Universidad Carlos III de Madrid

Escuela Politécnica Superior

Departamento de Tecnología Electrónica

Proyecto Fin de Carrera

Implementación de Interfaces de Dispositivos para Sistemas Empotrados basados en Microcontroladores ARM7

Ingeniería Técnica de Telecomunicación: Telemática

Autor: Sandra Cid Pantoja

Tutor: Raúl Sánchez Reíllo

Septiembre 2010

A todas esas personas,

que forman parte de mi vida,

por estar ahí incondicionalmente.

Agradecimientos

En primer lugar quiero dar las gracias a mis padres, Lola y Silver, por haber estado a mi lado siempre, tanto en los momentos buenos como en los momentos malos, por haberme ayudado cuando lo he necesitado, por soportar todos esos momentos en los que, por desgracia, os lo he hecho pasar mal, por, en definitiva, ser como sois. Os quiero.

No me olvido de nombrar al resto de mi familia, en especial a mi hermano, Sergio; mis yayos, Lola, Goya, Luz y Miguel; mis tios, Elena, Pilar, Miguel y Domingo; mis primas, Patrizia, Paula y Marta; y mis perros, Dana y Rony. Gracias por todos esos buenos momentos que hemos vivido juntos.

A Cris, con quien inicié hace muchos años una gran amistad, por ser ahora mismo como una hermana para mí. Porque hemos crecido y madurado juntas, porque hemos pasado por todos esos buenos momentos juntas, aunque algunos de ellos hayan sido "surrealistas", por llegar a tener esta confianza que nos hace saber todo una de la otra.

A todos esos amigos, con quien mantengo más o menos contacto, pero que no han pasado por mi vida inadvertidos, en especial a Rober (mi clon), Raquel (La Pollo), Dani (Wally Potter), Ainhoa, Alex, Iván, Joan, Víctor, Cristina, Javi, Blanca, Silvia, por haberme hecho pasar tantos buenos momentos en estos años.

A todos mis compañeros de la carrera, en especial a Patricia, Alma, Luis, Pau, Aída y Nacho, porque con ellos he pasado tanto días estresantes de clases, prácticas y estudio, como días de relax en esos bares de "El Callejón", o tirados en el césped del Campus, porque han llegado a convertirse en buenos amigos.

A mi tutor, Raúl, por haberme dado la oportunidad de conocer este mundo tan interesante como es la Biometría, por su apoyo y ayuda, por ser, a parte de tutor y jefe, amigo.

A esa nueva "familia" que he tenido el placer de conocer gracias a mi tutor, "Los Gutitos", por toda la ayuda que he recibido cuando lo he necesitado, por hacer más llevaderos los días de curro, por todos esos viajes que hemos compartido, por todas esas "gutifiestas " en la que he disfrutado como una enana.

A todas esas personas que no he nombrado que han aportado su granito de arena para que haya llegado hasta aquí hoy.

Muchas gracias a tod@s

Índice General

1.Introducción.....1

1.1.Objetivos.....1

1.2.Estructura del Documento.....3

2.Biometría.....5

2.1.Introducción a la Biometría.....5

2.2.Identificación biométrica 6

2.3.Fases de un sistema de identificación biométrica.....7

2.4.Funcionamiento en un sistema de identificación biométrica.....10

2.5.Modalidades de identificación biométricas.....11

3. Tarjetas inteligentes.....19

3.1.Antecedentes a la tarjeta inteligente.....19

3.2.Arquitectura de la tarjeta inteligente.....21

3.3.Sistema operativo de la tarjeta inteligente (SOTI).....25

3.4.Sistema de ficheros de una tarjeta inteligente 26

3.5.Seguridad en la tarjeta inteligente.....27

4.Protocolo USB.....29

4.1.Introducción a USB.....29

4.1.1.Antecedentes.....29

4.1.2.Descripción General.....30

4.2.Topología.....31

4.2.1.Host USB....31
4.2.2.Controlador del host USB....32
4.2.3.Dispositivos USB....33
4.3.Flujo de datos....35
4.3.1.Endpoints y direcciones del dispositivo....36
4.3.2.Pipes....36
4.3.3.Tipos de Transferencias....38
4.4.Capa de protocolo....40
4.4.1.Formato de los Campos....40
4.4.2. Formato de los Paquetes....41
4.4.3.Transacciones....42
4.4.4.Split....43

5.Protocolo TCP/IP....45

5.1.Introducción a TCP/IP....45
5.1.1.Antecedentes....45
5.1.2.Estructura Interna....45
5.2.Nivel de Red: IP....46
5.2.1.Introducción....46
5.2.2.Encaminamiento....47
5.2.3.Pasarelas....48
5.2.4.Resolución de direcciones....48
5.3.Nivel de Transporte: TCP....49
5.3.1.Introducción....49
5.3.2.Fragmentación....51
5.3.3.Fases de TCP....51
5.3.4.Formato de segmentos TCP....52

6.Diseño del Sistema....55

6.1.Descripción de los dispositivos....55
6.1.1.Placa de Desarrollo: MCB2470....55
6.1.2.Lector de Tarjetas Inteligentes: SDI010....57

6.1.3.Lector de Tarjetas Inteligentes: SCL010....58
6.1.4.Pantalla Táctil....59
6.1.5.Sensor de Huella....60
6.2.Análisis del protocolo de los dispositivos....62
6.2.1.Introducción....63
6.2.2.Análisis del Lector de Tarjetas Inteligentes SDI010....63
6.2.3.Análisis del Lector de Tarjetas Inteligentes SCL010....67

7.Desarrollo del Sistema....69
7.1.Lector de Tarjetas Inteligentes vía USB....69
7.1.1.Introducción a la conexión USB....69
7.1.2.Comunicación con el Lector de Tarjetas Inteligentes....71
7.1.3.Diagramas de Flujo de las Principales Funciones....72
7.2.Pantalla Táctil vía SPI....75
7.2.1.Introducción a la conexión SPI....75
7.2.2.Pantallas definidas para el Sistema Desarrollado....77
7.2.3.Relación entre elementos de las pantallas y acciones a realizar....81
7.3.Sensor de Huella vía SPI....84
7.3.1.Introducción a la conexión SPI....84
7.3.2.Comunicación con el Sensor de Huella....84
7.4.Conexión Ethernet....85
7.4.1.Introducción a la Conexión de Red....85
7.4.2.Comunicación vía Ethernet....87
7.4.3.Diagramas de Flujo de las Principales Funciones....91
7.5.Aplicación de escucha del PC....94
7.5.1.Introducción....94
7.5.2.Diagramas de flujo....96
7.6.Sistema Principal....100
7.6.1.Métodos auxiliares para el Sistema Principal....100
7.6.2.Diagramas de Flujo de los Métodos Auxiliares....101
7.6.3.Diagrama de Flujo del Método Principal....108

8.Presupuesto.....111

8.1.Descomposición de actividades.....111

9.Conclusiones y Líneas de Futuro.....115

9.1.Conclusiones.....115

9.2.Líneas de Futuro.....116

10.Bibliografía 119

Listado de Acrónimos

- **ADC:** conversor analógico-digital (*Analog to Digital Converter*)
- **CPU**: Unidad central de procesamiento (*Central Processing Unit*)
- **DAC:** conversor digital-analógico (*Digital to Analog Converter)*
- **DF**: Fichero dedicado *(Dedicated File)*
- **DMA:** para comunicar con la RAM sin pasar por el procesador *(Direct Memory Access)*
- **EEPROM / E2PROM**: Memoria de solo lectura reescribible eléctricamente *(Electrically Erasable Programmable Read Only Memory)*
- **EF**: Fichero elemental (*Elementary file)*
- **FAR**: Tasa de falsa aceptación (*False Accept Rate*)
- **FIR**: Tasa de falsa identificación (*False Identification Rate*)
- **FLASH:** Memoria EEPROM desarrollada que permite que múltiples posiciones de memoria sean escritas o borradas en una misma operación de programación mediante impulsos eléctricos.
- **FMR**: Tasa de falso reconocimiento (*False Match Rate*)
- **FNMR**: Tasa de falso no reconocimiento (*False Non-Match Rate*)
- **FRR**: Tasa de falso rechazo (*False Reject Rate*)
- **FTA**: Fallo al adquirir (*Failure To Acquire*)
- **FTR** o **FER**: Tasa de fallo al inscribir (*Failure To Enroll Rate*)
- **I^2C:** bus de comunicaciones en serie (*Inter-Integrated Circuit*)
- **IrDA:** estándar físico en la forma de transmisión y recepción de datos por rayos infrarrojos. (*Infrared Data Association)*
- **I^2S:** interfaz para audio digital (*Inter-IC Sound*)
- **LCD**: pantalla de cristal líquido (*Liquid Crystal Display*)
- **LED:** diodo emisor de luz (*Light-Emitting Diode*)
- **MAC:** identificador de 48 bits que corresponde de forma única a una interfaz de red (*Media Access Control)*
- **MF**: Fichero maestro *(Master File)*
- **MMC**: es un estándar de tarjeta de memoria *(MultiMedia Card)*
- **µP:** microprocesador.
- **PIN**: Número de Identificación Personal *(Personal Identification Number)*
- **RAM**: Memoria de acceso aleatorio (*Random access memory*)
- **RFID**: Identificación por radiofrecuencia (*Radio frequency Identification*)
- **ROM**: Memoria de solo lectura (*Read only memory*)
- **RS232:** interfaz para intercambio de datos en serie (*Recommended Standard 232*)
- **SD:** es un formato de tarjeta de memoria *(Secure Digital)*

