET);

     /* RECEPCIÓN del ATR -------------------------------------------------------*/

        Recepcion_ATR();
        Ciclo_Espera(100000);
    /* ABRIR ROOT --------------------------------------------------------------*/

        long_respuesta = 0;
        EnviarComando (APDU_ROOT, 9, SW_Buffer, &long_respuesta, &SW1, &SW2);
    /* ABRIR ACCESOS -----------------------------------------------------------*/
        long_respuesta = 0;
        EnviarComando (APDU_ACCESOS, 12, SW_Buffer, &long_respuesta, &SW1, &SW2);
     /* VERIFY ------------------------------------------------------------------*/
        while((pin_correcto==0)&&(intentos<3)&&(timeout<0xFFFFFFF0)){
        /* Permanecemos aquí hasta que se introduzca bien el PIN o se agoten los intent
os */
            Menu_PIN();
            opcion=30;
            while (opcion>=30){
                /* Si se saca la tarjeta en medio del proceso vuelve al principio */
                if ((GPIOE->IDR&0x80)==0){

                    /* REINICIAMOS todos lo valores ---------------------------------*
/

                    Pantalla(Black,White);
                    Reinicio_Datos_Tarjeta();
                    Reinicio_Datos();
                    CardInserted=0;
                    bajoconsumo=0;
                    opcion=0;

                     /* APAGAMOS todos los LEDs --------------------------------------*
/

                    GPIOD->BSRR |= (1<<23);
                    GPIOD->BSRR |= (1<<29);
                    GPIOD->BSRR |= (1<<19);
                    GPIOD->BSRR |= (1<<20);
                    break;
                }
                /* Introducción del PIN */
                Comprobar_Pantalla();
                LCD_SetBackColor(White);
                LCD_SetTextColor(Black);
                LCD_DisplayStringLine(Line3,mpass);
                LCD_SetBackColor(Blue);
                LCD_SetTextColor(White);

            }
            /* Comprobamos el PIN, siempre que la tarjeta esté insertada */
            if (CardInserted == 1){
                long_respuesta = 0;
                EnviarComando (APDU_VERIFY, 13, SW_Buffer, &long_respuesta, &SW1, &SW2)
;
                /* Comprobamos si el PIN es correcto con SW1 y SW2 */
                if ((SW1==0x90)&&(SW2==0x00)){
                pin_correcto=1;
```

```
                intentos=0;
                GPIOD->BSRR |= (1<<7);
                }
                else {
                    /* Si no es valido, volvemos a pedirlo (siempre que no se agoten lo
s intentos) */
                    opcion=30;
                    Pantalla_No_Valido();
                    LCD_DisplayStringLine(Line7,"CODIGO PIN NO VALIDO");
                    Reinicio_Datos();
                    Delay(100);
                    Pantalla(Black,White);
                    intentos++;
                    if (intentos==3) LCD_DisplayStringLine(Line7," RETIRE LA TARJETA  "
);
                }
            }
            else break;

        }
        /* Leemos los ficheros si se ha introducido bien el PIN y la tarjeta continua i
nsertada */
        if ((CardInserted == 1)&&(intentos<3)&&(timeout<0xFFFFFFF0)){

         /* READ BINARY 1 ------------------------------------------------------------*
/
            long_respuesta = 0x14;
            EnviarComando (APDU_READ1, 5, SW_Buffer, &long_respuesta, &SW1, &SW2);


         /* IMPRIMIMOS EL NOMBRE DEL USUARIO (max. 10 carácteres) --------------------*
/
            Pantalla(Blue,White);
            LCD_DisplayStringLine(Line2,"Usuario/a:          ");
            i=0;
            columna=223;

            while (i < 10)
            {
                LCD_DisplayChar(Line4, columna, SW_Buffer[i]);
                columna -= 16;
                i++;
                Delay(15);
            }

        /* READ BINARY 2 ------------------------------------------------------------*/
            long_respuesta = 0x0A;
            EnviarComando (APDU_READ2, 5, SW_Buffer, &long_respuesta, &SW1, &SW2);

        /* COMPROBACION DE LOS DATOS ------------------------------------------------*/
            Comprobacion(SW_Buffer);
        }

     /* ESPERAR a SACAR la TARJETA -----------------------------------------------*/
        Delay(200);
        Pantalla(Black,White);
        LCD_DisplayStringLine(Line7," RETIRE LA TARJETA  ");
        while (CardInserted==1){
            if ((GPIOE->IDR&0x80)==0){

         /* REINICIAMOS todos lo valores ---------------------------------------*/

                Pantalla(Black,White);
                Reinicio_Datos_Tarjeta();
                Reinicio_Datos();
                CardInserted=0;
                bajoconsumo=0;
                opcion=0;
```

```
                timeout=0;

         /* APAGAMOS todos los LEDs ---------------------------------------------*/

                GPIOD->BSRR |= (1<<23);
                GPIOD->BSRR |= (1<<29);
                GPIOD->BSRR |= (1<<19);
                GPIOD->BSRR |= (1<<20);
            }
        }
    }
}
}

#ifdef  USE_FULL_ASSERT


/**
  * @brief  Reports the name of the source file and the source line number
  *   where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t* file, uint32_t line)
{
  /* User can add his own implementation to report the file name and line number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */

   LCD_DisplayStringLine(Line0, "assert_param error!!");

  /* Infinite loop */
  while (1)
  {
  }
}
#endif

/****************** (C) COPYRIGHT 2009 STMicroelectronics *****END OF FILE****/
```

```
/**
  ******************************************************************************
  * @file    IOExpander/main.h
  * @author  Mario Acevedo Aguilar
  * @version V 2.0
  * @date    15/08/2010
  * @brief   Header Main Program
  ****************************************************************************** **/

/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __MAIN_H
#define __MAIN_H

/* Includes ------------------------------------------------------------------*/
#include "stm32_eval.h"
#include "stm3210c_eval_lcd.h"
#include "stm3210c_eval_ioe.h"
#include "stm32f10x_bkp.h"
#include "stm32f10x_rtc.h"
#include "stm32f10x_pwr.h"

/* Variables que se utilizan fuera del main ----------------------------------*/
extern unsigned int opcion;
extern unsigned int bajoconsumo;
extern unsigned int modobajoconsumo;
extern unsigned long timeout;

/* Exported constants --------------------------------------------------------*/
#define BUTTON_MODE_GPIO
//#define BUTTON_MODE_EXTI

#define IOE_POLLING_MODE
//#define IOE_INTERRUPT_MODE

#ifdef BUTTON_MODE_GPIO
  #define BUTTON_MODE    Mode_GPIO
#else
  #define BUTTON_MODE    Mode_EXTI
#endif

/* Exported macro ------------------------------------------------------------*/
/* Exported functions ------------------------------------------------------- */
void TimingDelay_Decrement(void);
void Delay(vu32 nTime);

#endif /* __MAIN_H */

/****************** (C) COPYRIGHT 2009 STMicroelectronics *****END OF FILE****/
```