- **SDK**: Kit de desarrollo de software *(Software Development Kit)*
- **SOTI**: Sistema operativo de la tarjeta inteligente.
- **SPI:** principalmente para la transferencia de información entre circuitos integrados en equipos electrónicos (*Serial Peripheral Interface)*
- **STN:** tipo de LCD con más contraste *(Super-Twisted Nematic)*
- **TCP/IP**: Protocolo de Control de Transmisión / Protocolo de Internet *(Transmission Control Protocol / Internet Protocol)*
- **TFT:** transistor de Película Fina (*Thin Film Transistor)*
- **UART:** transmisor-receptor asíncrono universal *(Universal Asynchronous Receiver-Transmitter)*
- **UDP**: Protocolo de datagrama de usuario (*User Datagram Protocol*)
- **USB:** bus universal en serie (*Universal Serial Bus*)
- **VIC:** controlador de vectores de interrupción *(Advanced Vectored Interrupt Controller)*

1. Introducción

Este proyecto se centra en la implementación de distintas interfaces de un microcontrolador, como son USB, Ethernet y SPI. El entorno de desarrollo que vamos a utilizar para programar el microcontrolador es μVision4 de Keil.

La interfaz Ethernet será implementada para que el microcontrolador sea capaz de comunicarse vía red con un PC. La interfaz SPI será necesaria para poder comunicarse con una pantalla táctil y con un sensor de huella. Y, por último, la interfaz USB será desarrollada para poder establecer comunicación con distintos dispositivos, tales como lectores de tarjetas inteligentes.

En este capítulo, en primer lugar, se describirán los objetivos del proyecto, donde se explicará con más detalle el sistema a desarrollar, y el por qué necesitaremos implementar las interfaces mencionadas. Terminaremos el capítulo de introducción comentando la estructura del documento.

1.1. Objetivos

El sistema que se va a desarrollar está formado por un ordenador, un microcontrolador, un lector de tarjetas inteligentes, tanto con contactos como sin contactos, un sensor de huella y una pantalla táctil. El ordenador y el microcontrolador estarán conectados en red. Al microcontrolador estarán conectados el lector de tarjetas a través de una interfaz USB y la pantalla táctil y el sensor de huella a través de una interfaz SPI. En la figura 1.1 se muestra un esquema del conexionado.

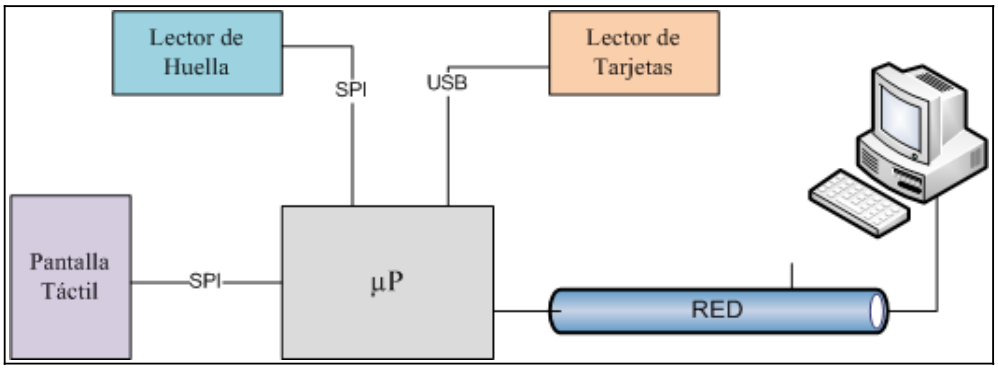


***Fig. 1.1**: Escenario planteado.*

En el ordenador se ejecutará una aplicación, que escuchará en un puerto TCP (por defecto será el 9999, pero podrá ser editado por el usuario), para establecer comunicación con el microcontrolador mediante el protocolo TCP/IP.

Las funcionalidades del microcontrolador serán:

- Configurar las interfaces USB, Ethernet y SPI para poder establecer comunicación con los distintos dispositivos del sistema.
- El microcontrolador tendrá que obtener las credenciales de los usuarios del lector de tarjetas vía USB. Para ello se analizará la comunicación entre un ordenador y el lector. El ordenador tendrá instalados los drivers correspondientes y en él se ejecutará una aplicación que sea capaz de comunicarse con él.
- El microcontrolador tendrá que obtener la imagen de la huella de los usuarios del sensor de huella vía SPI.
- Se desarrollará una interfaz gráfica para que el usuario pueda hacer uso de las funcionalidades del sistema mediante la pantalla táctil.

Los usuarios, a través de la pantalla táctil, podrán realizar las siguientes acciones:

- Se podrá definir la dirección IP del microcontrolador y el puerto con el cual se establecerá la conexión TCP para comunicarse con el PC.
- Se podrá definir la dirección IP del PC, en el que se estará ejecutando la aplicación de escucha, y el puerto con el cual se establecerá la conexión TCP para comunicarse con el microcontrolador.
- Se podrá enviar la huella del usuario al PC por red.
- Se podrá enviar el número de serie de la tarjeta inteligente al PC por red.

1.2. Estructura del Documento

En los siguientes capítulos de este documento se describen todos los aspectos de interés que tienen relación con el proyecto, desde el estado actual de la tecnología implicada hasta los resultados y conclusiones obtenidas tras la implementación, pasando por los aspectos más relevantes del desarrollo del proyecto.

- **1. Introducción:** se trata del presente capítulo. Se definen los objetivos que se pretenden conseguir con el proyecto.
- **2. Biometría:** breve introducción general a la biometría que ayude al lector a conocer su situación actual.
- **3. Tarjetas Inteligentes:** breve introducción a las tarjetas inteligentes que ayude al lector a conocer su funcionamiento.
- **4. Protocolo USB:** breve introducción al protocolo USB que ayude al lector a conocer su funcionamiento.
- **5. Protocolo TCP/IP:** breve introducción al protocolo TCP/IP que ayude al lector a conocer su funcionamiento.
- **6. Diseño del Sistema:** capítulo en el que se describe la arquitectura y funcionamiento del sistema desarrollado.
- **7. Desarrollo del Sistema:** dedicado a mostrar cómo se ha implementado el sistema según el diseño descrito en el capítulo anterior.
- **8. Conclusiones y Líneas de futuro:** donde se exponen las conclusiones obtenidas y se recogen las posibles líneas futuras de desarrollo.
- **9. Bibliografía:** información consultada para la realización de esta memoria.

2. Biometría

Como se ha comentado en el capítulo de introducción, en este proyecto se hará uso de un lector de huella para capturar la huella del usuario y posteriormente mandarla a un PC. En este capítulo se hará una breve introducción a la biometría para comprender el funcionamiento de un sistema de identificación biométrico y entender las distintas aplicaciones futuras de esta parte del proyecto.

2.1. Introducción a la Biometría

Etimológicamente la palabra biometría procede del griego '*bios*' (vida) y '*metron*' (medida) y hace referencia a la medida de las características biológicas de un individuo. Según el diccionario de la lengua española el término biometría es el "estudio mensurativo o estadístico de los fenómenos o procesos biológicos", es decir, la aplicación de técnicas matemáticas y estadísticas a las ciencias biológicas.

Una definición más precisa del significado que realmente tiene este término puede darse dentro del campo de la identificación de personas, donde la biometría se define como la ciencia que estudia las características anatómicas o de comportamiento de una persona, con el objeto de que pueda ser reconocida.

Es de sobra conocido por todos que el ser humano posee características particulares (rasgos biológicos), que le diferencian enormemente del resto de individuos de su especie, haciendo único e irrepetible a cada individuo. Características biofísicas como pueden ser el rostro, la huella de sus dedos, los ojos o las manos, y características de comportamiento como pueden ser su manera de caminar, la forma en que habla, o su manera de escribir.

No fue sin embargo hasta finales del siglo XIX cuando realmente se desarrolla la biometría como ciencia que utiliza parámetros biológicos para la identificación de un individuo. Es entonces cuando aparece el sistema de identificación antropométrica de Bertillón, mediante el cual una persona era identificada según una serie de medidas de su cuerpo (como el diámetro de su cabeza, la longitud de sus extremidades o el tamaño del tronco). En esta época también comienza a

estudiarse la identificación de personas a través de sus huellas dactilares. Se llevan a cabo multitud de estudios sobre la huella dactilar, estudiándose cuáles son los diferentes elementos característicos de una huella, su estabilidad con el tiempo en una persona, es decir, si varía o no con los años, y también profundos estudios sobre la unicidad de la huella. Todos estos estudios provocaron una gran aceptación de la huella dactilar como método de identificación biométrica en el ámbito policial. Y con su más de un siglo de existencia se ha convertido en la modalidad biométrica más desarrollada.

2.2. Identificación biométrica

Identificar es el acto de reconocer si una persona es aquella misma que se supone o se busca, o también facilitar determinados datos personales necesarios para ser reconocido.

La identidad de una persona puede determinarse de muy diferentes maneras. Una de ellas es solicitándole una determinada información que solamente él y la entidad donde quiere identificarse conocen, como un número de identificación personal (PIN) o una contraseña. El inconveniente de esta técnica es doble, por un lado el usuario puede perder u olvidar el patrón que le identifica, sobre todo si los patrones a recordar difieren según donde el usuario quiera autenticarse y tiene que recordar un número elevado de ellos. El otro inconveniente es el robo, cualquiera que consiga hacerse con la contraseña o número de identificación de otro usuario podrá suplantar su identidad. No obstante cuando un patrón de identificación es comprometido éste es fácilmente reemplazable por otro nuevo.

Otra manera de identificar a un usuario es mediante la posesión de un elemento que le identifique, como puede ser una llave, una tarjeta o un certificado digital. Este tipo de técnicas basadas en la posesión a menudo se complementan con otras técnicas de identificación para hacerlas más seguras. Por ejemplo, un usuario de un cajero electrónico debe autenticarse antes de efectuar cualquier operación económica. El usuario dispone de una tarjeta de banda magnética que contiene información acerca de su identidad, y además debe introducir un número secreto que solamente él y la entidad bancaria conocen para probar su identidad. Los inconvenientes de estas técnicas son los mismos que en la técnica anterior, ya que un usuario malintencionado podría hacerse con el número secreto y la tarjeta de identificación de otra persona.

La utilización de claves secretas y/o tarjetas de identificación no es suficiente en entornos que requieran un mayor nivel de seguridad, y en los cuales es imprescindible verificar sin lugar a dudas la identidad que un usuario dice ser. Es en este tipo de aplicaciones donde entran en juego

las modalidades de identificación biométricas.

La identificación biométrica se basa en el reconocimiento de características biológicas o de comportamiento, como pueden ser la huella dactilar, el iris, la voz o la firma.

Las modalidades de identificación biométrica, frente a otras formas de autenticación personal, presentan la ventaja de que los patrones son distintivos del individuo, no pueden ser olvidados, ya que siempre los llevamos con nosotros, no pueden ser sustraídos y no son fácilmente reproducibles. Debido a que la persona a ser identificada necesita estar físicamente frente al terminal de identificación la biometría es una técnica más confiable.

Sin embargo no todo son ventajas, para evitar el fraude en los sistemas de autenticación biométricos es necesaria la realización de pruebas colaterales, (como la detección de elemento vivo), de manera que se tenga constancia de que lo que realmente se encuentra delante del punto de identificación es una persona viva y no una imagen o una copia. Otro inconveniente importante es el siguiente; si una información biométrica es comprometida no es posible reemplazarla. Por ejemplo en el caso de identificación por número secreto o por contraseña, si la clave de la cuenta de un usuario es comprometida puede sustituirse fácilmente por otra nueva, estando el resto de cuentas con distinta clave a salvo. Si una información biométrica es comprometida el atacante podría tener acceso a parte o a todo el sistema, dependiendo del nivel de privilegio asociado a la biometría. Otros inconvenientes de la identificación biométrica son que la identificación se da en términos de probabilidad y también que determinados equipamientos presentan un coste de adquisición y mantenimiento elevados.

2.3. Fases de un sistema de identificación biométrica

Un sistema de identificación biométrica se basa en la medición, ya sea directa o indirecta, de las características biológicas o de comportamiento de un usuario para identificarle de forma automática. Para ello se pueden utilizar técnicas estadísticas como reconocimiento de patrones de señal o comparación, y también técnicas de inteligencia artificial como lógica borrosa o redes neuronales. Independientemente de la técnica biométrica utilizada se sigue un esquema que consta de dos fases o etapas diferentes: reclutamiento y utilización.

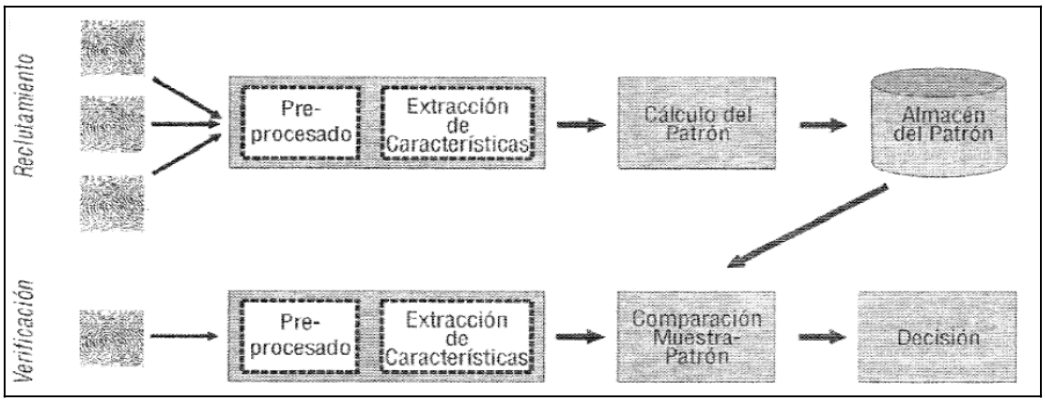


***Fig. 2.1**: Etapas en un sistema de identificación biométrica (Fuente:[1]).*

- **Reclutamiento**: en esta primera fase los datos biométricos del usuario son capturados y procesados para posteriormente extraer su información característica. Esta información, conocida como patrón biométrico, caracteriza al usuario, y puede ser almacenada bien en una base de datos o bien en un dispositivo portátil como una tarjeta inteligente.
- **Utilización**: una vez obtenido y almacenado el patrón biométrico del usuario éste está preparado para utilizar el sistema. Para ello cada vez que el usuario desee autenticarse debe presentar su rasgo biométrico al sistema, que calculará el patrón biométrico de la muestra obtenida del usuario, comparándolo con el patrón almacenado del usuario. Como resultado de la comparación se obtendrá una determinada probabilidad de éxito o fracaso en la identificación.

Como puede observarse en la figura anterior, las fases de reclutamiento y de utilización comparten una serie de pasos que son comunes:

- **Captura**: en esta primera fase se obtienen los datos biométricos o de comportamiento del usuario, dependiendo este proceso de la modalidad biométrica empleada. Durante la fase de reclutamiento el proceso de captura se realiza de manera supervisada, es decir, una persona comprueba cómo se toman los datos, cerciorándose además de la identidad de la persona.
- **Pre-procesado y extracción de características**: la fase de pre-procesado modifica los datos capturados de manera que resulte más fácil obtener las características más significativas de los mismos. El proceso de extracción de características es la parte más importante del sistema de identificación biométrica, ya que determina la capacidad del sistema para distinguir entre dos sujetos distintos con características biométricas parecidas. En la fase de reclutamiento una vez obtenidas las características de la muestra capturada se calcula el patrón identificativo del usuario y se almacena. Este patrón caracterizará al usuario.

- **Comparación y decisión**: en la fase de utilización, una vez obtenidas las características de la muestra en cuestión, se comparan con el conjunto de patrones almacenados en el sistema y se toma una decisión. Es importante destacar que el proceso de comparación no se trata de una comparación de igualdad entre muestras, ya que estas mismas pueden verse modificadas según se realice el proceso de captura, e incluso las características biológicas del sujeto pueden variar levemente.

Por este motivo el proceso de comparación consiste básicamente en una medida de semejanza entre el patrón de características de la muestra tomada y los patrones almacenados en el sistema. Esta comparación puede realizarse de muy diversas maneras: mediante métricas como la distancia Euclídea o la distancia de Hamming, técnicas estadísticas como el empleo de funciones de distribución, o técnicas basadas en modelado de problemas, como redes neuronales.

Como resultado de la comparación se obtiene una probabilidad de semejanza entre la muestra presentada y una de las muestras anteriormente registradas en el sistema. Según un umbral previamente establecido, que establece el nivel de seguridad del sistema, el éxito o fracaso de la comparación vendrá determinado.

Existen una serie de parámetros que reflejan la calidad del sistema de identificación biométrica:

- **FAR** (False Accept Rate): la tasa de falsa aceptación indica el porcentaje de número de veces que el sistema produce una falsa aceptación. Es decir cuando un individuo es identificado como usuario de manera incorrecta.
- **FMR** (False Match Rate): es la probabilidad de que las personas no autorizadas sean incorrectamente reconocidas durante el proceso de comparación de características. A diferencia de la FAR la FMR no contabiliza los intentos previamente rechazados.
- **FRR** (False Reject Rate): la tasa de falso rechazo indica la probabilidad de que un usuario autorizado sea rechazado por el sistema.
- **FNMR** (False Non Match Rate): es la tasa a la que las personas autorizadas sean falsamente no reconocidas durante el proceso de comparación de características.
- **FTA** (Failure To Acquire): es el número de intentos de verificación o identificación sin éxito por error de personas inscritas correctamente. Puede producirse cuando la imagen tomada por el sensor no sea de calidad suficiente.
- **FTE** o **FER** (Failure To Enroll Rate): es la proporción de personas que no consiguen realizar con éxito el procedimiento de inscripción.