```
/**
  ******************************************************************************
  * @file    IOExpander/Funciones.c
  * @author  Mario Acevedo Aguilar
  * @version V 2.0
  * @date    15/08/2010
  * @brief   Bloque de Funciones Utilizadas
  ****************************************************************************** **/

/* Includes ------------------------------------------------------------------*/

#include "main.h"
#include "Funciones.h"
#include "Configuracion.h"
#include "Tarjeta.h"

/* Variables -----------------------------------------------------------------*/


signed short int posicion=0; // Posicion de caracteres introducidos
TS_STATE* Estado_Pantalla;

/* Arrays/Punteros para representar o guardar informacion ----------------------*/


unsigned char clave[20]="default             ";
unsigned char PIN[8]={0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};
unsigned char user[20]="                    ";
unsigned char letra[20]="                    ";
unsigned char mpass[20]="                    ";
unsigned char* mens;
unsigned char* mens2;

/* Funciones -----------------------------------------------------------------*/


/* Inicio: Primera OPCION del Sistema: Introducir TARJETA/ Meter CONTRASEÑA-------*/
void Inicio(void)
{

    LCD_SetTextColor(White);
    LCD_SetBackColor(Blue);
    LCD_DisplayStringLine(Line1, "                    ");
    LCD_DisplayStringLine(Line2, "   SI HA OLVIDADO   ");
    LCD_DisplayStringLine(Line3, "     LA TARJETA     ");
    LCD_DisplayStringLine(Line4, "                    ");
    LCD_DisplayStringLine(Line5, "                    ");
    LCD_DisplayStringLine(Line6, "     PULSE AQUI     ");
    LCD_DisplayStringLine(Line7, "                    ");
    LCD_DisplayStringLine(Line8, "                    ");
    LCD_DisplayStringLine(Line9, "                    ");

    LCD_DrawRect(130,300,50,280);

    STM_EVAL_LEDOff(LED1);
    STM_EVAL_LEDOff(LED2);
    STM_EVAL_LEDOff(LED3);
    STM_EVAL_LEDOff(LED4);

}

/* Menu: Introducir CONTRASEÑA-----------------------------------------------------*/
void Menu(void)
{
    LCD_SetTextColor(White);
    LCD_SetBackColor(Blue);
    LCD_DisplayStringLine(Line1,"                    ");
    LCD_DisplayStringLine(Line2,"PASSWORD:           ");
```

```
    LCD_SetBackColor(White);
    LCD_DisplayStringLine(Line3,"                    ");
    LCD_SetBackColor(Blue);
    LCD_DisplayStringLine(Line4,"                    ");
    LCD_DisplayStringLine(Line5,"                    ");
    LCD_DisplayStringLine(Line6,"      ACEPTAR       ");
    LCD_DisplayStringLine(Line7,"                    ");
    LCD_DisplayStringLine(Line8,"                    ");
    LCD_DisplayStringLine(Line9,"                    ");


}

/* Menu: Introducir PIN ------------------------------------------------------*/
void Menu_PIN(void)
{
    LCD_SetTextColor(White);
    LCD_SetBackColor(Blue);
    LCD_DisplayStringLine(Line0,"                    ");
    LCD_DisplayStringLine(Line1,"                    ");
    LCD_DisplayStringLine(Line2,"INTRODUCE PIN:      ");
    LCD_SetBackColor(White);
    LCD_DisplayStringLine(Line3,"                    ");
    LCD_SetBackColor(Blue);
    LCD_DisplayStringLine(Line4,"                    ");
    LCD_DisplayStringLine(Line5,"                    ");
    LCD_DisplayStringLine(Line6,"      ACEPTAR       ");
    LCD_DisplayStringLine(Line7,"                    ");
    LCD_DisplayStringLine(Line8,"                    ");
    LCD_DisplayStringLine(Line9,"                    ");


}

/* Pantalla: Define el COLOR del FONDO y el COLOR de LETRA -----------------------*/
void Pantalla(__IO uint16_t Fondo,__IO uint16_t Letra)
{
    LCD_SetTextColor(Letra);
    LCD_SetBackColor(Fondo);

    LCD_DisplayStringLine(Line0, "                     ");
    LCD_DisplayStringLine(Line1, "                     ");
    LCD_DisplayStringLine(Line2, "                     ");
    LCD_DisplayStringLine(Line3, "                     ");
    LCD_DisplayStringLine(Line4, "                     ");
    LCD_DisplayStringLine(Line5, "                     ");
    LCD_DisplayStringLine(Line6, "                     ");
    LCD_DisplayStringLine(Line7, "                     ");
    LCD_DisplayStringLine(Line8, "                     ");
    LCD_DisplayStringLine(Line9, "                     ");
}

/* Mostrar_Mensaje_Presentacion: Mostrar MENSAJE de BIENVENIDA--------------------*/
void Mostrar_Mensaje_Presentacion(void)
{

    int rep=0;
    unsigned int refcolumn = 319;

    unsigned char* pres1="   Bienvenido al:   ";
    unsigned char* pres2=" SISTEMA DE CONTROL ";
    unsigned char* pres3=" DE ACCESOS BASADO  ";
    unsigned char* pres4="EN MICROPROCESADORES";
    unsigned char* pres5=" Autor:  ";
    unsigned char* pres6="Mario Acevedo   ";
    unsigned char* pres7="    Universidad     ";
    unsigned char* pres8=" Carlos III Madrid  ";
```

```
    unsigned char* dib1=">";
    unsigned char* dib3=" ";

    /* Configuramos la PANTALLA con FONDO NEGRO y LETRA VERDE --------------------*/

    Pantalla(Black,Green);

    /* Bienvenido al SISTEMA DE CONTROL DE ACCESOS BASADO EN MICROPROCESADORES----*/

    while ((*pres1 != 0) && (*pres2 != 0) && (*pres3 != 0) && (*pres4 != 0) && (rep < 2
0))
    {
        LCD_DisplayChar(Line1, refcolumn, *pres1);
        LCD_DisplayChar(Line4, refcolumn, *pres2);
        LCD_DisplayChar(Line5, refcolumn, *pres3);
        LCD_DisplayChar(Line6, refcolumn, *pres4);
        refcolumn -= 16;
        pres1++;
        pres2++;
        pres3++;
        pres4++;
        rep++;
        Delay(15);
    }

    refcolumn=319;
    rep=0;
    Delay(250);

    /* Autor ---------------------------------------------------------------------*/

    while ((*pres5 != 0) && (rep < 20))
    {
        LCD_DisplayChar(Line8, refcolumn, *pres5);
        refcolumn -= 16;
        pres5++;
        rep++;
        Delay(15);
    }

    refcolumn=319;
    rep=0;

    /* Mario Acevedo -------------------------------------------------------------*/

    while ((*pres6 != 0) && (rep < 20))
    {
        LCD_DisplayChar(Line9, refcolumn, *pres6);
        refcolumn -= 16;
        pres6++;
        rep++;
        Delay(15);
    }

    refcolumn=319;
    rep=0;
    Delay(200);