- **FIR** (False Identification Rate): es la probabilidad de que una persona autorizada sea identificada pero se le asigne un identificador falso.

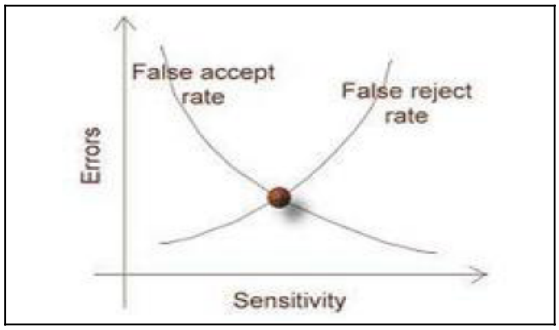


***Fig. 2.2**: Selección del umbral en un sistema de identificación biométrica (Fuente:[7]).*

La elección de un umbral muy alto provoca que el sistema sea muy restrictivo, la tasa de aceptación de usuarios no autorizados se reduce, pero sin embargo la probabilidad de rechazar a usuarios autorizados aumenta. Sin los usuarios son rechazados erróneamente con frecuencia parecerá que el sistema no funciona correctamente, incrementando el grado de insatisfacción de los usuarios. Un umbral más bajo provocará el efecto contrario, el sistema se relajará, permitiendo un mayor porcentaje de accesos de usuarios no autorizados. Por tanto la decisión de un umbral adecuado es imprescindible según el nivel de seguridad y el grado de confianza y amigabilidad hacia el usuario que se le quiera dar al sistema.

2.4. Funcionamiento en un sistema de identificación biométrica

La metodología de funcionamiento de un sistema de identificación biométrica puede presentar dos esquemas de operación bien diferenciados.

El primero de ellos recibe el nombre de **reconocimiento o identificación** y se trata de un proceso de comparación de 1 a N. Consiste en identificar a un usuario dentro de todos los registrados previamente en el sistema, para ello se extraen las características biométricas del usuario a identificar y se comparan con las características existentes en la base de datos del sistema. Como resultado de la comparación puede identificarse al usuario con aquel con el que se haya obtenido una mayor probabilidad de parecido, es decir, siempre se identifica al usuario con uno de los existentes en la base de datos. O sin embargo puede existir un determinado umbral en el sistema, de manera que solo se identifique al usuario con otro siempre y cuando se supere una probabilidad determinada de semejanza. Este esquema de funcionamiento presenta algunos inconvenientes, el primero de ellos es la necesidad de una base de datos que almacene los patrones

característicos de todos los usuarios reclutados en el sistema, con los consecuentes requisitos de capacidad, seguridad y conectividad a la base de datos, ya que es imprescindible que siempre sea posible acceder a la misma y se garantice la privacidad de los datos intercambiados. Otro inconveniente es el tiempo necesario para efectuar la identificación, que puede ser elevado si el número de usuarios en el sistema es muy grande, ya que el número de comparaciones que hay que realizar es alto.

El otro esquema de funcionamiento recibe el nombre de **autenticación o verificación** y se trata de un proceso 1 a 1. En este esquema de funcionamiento el usuario debe dar a conocer su identidad en el sistema biométrico. El sistema, a partir de las características biométricas del usuario calcula su patrón característico y lo compara con el patrón facilitado. Si como resultado de la comparación se obtiene una probabilidad de similitud que supera un determinado umbral el sistema autentica al usuario, ya que da por hecho que el usuario es aquel quien dice ser. El patrón característico del usuario puede estar almacenado en una base de datos tal y como sucede en el esquema de identificación, o por el contrario estar contenido en un dispositivo portátil como una tarjeta inteligente. En este caso se suprime la necesidad de una base de datos que almacene los patrones característicos, y el usuario a identificar debe presentar dicho elemento identificativo en el momento en el que quiera probar tal identidad mediante sus características biológicas o de comportamiento. Otra ventaja que presenta este esquema es la velocidad, en general el proceso de verificación es mucho más rápido que el de identificación, ya que solamente es necesario efectuar una comparación de patrones biométricos.

2.5. Modalidades de identificación biométricas

Las modalidades de identificación biométricas se fundamentan en el análisis de una característica fisiológica o de comportamiento. Aunque en principio cualquier parte del cuerpo humano, o característica de comportamiento de una persona, serían susceptibles de ser usadas para la identificación, se atiende a una serie de criterios prácticos. Lo ideal es que la característica utilizada para la identificación se demuestre única y propia de la persona a identificar. Para ello se selecciona una característica robusta, no sujeta a grandes cambios, que sea lo más distintiva posible respecto al resto de la población, que sea una característica disponible en la mayor cantidad de individuos posibles y que cuente con la aceptación del usuario, ya que algunas veces éste puede percibir al sistema biométrico como excesivamente intrusivo.

En general si el proceso identificativo se hace atendiendo a la anatomía del usuario se habla

de Biometría Estática, y si se realiza según la forma en que el sujeto se comporta se llama Biometría Dinámica. Habitualmente los dispositivos que miden el comportamiento requieren de la cooperación del usuario, por ejemplo, que el usuario diga su nombre o una determinada frase frente al sistema de reconocimiento. Los que miden alguna característica fisiológica también pueden requerir de la colaboración del usuario, aunque existen casos en los cuales el proceso de identificación puede pasar totalmente inadvertido para el usuario, que no tiene que llevar a cabo ninguna acción. Por ejemplo, en la identificación del rostro una videocámara capta una imagen del usuario que se aproxima al sistema, y la procesa automáticamente para su identificación.

Existen gran cantidad de modalidades de identificación biométrica, que para llevar a cabo el proceso de identificación registran un aspecto exclusivo del individuo. Las más utilizadas habitualmente son las siguientes:

- **Huella dactilar**

La identificación de individuos mediante huellas dactilares ha sido siempre reconocida como una de las mejores modalidades de identificación biométrica, ya que se trata de una modalidad profundamente estudiada y cuenta a sus espaldas con más de un siglo de existencia. Multitud de estudios científicos avalan la unicidad de la huella de un individuo, ya que ha sido demostrado que no existen dos dedos con huellas idénticas, ni siquiera entre dedos de una misma persona, ni tampoco entre gemelos. La estabilidad de la huella de una persona con el transcurso del tiempo también ha sido verificada.

Los sistemas biométricos de huella dactilar suelen estar basados en un sensor óptico que captura la huella digital del usuario. Una vez tomada la imagen se lleva a cabo el proceso de extracción de características que puede realizarse de diferentes maneras. Una forma consiste en estudiar la correlación entre la imagen tomada y otra previamente almacenada. Habitualmente el proceso de extracción de características se realiza analizando los elementos de la huella, bien analizando los poros del dedo, o bien obteniendo las minucias de la huellas. Una huella dactilar está formada por una sucesión de crestas, cuyo flujo sufre una serie de puntos singulares denominados **minucias**. Aunque la clasificación de los diferentes tipos de minucias puede ser extensa las minucias más frecuentes son las siguientes:

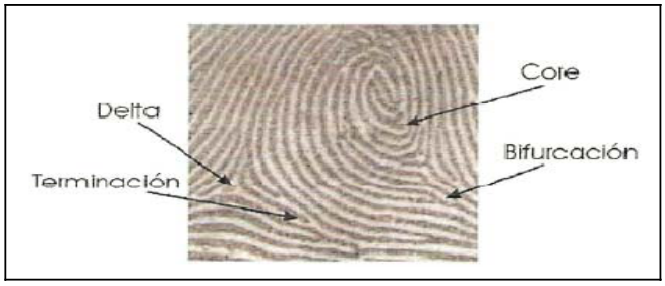


***Fig. 2.3**: Elementos característicos de una huella dactilar (Fuente: [8]).*

- ✔ **Núcleo**: es el punto de la huella donde la orientación de las crestas tiende a converger.
- ✔ **Terminación**: punto de fin de una cresta.
- ✔ **Bifurcación**: punto donde una cresta se divide en dos.
- ✔ **Delta**: punto donde el flujo de las crestas presenta una divergencia.

Como resultado del análisis de los puntos más significativos de la huella se obtiene un patrón dactilar denominado vector de características, que caracteriza unívocamente al usuario. Este patrón garantiza también que la huella original del que procede no puede ser reconstruida a partir de él. A pesar de ser una modalidad muy desarrollada y de medio coste, la identificación por huella presenta una serie de inconvenientes. El primero de ellos es la connotación policial que siempre ha venido asociada a esta tecnología. Otro inconveniente deriva del propio proceso de captura de la huella, ya que la presión y la posición del dedo sobre el lector pueden variar la imagen capturada y producir falso rechazo. Además para evitar posibles riesgos de suplantación de identidad es necesaria la realización de pruebas complementarias durante el proceso de captura, como la detección de elemento vivo.

- **Geometría de la mano**

Este tipo de modalidad puede centrarse en el estudio de determinados parámetros morfológicos de la mano o incluso de un determinado dedo del usuario, como pueden ser la anchura, la altura o las dimensiones. Otras modalidades más avanzadas se basan en la utilización de sensores de infrarrojos capaces de detectar el patrón venoso de la mano.