    /* Configuramos la PANTALLA con FONDO BLANCO y LETRA AZUL --------------------*/

    Pantalla(Grey,Blue);

    /* ENTRADA DE >>>>>>>>>>>>> --------------------------------------------------*/


    while ((*dib1 != 0) && (rep < 20))
    {
        LCD_DisplayChar(Line0, refcolumn, *dib1);
```

```
        LCD_DisplayChar(Line9, refcolumn, *dib1);
        refcolumn -= 16;
        rep++;
        Delay(15);
    }

    refcolumn=319;
    rep=0;
    Delay(20);

    /* Universidad Carlos III Madrid ------------------------------------------*/

    while ((*pres7 != 0) && (*pres8 != 0) && (*dib3 != 0) && (rep < 20))
    {
        LCD_DisplayChar(Line0, refcolumn, *dib3);
        LCD_DisplayChar(Line1, refcolumn, *pres7);
        LCD_DisplayChar(Line2, refcolumn, *pres8);
        LCD_DisplayChar(Line9, refcolumn, *dib3);
        refcolumn -= 16;
        pres7++;
        pres8++;
        rep++;
        Delay(15);
    }

    /* SIMBOLO de la UNIVERSIDAD ----------------------------------------------*/

        Delay(50);
        LCD_DisplayStringLine(Line5,"        UC3M        ");
        for(i=60;i<75;i++){
        LCD_DrawCircle(156, 157, i);
        }
        LCD_SetTextColor(Yellow);
        LCD_DrawRect(144, 192, 8, 72);
        LCD_DrawRect(184, 192, 8, 72);
        for(i=0;i<8;i++){
        LCD_DrawLine((144+i), 192, 71, Horizontal);
        LCD_DrawLine((184+i), 192, 71, Horizontal);
        }

        LCD_DrawRect(152, 176, 32, 8);
        LCD_DrawRect(152, 160, 32, 8);
        LCD_DrawRect(152, 144, 32, 8);
        for(i=0;i<8;i++){
        LCD_DrawLine(152, (176-i), 32, Vertical);
        LCD_DrawLine(152, (160-i), 32, Vertical);
        LCD_DrawLine(152, (144-i), 32, Vertical);
        }
        Delay(200);
        Pantalla(Blue,White);

}

/* EDITAR HORA ------------------------------------------------------------------*/
void Editar_Hora(void)
{
    static unsigned int cont=0;
    unsigned int a=0;
    unsigned int columna=0;

    static JOY_State_TypeDef JoyState = JOY_NONE;

    /* Muestro en ROJO el valor que estemos cambiando -----------------------------*/

    LCD_SetBackColor(Black);
    columna=144;
    for (a=12;a<20;a++)
    {
```

```
        if((cont==0)&&(a>=15)&&(a<=16)) LCD_SetTextColor(Red);
        else if ((cont==1)&&(a>=12)&&(a<=13))LCD_SetTextColor(Red);
        else LCD_SetTextColor(White);
        LCD_DisplayChar(Line4, columna, tiempo[a]);
        columna -= 16;

    }

    /* Segun la direccion que pulsemos, cambiaremos de valor o pasaremos a otro --*/

    JoyState = IOE_JoyStickGetState();
    Delay(30);
    switch (JoyState)
    {
      case JOY_NONE:
        break;
      case JOY_UP:
        if (cont==0)valorRTC=valorRTC+60;
        else if(cont==1)valorRTC=valorRTC+3600;
        bajoconsumo=0;
        break;
      case JOY_DOWN:
        if (cont==0)valorRTC=valorRTC-60;
        else if(cont==1)valorRTC=valorRTC-3600;
        bajoconsumo=0;
        break;
      case JOY_LEFT:
        cont++;
        bajoconsumo=0;
        break;
      case JOY_RIGHT:
        cont--;
        bajoconsumo=0;
        break;
      case JOY_CENTER:
        opcion=0;
        editar_hora=0;
        bajoconsumo=0;
        Pantalla(Black,White);
        break;
      default:
        break;
    }
    if(cont>=1){
        cont=1;
    }

    if(cont<=0){
        cont=0;
    }

    /* Si sobrepasamos la hora, ésta se reinicia ---------------------------------*/

    if (valorRTC>= 0x00015180) valorRTC=0;

}

/* PANTALLA para EDITAR HORA -------------------------------------------------------*/
void Pantalla_Hora(void)
{
    unsigned int a=0;

    Pantalla(Grey,Black);
    LCD_DisplayStringLine(Line1,"  CAMBIO");
    LCD_DisplayStringLine(Line2,"    DE");
    LCD_DisplayStringLine(Line3,"   HORA");
    LCD_DisplayStringLine(Line8,"UTILICE EL JOYSTICK");
```

```
    LCD_SetTextColor(White);
    for(a=1;a<72;a++){
        LCD_DrawCircle(108, 83, a);
    }
    LCD_SetTextColor(Black);
    for(a=71;a<81;a++){
        LCD_DrawCircle(108, 83, a);
    }
}

/* EDITAR FECHA ------------------------------------------------------------------*/
void Editar_Fecha(void)
{

    static unsigned int cont=0;
    unsigned int a=0;
    unsigned int columna=0;
    static JOY_State_TypeDef JoyState = JOY_NONE;

    LCD_SetBackColor(White);
    LCD_SetTextColor(Black);
    columna=240;
        for (a=0;a<10;a++)
        {
            if((cont==0)&&(a>=8)&&(a<=9)) LCD_SetTextColor(Red);
            else if ((cont==1)&&(a>=5)&&(a<=6))LCD_SetTextColor(Red);
            else if ((cont==2)&&(a>=2)&&(a<=3))LCD_SetTextColor(Red);
            else LCD_SetTextColor(Black);
            LCD_DisplayChar(Line4, columna, tiempo[a]);
            columna -= 16;
        }



    JoyState = IOE_JoyStickGetState();

    Delay(30);
    switch (JoyState)
    {
      case JOY_NONE:
        break;
      case JOY_UP:
        if (cont==0)year++;
        else if(cont==1)month++;
        else if(cont==2)day++;
        bajoconsumo=0;
        break;
      case JOY_DOWN:
        if (cont==0)year--;
        else if(cont==1)month--;
        else if(cont==2)day--;
        bajoconsumo=0;
        break;
      case JOY_LEFT:
        cont++;
        bajoconsumo=0;
        break;
      case JOY_RIGHT:
        cont--;
        bajoconsumo=0;
        break;
      case JOY_CENTER:
        editar_fecha=0;
        opcion=0;
        bajoconsumo=0;
        Pantalla(Black,White);
        break;
      default:
```

```
        break;
    }

    if(cont>=2){
        cont=2;
    }

    if(cont<=0){
        cont=0;
    }
    if (valorRTC>= 0x00015180) valorRTC=0;


}

/* PANTALLA para EDITAR FECHA ------------------------------------------------*/
void Pantalla_Fecha(void)
{
    unsigned int a=0;

    Pantalla(Grey,Black);
    LCD_DisplayStringLine(Line2,"  CAMBIO DE FECHA   ");
    LCD_DisplayStringLine(Line8,"UTILICE EL JOYSTICK");

    LCD_DrawRect(78, 304, 59, 288);
    for(a=0;a<59;a++){
    LCD_DrawLine((78+a), 303, 286, Horizontal);
    }
    LCD_SetTextColor(White);
    LCD_DrawRect(95, 289, 24, 256);
    for(a=0;a<26;a++){
        LCD_DrawLine((96+a), 288, 254, Horizontal);
    }

}

/* Comprobar la INTERACCION con la PANTALLA TACTIL ---------------------------*/
void Comprobar_Pantalla(void)
{
    unsigned int horizontal=0, vertical=0;
    unsigned int a=0;
    Estado_Pantalla = IOE_TS_GetState();
    Delay(40);
         if (Estado_Pantalla->TouchDetected)
        {
            horizontal = Estado_Pantalla->X;
            vertical = Estado_Pantalla->Y;
            bajoconsumo=0;

            switch(opcion){

                /* Pulsamos el boton PULSE AQUI ----------------------------------*/
                case 1:

                    if ((horizontal>20) && (horizontal<300)){
                        if ((vertical>130) && (vertical<180)){
                        opcion = 2;
                        }
                    }
                    break;
                case 3:
                /* Pulsamos para ESCRIBIR ----------------------------------------*/
                    if ((horizontal>0) && (horizontal<320)){
                        if ((vertical>72) && (vertical<96)){
                            opcion = 4;
                            Reinicio_Datos();
                            posicion=0;
                        }
```

```
                /* Pulsamos el boton ACEPTAR -------------------------------------*/
                        else if ((vertical>144) && (vertical<168)){
                            LCD_DisplayStringLine(Line6,"      ACEPTAR       ");
                            opcion=6;
                            }
                    }
                    break;
                case 5:
                /* Pulsamos fuera del TECLADO -------------------------------------*/
                    if (((vertical>0) && (vertical<144))){
                        if ((horizontal>0) && (horizontal<320)){

                        opcion = 3;
                        Ocultar_Teclado();
                       }
                    }
                /* Pulsamos el TECLADO para ESCRIBIR -----------------------------*/
                    else if ((vertical>144) && (vertical<240)){
                    Escribir(horizontal,vertical);
                    }
                    break;
                /* INTRODUCIMOS LA PUERTA AL PRINCIPIO DEL PROGRAMA --------------*/
                case 10:
                    if ((horizontal>0) && (horizontal<320)){
                        if ((vertical>72) && (vertical<96)){
                            opcion = 11;
                            posicion=0;
                            Teclado_PIN();
                        }
                        else if ((vertical>144) && (vertical<168)){
                            LCD_DisplayStringLine(Line6,"      ACEPTAR       ");
                            opcion=12;
                        }
                    }
                    break;
                case 11:
                    if (((vertical>0) && (vertical<144))){
                        if ((horizontal>0) && (horizontal<320)){
                        opcion=10;
                        Reinicio_Datos();
                        Ocultar_Teclado();
                       }
                    }
                    else if ((vertical>144) && (vertical<240)){
                    Escribir_PIN(horizontal,vertical);
                    }
                    break;
                /* GUARDAMOS LA PUERTA INTRODUCIDA -------------------------------*/
                case 12:
                    puerta=(unsigned int)(PIN[1]+(PIN[0]*10));
                    LCD_DisplayStringLine(Line4,"   DATOS GRABADOS  ");
                    Delay(50);
                    opcion=20;
                    break;
                /* INTRODUCIMOS LA CONTRASEÑA AL PRINCIPIO DEL PROGRAMA ----------*/
                case 20:
                    if ((horizontal>0) && (horizontal<320)){
                        if ((vertical>72) && (vertical<96)){
                            opcion = 21;
                            posicion=0;
                            Teclado();
                        }
                        else if ((vertical>144) && (vertical<168)){
                            LCD_DisplayStringLine(Line6,"      ACEPTAR       ");
                            opcion=22;
                        }
                    }
                    break;
```

```
                case 21:
                    if (((vertical>0) && (vertical<144))){
                        if ((horizontal>0) && (horizontal<320)){
                        opcion=20;
                        Reinicio_Datos();
                        Ocultar_Teclado();
                       }
                    }
                    else if ((vertical>144) && (vertical<240)){
                    Escribir(horizontal,vertical);
                    }
                    break;
                /* GUARDAMOS LA CONTRASEÑA INTRODUCIDA ---------------------------*/
                case 22:
                    for(a=0;a<20;a++){
                    clave[a]=user[a];
                    }
                    LCD_DisplayStringLine(Line4,"   DATOS GRABADOS  ");
                    Delay(50);
                    opcion=0;
                    break;
                /* INTRODUCIMOS el PIN de seguridad de la TARJETA ----------------*/
                case 30:
                    if ((horizontal>0) && (horizontal<320)){
                        if ((vertical>72) && (vertical<96)){
                            opcion = 31;
                            posicion=0;
                            Teclado_PIN();
                        }
                        else if ((vertical>144) && (vertical<168)){
                            LCD_DisplayStringLine(Line6,"      ACEPTAR       ");
                            opcion=32;
                        }
                    }
                    break;
                case 31:
                    if (((vertical>0) && (vertical<144))){
                        if ((horizontal>0) && (horizontal<320)){
                        opcion=30;
                        Reinicio_Datos();
                        Ocultar_Teclado();
                       }
                    }
                    else if ((vertical>144) && (vertical<240)){
                    Escribir_PIN(horizontal,vertical);
                    }
                    break;
                case 32:
                    for (a=0;a<8;a++){
                    APDU_VERIFY[a+5]=PIN[a];
                    }
                    Delay(50);
                    opcion=0;
                    break;
                default:
                    break;
            }
        }
}

/* Teclado: MUESTRA por PANTALLA la INTERFAZ de TECLADO ---------------------------*/
void Teclado(void)
{
  unsigned int a=0;

    LCD_SetBackColor(Grey);
    LCD_SetTextColor(Black);
```

```
    LCD_DisplayStringLine(Line6, "1 2 3 4 5 6 7 8 9 0 ");
    LCD_DisplayStringLine(Line7, "q w e r t y u i o p ");
    LCD_DisplayStringLine(Line8, "a s d f g h j k l <<");
    LCD_DisplayStringLine(Line9, "  z x c v b n m   OK");

    for(a=320;a>31;a=a-32){
        LCD_DrawRect(144,a,24,32);
        LCD_DrawRect(168,a,24,32);
        LCD_DrawRect(192,a,24,32);
        LCD_DrawRect(216,a,24,32);
    }