Estos sistemas habitualmente son más rápidos que otros sistemas biométricos, ya que son bastante sencillos y requieren una carga computacional pequeña. Tienen la capacidad de aprender, ya que a medida que el usuario se identifica a lo largo del tiempo el sistema detecta la evolución sufrida por la geometría de la mano. El principal atractivo de estos sistemas recae en la elevada aceptación y facilidad de uso por parte de los usuarios, a diferencia de otras modalidades como las

de huella, la geometría de la mano no tiene connotación policial. Sin embargo, la unicidad y estabilidad de la mano no han sido probadas en grandes poblaciones. Las tasas de falsa aceptación y falso rechazo son peores que en otras tecnologías, lo que ha provocado que este tipo de modalidades no se utilicen en entornos de alta seguridad, y la detección de mano viva depende de la realización de pruebas colaterales en la detección.

- **Iris, retina**

Los sistemas de identificación mediante patrones oculares pueden estar basados en dos esquemas diferentes, topografía del iris y topografía de la retina. Ambos esquemas aprovechan las ventajas que las características oculares en las que se basan ofrecen. La textura del iris presenta un elevado grado de unicidad, muy superior al resto de tecnologías biométricas comúnmente empleadas. Además la protección que le ofrece la córnea hace que dicho patrón permanezca inalterable durante la vida del sujeto. Por otra parte, el patrón de los vasos sanguíneos de la retina presenta una mayor unicidad que el patrón de iris, esto, unido a la casi imposible modificación de esta característica convierten a la exploración de la retina en una modalidad biométrica aún más segura que la de reconocimiento de iris.

Los métodos de autenticación basados en patrones oculares presentan una tasa de falsa aceptación cercana a cero, debido a la muy elevada unicidad de las características biométricas que usan. El tejido ocular se degrada muy rápidamente tras la muerte del individuo, de manera que el sistema biométrico puede detectar con relativa facilidad si el ojo pertenece a un organismo vivo o no. Para evitar la utilización de imágenes de alta resolución de un ojo humano frente al sistema se llevan a cabo pruebas colaterales en la detección, como la comprobación de la circulación sanguínea (pulsaciones) en el tejido ocular.

Sin embargo, a pesar de todo ello, este tipo de sistemas no cuentan con una elevada aceptación por parte de los usuarios. En primer lugar el hecho de que el usuario tenga que acercar su rostro al sistema y que un haz de luz láser pase por su ojo provoca incomodidad en el usuario, y una cierta desconfianza inicial al uso del sistema (aunque en realidad el sistema solamente capte una simple fotografía). Es por eso por lo que esta tecnología solo se ha implantado en entornos de extrema seguridad, donde el grupo de usuarios es muy reducido, y se es consciente del nivel de seguridad requerido. Otro inconveniente importante es que el estudio del ojo puede ser altamente intrusivo, ya que puede desvelar información privada que los usuarios no tienen por qué querer dar, como ciertas enfermedades o el consumo de alcohol o drogas. Es importante destacar el elevado coste de este tipo de sistemas ópticos.

- **Rostro**

El método de reconocimiento facial es el método de identificación que de manera más natural y con mayor frecuencia realiza nuestro cerebro, ya que a diario necesitamos reconocer a las personas que nos rodean. Los sistemas de identificación por rostro disponen de una cámara que graba al usuario y analizan sus características faciales para realizar la identificación. Este método puede resultar enormemente cómodo para el usuario, ya que puede pasar incluso inadvertido para él. Sin embargo presenta un gran inconveniente, que es la variabilidad de las características del rostro del sujeto a lo largo del tiempo. Factores como la edad, la expresión o simples cambios en el rostro (peinado, barba, gafas) dificultan el proceso de identificación.

- **Oreja**

Estudios forenses han demostrado que la oreja de un ser humano posee multitud de características que son propias del mismo, lo que significa que pueden ser utilizadas para su identificación. Esta modalidad, de estudio bastante reciente, requiere que el usuario descubra su oreja frente a una cámara, lo que puede resultar bastante intrusivo para determinadas culturas, e incómodo para personas de pelo largo.

- **Voz**

El reconocimiento de voz es una modalidad de identificación biométrica bastante estudiada que utiliza la voz del sujeto para realizar su identificación. Existen multitud de algoritmos, tanto para obtener los rasgos característicos de la voz, como para realizar el proceso de comparación con los patrones almacenados. Determinadas aplicaciones requieren que el sujeto pronuncie de la manera más fiel posible un código de acceso previamente grabado, ya sea un número secreto o simplemente su nombre y apellidos. Este tipo de sistemas son muy sensibles a ataques de repetición mediante grabadoras de sonidos. Otras técnicas más avanzadas son independientes del texto pronunciado, y tratan de identificar ciertas características de la voz del usuario y no lo que realmente éste dice. En estas aplicaciones se invita al usuario a pronunciar una frase diferente cada vez que desee acceder al sistema.

La identificación mediante la voz es una modalidad de identificación de muy bajo coste, ya que no son necesarios equipos muy avanzados para ser llevada a cabo, y está bastante aceptada. En algunas aplicaciones, como servicios de atención telefónica, puede resultar inapreciable para el usuario. Sin embargo los sistemas de reconocimiento por voz presentan una serie de inconvenientes importantes: el tono de voz del usuario puede sufrir grandes cambios como

consecuencia de su estado de ánimo, la edad o simplemente enfermedades tan comunes como el resfriado o la afonía. Este tipo de factores repercuten directamente en la calidad del sistema, ya que pueden producir falso rechazo. Para un óptimo funcionamiento del sistema se requiere un entorno con una buena acústica y carente de ecos y ruidos externos, de manera que las interferencias se reduzcan al máximo. Esto no siempre es posible, por lo que se buscan algoritmos que minimicen el efecto de estos factores.

- **Andadura, dinámica de teclado y firma**

Todas ellas son modalidades de identificación biométricas basadas en características de comportamiento, y por tanto susceptibles de ser imitadas.

Las técnicas de reconocimiento de andadura analizan la manera particular en la que un individuo camina, son técnicas de desarrollo muy reciente y presentan como principal inconveniente que no son aplicables a personas que no pueden caminar, ni tampoco en aquellas otras que presenten una lesión que les impida caminar con normalidad.

Las técnicas de dinámica de teclado utilizan para la identificación el ritmo característico con el que una persona es capaz de escribir con un teclado. Los estudios realizados han determinado un alto grado de unicidad en este comportamiento, sin embargo, además de los problemas de imitación que toda modalidad de identificación basada en comportamiento presenta, tiene la limitación de no ser aplicable a usuarios con dificultades a la hora de teclear.

El reconocimiento de firma se trata de una modalidad profundamente estudiada, y utilizada desde hace mucho tiempo como método de identificación de personas. La evolución de la tecnología ha intentado solventar los problemas de falsificación siempre ligados a ella, y hoy en día se habla de técnicas de verificación de escritura, en las cuales mediante un lápiz especial se analizan características dinámicas de la escritura del individuo, como el tiempo necesario para firmar, la presión del lápiz o el ángulo de colocación del mismo.

- **Olor**

Es una modalidad de desarrollo muy reciente que se basa en el análisis del olor corporal del individuo para su reconocimiento. Los sensores de olor, aún en desarrollo, utilizan un proceso químico similar al que se produce entre la nariz y el cerebro, de manera que agentes externos como otros olores o perfumes no enmascaren el olor particular de cada uno.

- **ADN**

El ADN es sin lugar a dudas la única característica biométrica que permite identificar unívocamente y sin ambigüedades a un sujeto. Ya que en dicho código está contenida toda la información genética del individuo, que es única e irrepetible. La principal limitación en la actualidad es la tecnología, ya que se requieren sistemas de identificación automática en tiempo real capaces de extraer las características genéticas del individuo, y que además resulten cómodos de usar para el usuario. Como toda nueva tecnología la identificación biométrica mediante ADN plantea ventajas evidentes, pero también riesgos, en este caso sobre todo intrusivos, como la invasión de la privacidad y el control exhaustivo que produciría el disponer del código genético completo de un usuario.

3. Tarjetas inteligentes

Como se ha comentado en el capítulo de introducción, en este proyecto se hará uso de un lector de tarjetas para conseguir el número de serie de tarjetas inteligentes y posteriormente mandarlo a un PC. En este capítulo se hará una breve introducción a las tarjetas inteligentes para comprender su funcionamiento y entender los pasos llevados a cabo para conseguir el número de serie, que se explicarán en detalle en el apartado 6.2.

3.1. Antecedentes a la tarjeta inteligente

Las tarjetas son un elemento de identificación portátil, muy extendido, que puede ser utilizado como medio de pago.

Existen diversas tecnologías en las que pueden estar basadas las tarjetas:

- **Tarjetas de Papel**

Las tarjetas de papel más conocidas son las Tarjetas de Visita: No permiten trasvase electrónico de la información. Se deterioran fácilmente. No está normalizada su forma.