    LCD_SetBackColor(Blue);
    LCD_SetTextColor(White);
}

/* Ocultar_Teclado: ESCONDE la INTERFAZ de TECLADO ------------------------------*/
void Ocultar_Teclado(void)
{

LCD_SetBackColor(Blue);
LCD_SetTextColor(White);
LCD_DisplayStringLine(Line4, "                    ");
LCD_DisplayStringLine(Line5, "                    ");
LCD_DisplayStringLine(Line6, "      ACEPTAR       ");
LCD_DisplayStringLine(Line7, "                    ");
LCD_DisplayStringLine(Line8, "                    ");
LCD_DisplayStringLine(Line9, "                    ");


}

/* Escribir: Introducir CARACTERES mediante el CONTACTO con la PANTALLA TACTIL----*/
void Escribir(int horizontal,int vertical)
{
    /* Segun la POSICION y la MATRIZ del TECLADO escribimos un caracter u otro ---*/

    if ((posicion>=0)&&(posicion<20)){

        if ((vertical>=144) && (vertical<168)){

            if((horizontal<320) && (horizontal>=288)){
            user[posicion]='0';
            letra[18]='0';
            }
            else if((horizontal<288) && (horizontal>=256)){
            user[posicion]='9';
            letra[16]='9';
            }
            else if((horizontal<256) && (horizontal>=224)){
            user[posicion]='8';
            letra[14]='8';
            }
            else if((horizontal<224) && (horizontal>=192)){
            user[posicion]='7';
            letra[12]='7';
            }
            else if((horizontal<192) && (horizontal>=160)){
            user[posicion]='6';
            letra[10]='6';
            }
            else if((horizontal<160) && (horizontal>=128)){
            user[posicion]='5';
            letra[8]='5';
            }
            else if((horizontal<128) && (horizontal>=96)){
            user[posicion]='4';
            letra[6]='4';
```

```
            }
            else if((horizontal<96) && (horizontal>=64)){
            user[posicion]='3';
            letra[4]='3';
            }
            else if((horizontal<64) && (horizontal>=32)){
            user[posicion]='2';
            letra[2]='2';
            }
            else if((horizontal<32) && (horizontal>=0)){
            user[posicion]='1';
            letra[0]='1';
            }
        }
        else if ((vertical>=168) && (vertical<192)){
            if((horizontal<320) && (horizontal>=288)){
            user[posicion]='p';
            letra[18]='p';
            }
            else if((horizontal<288) && (horizontal>=256)){
            user[posicion]='o';
            letra[16]='o';
            }
            else if((horizontal<256) && (horizontal>=224)){
            user[posicion]='i';
            letra[14]='i';
            }
            else if((horizontal<224) && (horizontal>=192)){
            user[posicion]='u';
            letra[12]='u';
            }
            else if((horizontal<192) && (horizontal>=160)){
            user[posicion]='y';
            letra[10]='y';
            }
            else if((horizontal<160) && (horizontal>=128)){
            user[posicion]='t';
            letra[8]='t';
            }
            else if((horizontal<128) && (horizontal>=96)){
            user[posicion]='r';
            letra[6]='r';
            }
            else if((horizontal<96) && (horizontal>=64)){
            user[posicion]='e';
            letra[4]='e';
            }
            else if((horizontal<64) && (horizontal>=32)){
            user[posicion]='w';
            letra[2]='w';
            }
            else if((horizontal<32) && (horizontal>=0)){
            user[posicion]='q';
            letra[0]='q';
            }
        }
        else if ((vertical>=192) && (vertical<216)){

            if((horizontal<320) && (horizontal>=288)){
                user[posicion-1]=' ';
                mpass[posicion-1]=' ';
                posicion=posicion-2;
            letra[18]='<';
            letra[19]='<';
            }
            else if((horizontal<288) && (horizontal>=256)){
            user[posicion]='l';
            letra[16]='l';
```

```
            }
            else if((horizontal<256) && (horizontal>=224)){
            user[posicion]='k';
            letra[14]='k';
            }
            else if((horizontal<224) && (horizontal>=192)){
            user[posicion]='j';
            letra[12]='j';
            }
            else if((horizontal<192) && (horizontal>=160)){
            user[posicion]='h';
            letra[10]='h';
            }
            else if((horizontal<160) && (horizontal>=128)){
            user[posicion]='g';
            letra[8]='g';
            }
            else if((horizontal<128) && (horizontal>=96)){
            user[posicion]='f';
            letra[6]='f';
            }
            else if((horizontal<96) && (horizontal>=64)){
            user[posicion]='d';
            letra[4]='d';
            }
            else if((horizontal<64) && (horizontal>=32)){
            user[posicion]='s';
            letra[2]='s';
            }
            else if((horizontal<32) && (horizontal>=0)){
            user[posicion]='a';
            letra[0]='a';
            }
        }
        else if ((vertical>=216) && (vertical<240)){

            if((horizontal<320) && (horizontal>=288)){
            letra[18]='<';
            letra[19]='<';
            if(opcion<20)opcion=3;
            else opcion=20;
            Ocultar_Teclado();
            }
            else if((horizontal<288) && (horizontal>=256)){
            posicion--;
            }
            else if((horizontal<256) && (horizontal>=224)){
            user[posicion]='m';
            letra[14]='m';
            }
            else if((horizontal<224) && (horizontal>=192)){
            user[posicion]='n';
            letra[12]='n';
            }
            else if((horizontal<192) && (horizontal>=160)){
            user[posicion]='b';
            letra[10]='b';
            }
            else if((horizontal<160) && (horizontal>=128)){
            user[posicion]='v';
            letra[8]='v';
            }
            else if((horizontal<128) && (horizontal>=96)){
            user[posicion]='c';
            letra[6]='c';
            }
            else if((horizontal<96) && (horizontal>=64)){
            user[posicion]='x';
```

```
            letra[4]='x';
            }
            else if((horizontal<64) && (horizontal>=32)){
            user[posicion]='z';
            letra[3]='z';
            }
            else if((horizontal<32) && (horizontal>=0)){
            posicion--;
            }
        }
        Delay(40);
        if(posicion>=0){
        mpass[posicion]='*';
        posicion++;
        }

    }

    /* Si llegamos a los 20 caracteres, no nos dejará introducir más ------------*/

    else {

        if ((vertical>=192) && (vertical<216)){
            if((horizontal<320) && (horizontal>=288)){
                user[posicion-1]=' ';
                mpass[posicion-1]=' ';
                posicion--;
                letra[18]='<';
                LCD_DisplayStringLine(Line4,"                    ");
            }
        }
        else if ((vertical>=216) && (vertical<240)){

            if((horizontal<320) && (horizontal>=288)){
            if(opcion<20)opcion=3;
            else opcion=20;
            Ocultar_Teclado();
            }
        }
        else LCD_DisplayStringLine(Line4," MAX. 20 CARACTERES ");
    }

if ((posicion<=0)|(posicion>20)) posicion=0;
if (posicion==20) posicion=20;