- **Tarjetas de Plástico**

A principios de los años 50 surgió la tarjeta de plástico: Mayor duración y mejor imagen. Permitía procesado por bacaladera y OCR. Tamaño normalizado

- **Tarjeta de Banda Magnética**

A la Tarjeta de Plástico, se le añadió una Banda de Material Ferromagnético, para evitar el fraude y facilitar la transmisión electrónica de información. Su uso se ha extendido mundialmente gracias a la adopción de esta tecnología por la Banca Mundial. Se basa en el mismo mecanismo de las cintas de música o de vídeo.

Características positivas:

- ✔ Bajo precio.
- ✔ Conocido y aceptado por el usuario

Características negativas:

- ✗ Los datos están grabados superficialmente
- ✗ Fácil copia y muy posible fraude
- ✗ La banda magnética puede borrarse por campos electromagnéticos existentes (por ejemplo, salas de RMN)
- ✗ Terminales caros.
- ✗ Capacidad reducida de almacenamiento (225 caracteres)

• Tarjetas Ópticas

En 1981, Jerome Drexler creó la denominada *LaserCard*, también conocida como Tarjeta Óptica o Tarjeta Láser. Su tecnología es análoga a la de los CD-ROM.

Esta tecnología vino empujada por:

- El avance conseguido en el desarrollo de la tecnología óptica.
- La escasa capacidad de las Tarjetas de Banda Magnética.
- El coste y la no muy alta capacidad de las Tarjetas con Circuito Integrado.

Ventajas:

- ✔ Alta capacidad de almacenamiento (hasta 4,1 MB).
- ✔ Drivers que simulan la re-escritura en la tarjeta (realmente no se puede recuperar el espacio borrado).
- ✔ Resistentes a campos electromagnéticos.

Inconvenientes:

- ✗ La información está grabada superficialmente.
- ✗ Copia fácil y muy posible fraude.
- ✗ Los terminales son mucho más caros.
- ✗ Por motivos comerciales, las tarjetas son más caras que las de circuito integrado.
- ✗ No es una tecnología aceptada por la mayoría de los sectores y países (sólo la ven atractiva en Japón o parte del sector sanitario).

• Tarjetas con Circuito Integrado:

- Pueden ser de memoria o de microprocesador (inteligentes)
- Pueden ser con contactos o con comunicación a distancia.

3.2. Arquitectura de la tarjeta inteligente

Se denomina Tarjeta con Circuito Integrado aquella que tiene un chip empotrado en su interior. Su nacimiento se fecha, por un lado, en 1970 en Japón, y por otro, en 1974 en Francia. Debido al éxito comercial, es ésta última fecha la que se toma como origen, y al periodista francés *Roland Moreno*, como su inventor. Se fundó la empresa *Innovatron* para extender la tarjeta (y sus patentes) por todo el mundo.

A finales de los años 70 se ven los primeros productos, empujados por diversos proyectos de innovación tecnológica en Francia (banca, telefonía pública, sanidad, universidad, etc.). Aparecieron dos líneas diferenciadas:

- **Tarjetas de Memoria**
  - Son conocidas por su utilización como Tarjeta de Pre-pago Telefónico en cabinas.
  - Proporcionan una seguridad media-baja, aunque muy superior a las de Banda Magnética y/o Ópticas.
  - Tienen un coste superior a la banda magnética pero inferior a las Ópticas y a las Tarjetas Inteligentes.
  - Su capacidad de almacenamiento está en unos 4KB, aunque puede ser superior (no sigue creciendo por razones comerciales).
  - Su uso principal es para pre-pago con poca seguridad y programas de fidelización.
- **Tarjetas con Microprocesador: Tarjetas Inteligentes**
  - Su mayor difusión actual viene dada por los denominados Monederos Electrónicos y, sobre todo, por la telefonía GSM (como módulo SIM).
  - Proporcionan una seguridad muy alta, pudiendo hacer incluso criptografía pública, o tomar decisiones como auto bloquearse.
  - Su capacidad de almacenamiento está en unos 4 – 16 KB.
  - Su coste es elevado para muchas aplicaciones, pero los lectores son muy baratos.
  - Fracasos en distintos lanzamientos, han provocado un cierto rechazo por parte de la población.
  - Permiten realizar multi-aplicación y multi-proveedor.

Una Tarjeta Inteligente es una Tarjeta con un chip empotrado en su interior, que tiene los siguientes bloques:

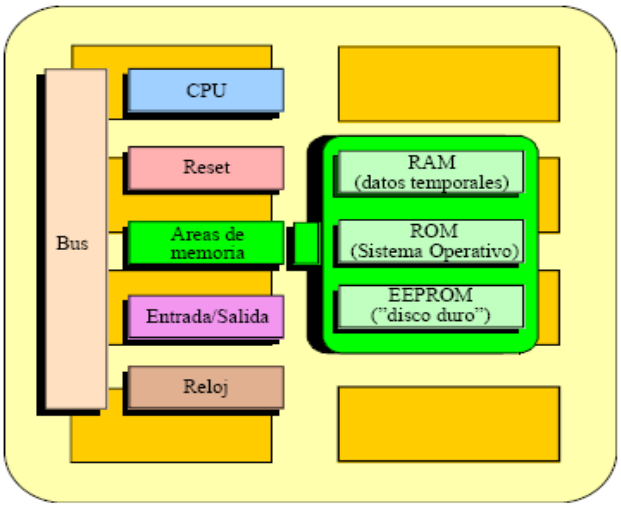


***Fig. 3.1**: Bloques de una tarjeta inteligente (Fuente: [9]).*

- Unidad central de proceso (CPU)

La CPU es el componente más importante de la tarjeta inteligente, se encarga de ejecutar las operaciones a bajo nivel, interconectando entre sí los distintos bloques funcionales de la tarjeta y tomando las decisiones necesarias. El sistema operativo de la tarjeta permite controlar el conjunto de operaciones que se pueden realizar dentro de la tarjeta, de manera que el bloque de la CPU pasa prácticamente desapercibido para el usuario final. Tradicionalmente la CPU de una tarjeta inteligente se ha basado en microprocesadores de 8 bits, como la familia 8051 de Intel, o la familia 68HC05 de Motorola. En otros ámbitos como en los ordenadores personales los microprocesadores han sufrido grandes evoluciones, apareciendo procesadores de 64 bits. En el campo de las tarjetas inteligentes la evolución no ha sido tan notoria, aunque han aparecido procesadores de 16 bits, e incluso algunos de 32 bits con tecnología RISC (Reduced Instruction Set Code), sobre todo en tarjetas por radiofrecuencia, la mayoría de las tarjetas siguen utilizando procesadores de 8 bits. La explicación es simple, las aplicaciones habituales de las tarjetas inteligentes no necesitan mayor potencia de cálculo para realizar satisfactoriamente su tarea, y el precio de cambio de adoptar procesadores más potentes resulta demasiado elevado. Por otra parte la evolución de la tecnología ha permitido la aparición de coprocesadores matemáticos en ciertas tarjetas, que permiten realizar los cálculos de forma mucho más rápida. También se han introducido bloques adicionales con funcionalidades criptográficas, así los algoritmos de cifrado requeridos para operaciones seguras pueden ser realizados en muy poco tiempo, una cantidad mucho menor que la necesaria si todo el cálculo lo tuviera que realizar directamente el microprocesador.

- Memoria

La memoria de una tarjeta inteligente se compone de varios bloques de memoria, cada uno de los cuales puede ser de un tipo diferente de memoria, según la función que desempeñe. Habitualmente se distinguen los siguientes tipos de memoria:

- **Memoria de acceso aleatorio** (RAM): este tipo de memoria es volátil, es decir su contenido se pierde cuando cesa la alimentación. Es utilizada por el microprocesador para ejecutar las instrucciones, depositando en ella resultados intermedios y también datos de entrada/salida. Esta memoria suele ser transparente tanto para el programador como para el usuario final, ya que no es directamente accesible, y es el propio sistema operativo de la tarjeta inteligente el encargado de gestionarla. Habitualmente el tamaño de la memoria RAM es de muy pocos kilobytes, lo que significa que su uso tiene que estar muy optimizado.
- **Memoria de solo lectura** (ROM): esta memoria contiene el código del sistema operativo de la tarjeta inteligente. Su tamaño suele ser de unos pocos kilobytes, de manera que el sistema operativo se desarrolla en un lenguaje de bajo nivel para aprovechar mejor el espacio disponible. La memoria ROM también es transparente, lo que impide cualquier tipo de modificación o lectura por parte de un usuario malintencionado, y aporta seguridad a las operaciones realizadas.
- **Memoria de solo lectura reescribible eléctricamente** (EEPROM): se trata de la única memoria accesible directamente tanto para el programador como para el usuario final, en ella se almacenan datos necesarios para realizar las operaciones. Su tamaño, aunque mayor que las anteriores, suele ser también del orden de kilobytes, y puesto que es la única memoria aprovechable por el usuario su tamaño es el que se asocia al tamaño de la memoria de la tarjeta. Gracias al sistema operativo de la tarjeta esta memoria no volátil se organiza en ficheros y directorios, resultando más sencilla su utilización.