}

/* COMPROBAR si la CONTRASEÑA introducida es VALIDA ------------------------------*/
int Comprobar_Clave(void)
{
    unsigned int a=0;
    unsigned int incorrecto=0;
    static unsigned int novalido = 0;

    for (a=0;a<20;a++){

        if ((user[a])!=(clave[a])) incorrecto=1;

    }
    if (incorrecto==1){
        Pantalla_No_Valido();
        mens=" DATOS NO CORRECTOS ";
        LCD_DisplayStringLine(Line7,mens);
        GPIOD->BSRR |= (1<<13);
        novalido++;

    }
    else if(incorrecto==0){
```

```
        Pantalla_Si_Valido();
        mens="  DATOS CORRECTOS   ";
        LCD_DisplayStringLine(Line7,mens);
        novalido=0;
        GPIOD->BSRR |= (1<<7);
    }


    /* Si sobrepasamos el número de intentos posibles, PENALIZACIÓN --------------*/

    if(novalido==0)opcion=8;
    else if(novalido==3){
        opcion=7;
        novalido=0;
        Pantalla_No_Valido();
        LCD_DisplayStringLine(Line7,"  PENALIZACION POR  ");
        LCD_DisplayStringLine(Line8," NUMERO DE INTENTOS ");

    }
    else opcion=0;

return opcion;
}

/* COMPROBAR si el HORARIO es VALIDO ----------------------------------------------*/
int Comprobar_Horario(void)
{
    static unsigned int novalido = 0;

    STM_EVAL_LEDOff(LED1);
    STM_EVAL_LEDOff(LED2);
    STM_EVAL_LEDOff(LED3);
    STM_EVAL_LEDOff(LED4);

    if ((valorRTC>=horario_entrada)&(valorRTC<=horario_salida)){
    Pantalla_Si_Valido();
    mens="   PUERTA ABIERTA   ";
    LCD_DisplayStringLine(Line7,mens);
    novalido=0;
    GPIOD->BSRR |= (1<<4);
    }
    else{
    novalido++;
    Pantalla_No_Valido();
    mens="     HORARIO NO     ";
    mens2="      ADMITIDO      ";
    LCD_DisplayStringLine(Line7,mens);
    LCD_DisplayStringLine(Line8,mens2);
    GPIOD->BSRR |= (1<<13);
    }


    if(novalido==0)opcion=0;
    else if(novalido==3){
        opcion=7;
        novalido=0;
        Pantalla_No_Valido();
        LCD_DisplayStringLine(Line7,"  PENALIZACION POR  ");
        LCD_DisplayStringLine(Line8," NUMERO DE INTENTOS ");
    }
    else opcion=0;


return opcion;
}

/* REINICIO de DATOS introducidos -------------------------------------------------*/
void Reinicio_Datos(void)
```

```
{
    unsigned int a = 0;

    for(a = 0 ; a < 8 ; a++){
        PIN[a] = 0x00;
    }
    for(a = 0 ; a < 20 ; a++){
    user[a]=0x20;
    mpass[a]=0x20;
    letra[a]=0x20;
    }
}

/* TECLADO para meter PIN TARJETA ------------------------------------------------*/
void Teclado_PIN(void)
{
   unsigned int a=0;

    LCD_SetBackColor(Grey);
    LCD_SetTextColor(Black);

    LCD_DisplayStringLine(Line6,"  1   2   3         ");
    LCD_DisplayStringLine(Line7,"  4   5   6    OK   ");
    LCD_DisplayStringLine(Line8,"  7   8   9         ");
    LCD_DisplayStringLine(Line9,"      0       Borrar");

    for(a=319;a>190;a=a-64){
        LCD_DrawRect(144,a,24,64);
        LCD_DrawRect(167,a,24,64);
        LCD_DrawRect(192,a,24,64);
        LCD_DrawRect(215,a,24,64);
    }
    LCD_DrawRect(144,127,48,128);
    LCD_DrawRect(191,127,48,128);
    LCD_SetBackColor(Blue);
    LCD_SetTextColor(White);
}

/* ESCRIBIR para meter PIN TARJETA ------------------------------------------------*/
void Escribir_PIN(int horizontal,int vertical)
{


    if ((posicion>=0)&&(posicion<8)){
        if ((vertical>=144) && (vertical<168)){
            if((horizontal>=196) && (horizontal<320)){
            if(opcion<=12)opcion=10;
            else opcion=30;
            Ocultar_Teclado();
            posicion--;
            }
            else if((horizontal>=128) && (horizontal<196)){
            PIN[posicion]=0x03;
            user[posicion]='3';
            }
            else if((horizontal>=64) && (horizontal<128)){
            PIN[posicion]=0x02;
            user[posicion]='2';
            }
            else if((horizontal>=0) && (horizontal<64)){
            PIN[posicion]=0x01;
            user[posicion]='1';
            }
        }
        else if ((vertical>=168) && (vertical<192)){

            if((horizontal>=196) && (horizontal<320)){
            if(opcion<=12)opcion=10;
```

```
            else opcion=30;
            Ocultar_Teclado();
            posicion--;
            }
            else if((horizontal>=128) && (horizontal<196)){
            PIN[posicion]=0x06;
            user[posicion]='6';
            }
            else if((horizontal>=64) && (horizontal<128)){
            PIN[posicion]=0x05;
            user[posicion]='5';
            }
            else if((horizontal>=0) && (horizontal<64)){
            PIN[posicion]=0x04;
            user[posicion]='4';
            }
        }
        else if ((vertical>=192) && (vertical<216)){

            if((horizontal>=196) && (horizontal<320)){
            posicion--;
            }
            else if((horizontal>=128) && (horizontal<196)){
            PIN[posicion]=0x09;
            user[posicion]='9';
            }
            else if((horizontal>=64) && (horizontal<128)){
            PIN[posicion]=0x08;
            user[posicion]='8';
            }
            else if((horizontal>=0) && (horizontal<64)){
            PIN[posicion]=0x07;
            user[posicion]='7';
            }
        }
        else if ((vertical>=216) && (vertical<240)){

            if((horizontal>=196) && (horizontal<320)){
            PIN[posicion-1]=0xFF;
            user[posicion-1]=' ';
            mpass[posicion-1]=' ';
            posicion-=2;
            }
            else if((horizontal>=128) && (horizontal<196)){
            posicion--;
            }
            else if((horizontal>=64) && (horizontal<128)){
            PIN[posicion]=0x00;
            user[posicion]='0';
            }
            else if((horizontal>=0) && (horizontal<64)){
            posicion--;
            }
        }

        Delay(40);
        if(posicion>=0){
        mpass[posicion]='*';
        posicion++;
        }

    }
    else {

        if ((vertical>=216) && (vertical<240)){
            if((horizontal>=196) && (horizontal<320)){
                PIN[posicion-1]=0xFF;
                user[posicion-1]=' ';
```

```
                mpass[posicion-1]=' ';
                posicion--;
                LCD_DisplayStringLine(Line5,"                    ");
            }
        }
        else if (((vertical>=168) && (vertical<192)) | ((vertical>=144) && (vertical<16
8))){

            if((horizontal>=196) && (horizontal<320)){
            if(opcion<=12)opcion=10;
            else opcion=30;
            Ocultar_Teclado();
            }
        }
        else LCD_DisplayStringLine(Line5," MAX. 8 CARACTERES ");
    }

if ((posicion<=0)|(posicion>8)) posicion=0;
if (posicion==8) posicion=8;


}

/* PANTALLA indicando que la OPERACION NO ha sido ADMITIDA ----------------------*/
void Pantalla_No_Valido(void)
{
    unsigned int a=0;
    Pantalla(Black,Red);
    for(a=1;a<48;a++){
    LCD_DrawCircle(72, 174, a);
    }
    LCD_SetTextColor(White);
    LCD_DrawRect(60, 210, 24, 72);
    for(a=0;a<24;a++){
    LCD_DrawLine((60+a), 210, 71, Horizontal);
    }
}
/* PANTALLA indicando que la OPERACION SI ha sido ADMITIDA ----------------------*/
void Pantalla_Si_Valido(void)
{
    unsigned int a=0;
    Pantalla(Black,Green);
    for(a=1;a<48;a++){
    LCD_DrawCircle(72, 174, a);
    }
    LCD_SetTextColor(White);
    for(a=0;a<26;a++){
    LCD_DrawLine((68+a), (210-a), 15, Horizontal);
    }
    for(a=0;a<38;a++){
    LCD_DrawLine((57+a), (146+a), 15, Horizontal);
    }
}
```

```
/**
  ******************************************************************************
  * @file    IOExpander/Funciones.h
  * @author  Mario Acevedo Aguilar
  * @version V 2.0
  * @date    15/08/2010
  * @brief   Header Bloque de Funciones Utilizadas
  ****************************************************************************** **/

/* Variables -----------------------------------------------------------*/


extern unsigned char clave[20];
extern unsigned char PIN[8];
extern unsigned char user[20];
extern unsigned char* mens;
extern unsigned char mpass[20];
extern unsigned char letra[20];


/* Funciones -----------------------------------------------------------*/
void Inicio(void);
void Menu(void);
void Menu_PIN(void);
void Pantalla(__IO uint16_t Fondo,__IO uint16_t Letra);
void Mostrar_Mensaje_Presentacion(void);
void Editar_Hora(void);
void Pantalla_Hora(void);
void Editar_Fecha(void);
void Pantalla_Fecha(void);
void Comprobar_Pantalla(void);
void Teclado(void);
void Ocultar_Teclado(void);
void Escribir(int horizontal,int vertical);
int Comprobar_Clave(void);
int Comprobar_Horario(void);
void Teclado_PIN(void);
void Escribir_PIN(int horizontal,int vertical);
void Reinicio_Datos(void);
void Pantalla_No_Valido(void);
void Pantalla_Si_Valido(void);
```

```
/**
  ******************************************************************************
  * @file    IOExpander/Tarjeta.c
  * @author  Mario Acevedo Aguilar
  * @version V 2.0
  * @date    15/08/2010
  * @brief   Funciones Tarjeta
  ****************************************************************************** **/
/* Includes ------------------------------------------------------------------*/

#include "main.h"
#include "Tarjeta.h"
#include "Configuracion.h"
#include "Funciones.h"

unsigned char    SW_Buffer[255];
unsigned char    ATR_Buffer[21];
unsigned char    APDU_VERIFY[13]= {0x00,0x20,0x00,0x00,0x08,0x00,0x00,0x00,0x00,0x00,0x
00,0x00,0x00};
unsigned int puerta=0x09;

/* Funciones -----------------------------------------------------------------*/


/* Recepcion ATR -------------------------------------------------------------*/
void Recepcion_ATR(void)
{
    while(cont_ATR<long_ATR){
        timeout++;
        if (timeout>0x00FFFFF0) break;
        if (((USART3->SR&0x20)!=0)) {
            ATR_Buffer[cont_ATR]=(unsigned char)USART3->DR;
            cont_ATR++;
    /* Calculamos la longitud del ATR mediante el bit T0 */

            if (cont_ATR==2){

                if((ATR_Buffer[1]&0x80)!=0) Bits_Interfaz++;
                if((ATR_Buffer[1]&0x40)!=0) Bits_Interfaz++;
                if((ATR_Buffer[1]&0x20)!=0) Bits_Interfaz++;
                if((ATR_Buffer[1]&0x10)!=0) Bits_Interfaz++;

                for(Bits_Historicos=0;Bits_Historicos<0x0F;Bits_Historicos++){
                    if((ATR_Buffer[1]&0x0F)==Bits_Historicos) break;
                }
                long_ATR=Bits_Interfaz+Bits_Historicos+2;
            }
        }
    }
}

/* Ciclos de Espera ----------------------------------------------------------*/
void Ciclo_Espera(unsigned int n)
{
    unsigned int i=0;
    for (i=0; i<n; i++);
}

/* Envio de Comandos ---------------------------------------------------------*/

void EnviarComando (unsigned char *comando, unsigned char long_comando, unsigned char *
respuesta, unsigned char *long_respuesta, unsigned char *SW1, unsigned char *SW2)
{

    unsigned char fin_PB = 0,fin_APDU = 0, cont_comando, cont_respuesta;
    unsigned char buffer_respuesta[255];
    unsigned short i=0;
```

```
    /* Envio de Comando ------------------------------------------------------*/

    USART3->SR &= ~(0x20);                                  // Limpiamos flag de recepción
    cont_comando =0;                                        // Ponemos a cero el contador

    while (cont_comando<5){
        timeout++;
        if (timeout>0x00FFFFF0) break;
        while ((USART3->SR&0x40)==0);                       // Esperamos a que la transimis
ión esté lista
        USART3->DR = *(comando+cont_comando);               // Mandamos byte a byte el coma
ndo
        cont_comando++;
    }

    /* Recepción del Byte de Procedimiento (PB) ------------------------------*/

    cont_respuesta = 0;
    fin_PB = 0;
    fin_APDU = 0;

    while(fin_PB==0){
        timeout++;
        if (timeout>0x00FFFFF0) break;
        if ((USART3->SR&0x20)!=0) {    // SI HAY DATOS en el REGISTRO de LECTURA
            buffer_respuesta[cont_respuesta] = (unsigned char)USART3->DR;
            cont_respuesta++;
            if (cont_respuesta>=3) {
                if (buffer_respuesta[cont_respuesta-1]!=0x60) {
                    if (((buffer_respuesta[cont_respuesta-1]&0xF0)==0x60)||((buffer_res
puesta[cont_respuesta-1]&0xF0)==0x90)) {
                        /* Recepción de los Bytes de Estado ----------------------*/
                        while ((USART3->SR&0x20)==0);
                        buffer_respuesta[cont_respuesta] = (unsigned char)USART3->DR;
                        *SW1 = buffer_respuesta[cont_respuesta-1];
                        *SW2 = buffer_respuesta[cont_respuesta];
                        *long_respuesta = 0;
                        fin_APDU = 1;
                        fin_PB=1;
                        }
                    else fin_PB=1;
                }
            }
        }
    }

    if (fin_APDU==0) {
        if (long_comando>5) {

        /* ENVIO del RESTO de la INSTRUCCIÓN -------------------------------------*/
            USART3->SR &= ~(0x20);              // Limpiamos flag de recepción
            for (i=0; i<10000; i++);
            while (cont_comando<long_comando){
                timeout++;
                if (timeout>0x00FFFFF0) break;
                while ((USART3->SR&0x40)==0);   // Transmit data register VACÍO
                USART3->DR = *(comando+cont_comando);
                cont_comando++;
            }