Al contrario que sucede con la CPU, en las memorias utilizadas en las tarjetas inteligentes sí que se ha producido una notable evolución en las prestaciones. En sus inicios estas tarjetas no contaban con memorias reescribibles, ya que o eran muy caras o la tecnología no estaba lo suficientemente desarrollada, de manera que las tarjetas eran de usar y tirar. La evolución tecnológica y el abaratamiento de los costes permitieron introducir las memorias EEPROM, y las tarjetas pudieron ser borradas y escritas un elevado número de veces. Posteriores mejoras tecnológicas redujeron los tiempos de borrado y escritura, y consiguieron aumentar la escala de integración incrementando la capacidad de almacenamiento.

- Bloque de entrada y salida

El bloque de entrada y salida permite la comunicación de la tarjeta inteligente con el exterior. Esta comunicación se realiza de forma serie, para ello el bloque obtiene los datos uno tras otro, los trata y los entrega a la CPU para que los procese. Para realizar este procedimiento el sistema operativo de la tarjeta debe disponer de unas rutinas de control muy robustas, de modo que la comunicación se realice según los protocolos de comunicación específicos de las tarjetas. Este bloque difiere de unas tarjetas a otras según la tarjeta sea con contactos o sin contactos. Si la tarjeta es sin contactos la comunicación a nivel físico se realiza por radiofrecuencia, de manera que la tarjeta tendrá que modular y demodular la señal de información, además de compensar en la medida de lo posible las posibles interferencias.

- Sistemas de control de la alimentación

Este bloque supervisa los niveles de alimentación de la tarjeta, de manera que se encuentren en todo momento dentro de unos umbrales preestablecidos, lo que garantiza la seguridad tanto física como lógica de la información almacenada en la tarjeta. Si la alimentación supera en algún momento un límite determinado este bloque debe cortar la alimentación de la tarjeta, para así evitar posibles daños a la CPU o a las memorias. Si por el contrario se produce una caída de tensión este bloque debe ser capaz de mantener la tensión de alimentación el tiempo suficiente para interrumpir a la CPU, y que ésta guarde el estado de ejecución.

- Circuito de arranque

Este bloque permite enviar a la tarjeta una señal de reset, que inicialice la CPU con unos valores por defecto que garanticen el correcto funcionamiento de la tarjeta. Durante el funcionamiento normal de la tarjeta puede enviarse una señal de reset para que ésta vuelva a sus condiciones de inicio.

- Sistema de supervisión del reloj

Se encarga de vigilar el reloj de funcionamiento de la CPU entregado desde el exterior, de modo que para un correcto funcionamiento éste debe tener unos niveles y una estabilidad determinados. Si este bloque detecta alguna anomalía en el reloj suministrado puede interrumpir a la CPU para que cese su funcionamiento.

Por tanto se puede decir que una Tarjeta Inteligente es como un ordenador de propósito específico:

- Tiene una CPU y una memoria RAM.
- Tiene un elemento que funciona como el disco duro (EEPROM).

Para el caso con contactos: tiene 8 contactos, de los que se usan 5 (Alimentación, Masa, Reset, Reloj y Entrada/Salida). La alimentación puede ser de 5 o 3 V.

Para el caso de sin contactos, pueden ser:

- De memoria: se les llama normalmente Tags
- Inteligentes: son las que se conocen por Tarjetas Sin Contactos

Sustituyen la comunicación a través de 5-8 contactos eléctricos, por una comunicación:

- Por acoplo capacitivo o inductivo: Consiguiendo distancias de comunicación cercanas a los 0 cm. Se denominan Tarjetas de Acoplo
- Por RF de baja potencia: Alcanzando distancias de 0 – 10 cm. Se denominan Tarjetas de Proximidad
- Por RF de alta potencia: Alcanzando distancias superiores a los 10 cm. Han recibido el nombre de Tarjetas de Vecindad

Tiene una antena por la que se envían y reciben los datos a través de RF. De la misma forma se realiza el Reset. La alimentación se toma de la corriente inducida en la antena por el campo de RF. De la frecuencia de ese campo se extrae el Reloj.

3.3. Sistema operativo de la tarjeta inteligente (SOTI)

Lo que hace realmente importante a una Tarjeta Inteligente, es que es un elemento activo, basado en un Sistema Operativo (SOTI)

Lo que el SOTI deja ver de la tarjeta, es solamente:

- Sus Estructuras de Datos
- Su Arquitectura de Seguridad
- Su Juego de Comandos
- La memoria EEPROM (nuestro disco duro) disponible

Además hay que tener en cuenta que un mismo SOTI puede ser desarrollado sobre diversos microprocesadores y un mismo microprocesador puede contemplar distintos SOTIs.

Internamente, el SOTI se va a encargar de:

- Gestionar la Memoria: La organizará en Ficheros y Directorios y controlará las Estructuras de Datos dentro de los Ficheros
- Respetar una Arquitectura de Seguridad: Controlar el acceso a los ficheros, restringir las operaciones que se pueden hacer con los ficheros, no dejar que se ejecute nada que no esté contemplado, cifrar, descifrar, crear claves de sesión, ...
- Establecer un Juego de Comandos: para identificar las operaciones que se pueden hacer, y su modo de empleo.

Los SOTIs pueden clasificarse atendiendo a:

- SOTIs propietarios de propósito específico: Son Sistemas Operativos creados con una misión específica, tal como un monedero electrónico, o GSM.
- SOTIs propietarios de propósito general: Son Sistemas Operativos creados por cada uno de los fabricantes, con el objetivo de ser utilizado en prácticamente todas las aplicaciones. Cada fabricante suele tener al menos un SOTI propietario. Los SOTIs suelen ser muy distintos entre sí.
- SOTIs abiertos: Son Tarjetas que tienen su Sistema Operativo disponible para que el programador introduzca nuevos comandos, estructuras, etc., utilizando lenguajes como el MEL, C y Java.

3.4. Sistema de ficheros de una tarjeta inteligente

Tal y como se ha mencionado anteriormente el SOTI se encarga de todo el proceso de gestión de memoria, así el usuario lo que realmente ve es un sistema de ficheros. El SOTI puede permitir una estructura de ficheros lineal, de manera que solo puedan existir ficheros distribuidos en un mismo nivel, o una estructura jerárquica, permitiendo directorios que contengan ficheros u otros directorios. Los ficheros pueden referenciarse habitualmente mediante un código hexadecimal, un nombre largo o un nombre corto, la posibilidad de utilizar una u otra manera depende exclusivamente de la implementación del SOTI.

Se distinguen los siguientes tipos de ficheros en el sistema de ficheros de una tarjeta inteligente:

- **Fichero maestro** (MF): representa el directorio raíz de la tarjeta.
- **Fichero dedicado** (DF): representa un directorio, es decir un fichero especial que puede contener ficheros e incluso otros ficheros dedicados si el SOTI lo soporta.

- **Fichero elemental** (EF): son lo que normalmente denominamos ficheros. Pueden ser internos (de uso especial por el SOTI: ficheros de claves, ficheros de ATR, ficheros de respuesta, etc...) o de trabajo (donde se van a poder almacenar todos los datos de la aplicación determinada. Pueden distinguirse varios tipos de ficheros elementales según su uso dentro de la tarjeta. Así pues existen ficheros internos, como los ficheros de claves, que sólo pueden ser escritos desde el exterior y nunca leídos. Ficheros sin estructura, llamados transparentes, donde la información no se encuentra estructurada. Ficheros organizados en registros, donde la información está organizada en bloques, ya sean de longitud fija o variable. Y ficheros de trabajo, que son los que utiliza el usuario libremente según las condiciones de acceso establecidas en la aplicación.

3.5. Seguridad en la tarjeta inteligente

La Seguridad en una Tarjeta Inteligente viene dada por los siguientes elementos:

- Protección contra ataques físicos: jaulas de Faraday, detección de luz, ordenación pseudo-aleatoria de la memoria, etc.
- Evita su funcionamiento fuera de su rango de operación.
- No permite ejecutar comandos que no estén contemplados.
- Permite grabar claves en ficheros de sólo escritura.
- Controla el número de presentaciones erróneas de cada una de las claves, bloqueándola cuando se supera un límite.
- Impide el uso de claves, si no está prohibida su lectura.
- Cada clave puede restringir el acceso a los ficheros de datos, ya sea en lectura, en escritura o en actualización
- Un fichero puede tener distintos accesos controlados por distintas claves, de forma que un usuario pueda leer, pero no pueda actualizar, mientras que otro sólo pueda actualizar o hacer las dos cosas.
- Una clave de la tarjeta puede utilizarse como clave de cifrado de datos, ya sea para devolver los datos cifrados, o para guardarlos cifrados en la tarjeta.
- Puede cifrar la comunicación con el exterior, a través de claves que conozcan la tarjeta y el ente autorizado externo.
- Para que los cifrados no se hagan siempre con la misma clave, la tarjeta puede tener un mecanismo para generar *Claves de Sesión*, y que estas claves sean las que se utilicen para

cifrar la comunicación.

- Todo esto, organizado adecuadamente, sirve para hacer Autenticación Interna (que el terminal sepa si esa tarjeta es adecuada) y Externa (que la tarjeta sepa si el terminal es adecuado).