        /* RECEPCION DEL SW ------------------------------------------------------*/
            cont_respuesta = 0;
            while(cont_respuesta<0x04){
                timeout++;
                if (timeout>0x00FFFFF0) break;
                if ((USART3->SR&0x20)!=0) {
                    buffer_respuesta[cont_respuesta] = (unsigned char)USART3->DR;
                    cont_respuesta++;
```

```
                    }
                }
                *SW1 = buffer_respuesta[cont_respuesta-2];
                *SW2 = buffer_respuesta[cont_respuesta-1];
                *long_respuesta = 0;
            }
            /* SI por el contrario queremos recibir RESPUESTA ------------------------*/
            else if (*long_respuesta > 0) {
                /* RECEPCION DEL DATOS + SW ----------------------------------------*/
                cont_respuesta = 0;
                while(cont_respuesta<((*long_respuesta) +2)){
                    timeout++;
                    if (timeout>0x00FFFFF0) break;
                    if ((USART3->SR&0x20)!=0) {
                        buffer_respuesta[cont_respuesta] = (unsigned char)USART3->DR;
                        cont_respuesta++;
                        }
                    }
                *SW1 = buffer_respuesta[cont_respuesta-2];
                *SW2 = buffer_respuesta[cont_respuesta-1];
                for (i=0; i<(cont_respuesta-2); i++) {
                    *(respuesta+i) = buffer_respuesta[i];
                    }
            }
        }
if (timeout<0x00FFFFF0) timeout=0;
}

/* Comprobacion DATOS TARJETA ------------------------------------------------------*/
void Comprobacion(unsigned char *usuario)
{
    unsigned int p=0;
    unsigned int correcto=0;
    unsigned int entrada, salida=0;

    /* Guardamos los Permisos de Hora en variables--------------------------------*/
    entrada=((unsigned int)(*(usuario+5)*3600)+(unsigned int)(*(usuario+6)*60));
    salida=((unsigned int)(*(usuario+7)*3600)+(unsigned int)(*(usuario+8)*60));

    /* Comprobamos si tiene permiso para esta puerta -----------------------------*/
    for(p=0;p<5;p++){
        if(*(usuario+p)!=puerta)correcto=0;
        else{
            correcto=1;
            break;
        }
    }

    /* Comprobamos el resto de parámetros de permisos ----------------------------*/
    if (correcto==1){
        if ((*(usuario+9)&mascara)==mascara){
            if ((valorRTC>=entrada)&&(valorRTC<=salida)){
                Pantalla_Si_Valido();
                LCD_DisplayStringLine(Line6,"  PUERTA ABIERTA    ");
                GPIOD->BSRR |= (1<<4);
            }
            else{
                Pantalla_No_Valido();
                LCD_DisplayStringLine(Line7,"HORARIO NO PERMITIDO");
                GPIOD->BSRR |= (1<<13);
            }
        }
        else{
            Pantalla_No_Valido();
            LCD_DisplayStringLine(Line7,"  DIA NO PERMITIDO  ");
            GPIOD->BSRR |= (1<<13);
        }
    }
```

```
    else{
        Pantalla_No_Valido();
        LCD_DisplayStringLine(Line6,"  PUERTA DENEGADA   ");
        GPIOD->BSRR |= (1<<13);
    }

}

/* Reinicio de DATOS TARJETA ------------------------------------------------------*/

void Reinicio_Datos_Tarjeta(void)
{
    int i=0;

    /* Desactivamos CMDVCC para permitir volver a DETECTAR la TARJETA. (Activo a nivel
bajo)*/

    GPIOE->BSRR =(1<<14);

    /* Reiniciamos TODAS las VARIABLES */

    cont_ATR=0;
    Bits_Interfaz=0;
    Bits_Historicos=0;

    /* Limpiamos TODOS los BUFFERS */

    for(i=0;i<21;i++){
        ATR_Buffer[i]=0;
    }
    for(i=0;i<50;i++){
        SW_Buffer[i]=0;
    }
}
```

```
/**
  ******************************************************************************
  * @file    IOExpander/Tarjeta.h
  * @author  Mario Acevedo Aguilar
  * @version V 2.0
  * @date    15/08/2010
  * @brief   Header Funciones Tarjeta
  ****************************************************************************** **/

/* Variables -----------------------------------------------------------------*/


    static unsigned short i=0;

    /* ABRIR ROOT ----------------------------------------------------------*/
    static unsigned char     APDU_ROOT[9]= {0x00,0xA4,0x04,0x00,0x04,'R','O','O','T'};
    /* ABRIR ACCESOS -------------------------------------------------------*/
    static unsigned char     APDU_ACCESOS[12]= {0x00,0xA4,0x04,0x00,0x07,'A','C','C','E
','S','O','S'};
    /* VERIFICAR CODIGO SEGURIDAD ------------------------------------------*/
    extern unsigned char     APDU_VERIFY[13];
    /* LEER REGISTRO 1 -----------------------------------------------------*/
    static unsigned char     APDU_READ1[5]= {0x00,0xB0,0x81,0x00,0x14};
    /* LEER REGISTRO 2 -----------------------------------------------------*/
    static unsigned char     APDU_READ2[5]= {0x00,0xB0,0x82,0x00,0x0A};
    /* BUFFER DE RESPUESTA -------------------------------------------------*/
    extern unsigned char     SW_Buffer[255];
    /* BUFFER DE ATR -------------------------------------------------------*/
    extern unsigned char     ATR_Buffer[21];
    /* Variables generales -------------------------------------------------*/
    static unsigned char SW1, SW2, long_respuesta;
    extern unsigned int puerta;
    static unsigned int cont_ATR = 0, long_ATR = 21, Bits_Interfaz = 0, Bits_Historicos
 = 0;

/* Funciones -----------------------------------------------------------------*/

void Recepcion_ATR(void);
void Ciclo_Espera(unsigned int n);
void EnviarComando (unsigned char *comando, unsigned char long_comando, unsigned char *
respuesta, unsigned char *long_respuesta, unsigned char *SW1, unsigned char *SW2);
void Comprobacion(unsigned char *usuario);
void Reinicio_Datos_Tarjeta(void);
```