4. Protocolo USB

Como se ha comentado en el capítulo de introducción los lectores de tarjetas irán conectados al microcontrolador vía USB. En este capítulo se hará una breve introducción al protocolo USB para comprender su funcionamiento y entender los pasos llevados a cabo para conseguir comunicarnos con los lectores, que se explicarán en detalle en el apartado 6.2.

4.1. Introducción a USB

4.1.1. Antecedentes

Uno de los grandes problemas que tienen los PCs es la escasez de determinados recursos, básicamente líneas de interrupción IRQs y canales de acceso directo a memoria DMAs. En ambos casos las capacidades del diseño inicial tuvieron que ser dobladas en 1984, tres años después de su lanzamiento, aprovechando la aparición de la gama AT (*Advanced Technology)*, un formato de placa base.

La instalación de periféricos ha sido un constante quebradero de cabeza para los ensambladores, que debían asignar los escasos recursos disponibles entre los dispositivos del sistema. Aunque el estándar PnP (Plug-and-Play) vino a aliviar en parte las dificultades mecánicas de cambiar *jumpers* en las placas, el problema seguía ahí, ya que desde la aparición del AT el diseño del PC no había sufrido cambios sustanciales.

Por otra parte, a pesar de que habían persistido desde los inicios del PC, y de su conveniencia para multitud de aplicaciones, los puertos serie y paralelo presentaban claras limitaciones en cuanto a capacidad de expansión y rendimiento se refiere.

Por estas razones, y como resultado de un intento de dotar al PC de un bus de alta velocidad que ofreciera las características ideales (PnP, universalidad, facilidad de conexión y desconexión...), un consorcio formado por multitud de empresas (las empresas que formaron el grupo inicial fueron siete: Compaq, Intel, IBM, Microsoft, Nec, Northern Telecom y Digital Equipment), desarrolló una nueva interfaz estándar para la conexión de dispositivos externos del

PC, el denominado puerto USB. Es un bus serie bidireccional y de bajo coste, diseñado como una extensión en la arquitectura estándar del PC, orientado principalmente a la integración de periféricos y, en sus orígenes, dirigido a la integración de dispositivos telefónicos CTI en los ordenadores (*Computer Telephony Integrations)*.

4.1.2. Descripción General

La especificación de USB proporciona una serie de características que pueden ser distribuidas en categorías. Estas características son comunes para todas las versiones:

- Fácil uso para los usuarios:
  - Modelo simple para el cableado y los conectores.
  - Detalles eléctricos aislados del usuario (terminaciones del bus).
  - Periféricos autoidentificativos.
  - Periféricos acoplados y reconfigurados de forma dinámica (*Hot Swappable*)
- Flexibilidad:
  - Amplio rango de tamaños de paquetes, permitiendo variedad de opciones de buffering de dispositivos.
  - Gran variedad de tasas de datos de los dispositivos, acomodando el tamaño de buffer para los paquetes y las latencias.
  - Control de flujo para el manejo del buffer construido en el protocolo.
- Ancho de banda isócrono:
  - Se garantiza un ancho de banda y bajas latencias apropiadas para telefonía, audio, …
  - Cantidad de trabajo isócrono que puede usar el ancho de banda completo del bus.
  - Control de flujo para el manejo del buffer construido en el protocolo.
- Amplia gama de aplicaciones y cargas de trabajo:
  - Adecuando el ancho de banda desde unos pocos kilobits por segundo hasta varios megabits por segundo.
  - Soporta tanto el tipo de transferencia isócrono como el asíncrono sobre el mismo conjunto de cables.
  - Conexiones múltiples, soportando operaciones concurrentes de varios dispositivos.
  - Soporta hasta 127 dispositivos físicos.
  - Soporta la transferencia de múltiples datos y flujos de mensajes entre el host y los

dispositivos.

- Robustez
  - Manejo de errores y mecanismos de recuperación ante fallos implementados en el protocolo.
  - Inserción dinámica de dispositivos.
  - Soporte para la identificación de dispositivos defectuosos.
- Implementación de bajo coste
  - Sub canal de bajo coste a 1.5 Mbps.
  - Conectores y cables de bajo coste.
  - Adecuado para el desarrollo de periféricos de bajo coste.

El bus USB puede trabajar en cuatro modos:

- Baja velocidad (USB 1.0): Tasa de transferencia de hasta 1,5 Mbps.
- Velocidad completa (USB 1.1): Tasa de transferencia de hasta 12 Mbps.
- Alta velocidad (USB 2.0): Tasa de transferencia de hasta 480 Mbps.
- Super alta velocidad (USB 3.0): Tasa de transferencia de hasta 4,8 Gbps.

4.2. Topología

La interconexión USB es la manera en la cual los dispositivos USB se conectan y comunican con el host, esto incluye: la topología del bus o el modelo de conexión entre los dispositivos USB y el host; los modelos de flujo de datos, es decir la forma en la que la información se mueve en el sistema entre los diversos elementos del mismo; la planificación USB que define la secuencia en la cual los dispositivos accederán al bus; y, finalmente, las relaciones entre las capas del modelo, y las funciones de cada capa.

La topología del bus USB adopta forma de estrella y se organiza por niveles. En un bus USB existen dos tipos de elementos: Anfitrión (host) y dispositivos. A su vez, los dispositivos pueden ser de dos tipos: Hubs y Funciones:

4.2.1. Host USB

El host es el sistema de computación completo, incluyendo el software y el hardware, sobre el cual se sostiene USB.

El host tiene la habilidad de procesar y gestionar los cambios de configuración que puedan ocurrir en el bus durante su funcionamiento. Gestiona el sistema y los recursos del bus (uso de la memoria del sistema, asignación del ancho de banda del bus y alimentación del bus). También ayuda al usuario con la configuración automática de los dispositivos conectados y reaccionando cuando son desconectados.

Un host puede soportar uno o mas buses USB. El host gestiona cada bus independientemente de los demás. Los recursos específicos del bus como el ancho de banda asignado son únicos a cada bus. Cada bus está conectado al host a través de un controlador del host.

Sólo hay un host en cualquier sistema USB. Desde la interfaz USB, hasta el sistema del host del ordenador, es lo que se llama controlador de host, y puede estar implementado como combinación de hardware, firmware o software. Integrado dentro del sistema del host hay un hub raíz que provee de un mayor número de puntos de acople al sistema.

Siempre que es posible, el software de USB usa la interfaz existente del sistema del host para gestionar las interacciones superiores.

4.2.2. Controlador del host USB

El controlador del host está formado por el hardware y el software que permite a los dispositivos USB ser conectados al host. Este controlador es el agente iniciador del bus, es decir, el que comienza las transferencias en el bus. El controlador del bus es el maestro en un bus USB. Otros buses como PCI (*Peripheral Component Interconnect*) permiten la presencia de múltiples maestros, donde cada uno arbitra sus accesos al bus. En la arquitectura USB sólo hay un controlador del host por cada bus USB, por eso no hay arbitraje para el acceso al bus.

El host USB interactúa con los dispositivos USB a través del controlador. Las funciones básicas del controlador del host son:

- Detectar la conexión o desconexión de dispositivos USB.
- Gestionar el flujo de control entre el host y los dispositivos.
- Gestionar el flujo de datos entre el host y los dispositivos.
- Coleccionar estadísticas de actividad y estado.
- Proveer una cantidad limitada de energía a los dispositivos conectados.

Hay dos implementaciones estandarizadas de la parte hardware de los controladores del

host USB. Ambas proporcionan la misma funcionalidad y rendimiento para la interconexión. Estas implementaciones son:

- El Universal Host Controller Inteface (UHCI) definido por Intel: la parte software tiene una gran responsabilidad para mantener el hardware en funcionamiento. Esto permite a esta implementación ser relativamente simple.
- Open Host Controller Interface (OpenHCI o OHCI) definido por Microsoft: la parte hardware tiene más responsabilidad en el mantenimiento del flujo de datos, para que la parte software tenga menos trabajo que hacer. Esta otra implementación tiende a ser más compleja.

4.2.3. Dispositivos USB

Los dispositivos USB pueden ser hubs que provean puntos de conexión adicionales a los existentes en el host, o bien, diferentes periféricos.

Cada dispositivo lleva consigo información que puede ser útil para identificar sus características. La información que describe al dispositivo se encuentra asociada con el canal de control. Esta información se divide en tres categorías:

- Estándar: Esta es la información cuya definición es común a todos los dispositivos USB. Incluye elementos como la identificación del fabricante, la clase, la gestión de energía....
- Clase: La definición de esta información varía dependiendo del aparato. Es una clasificación de los dispositivos en cuanto a sus prestaciones.
- USB Vendor: El fabricante del periférico puede poner aquí cualquier información deseada.

El software del host es capaz de determinar el tipo de dispositivo conectado, haciendo uso de esta información y de un direccionamiento individual. Todos los dispositivos USB son accedidos por una dirección USB, que es asignada de forma dinámica cuando se conecta. Cada aparato soporta, además, uno o más canales a través de los cuales el host puede comunicarse con el dispositivo. Una vez ha sido reconocido e identificado el dispositivo, el software del host puede hacer que los drivers del dispositivo apropiados obtengan el control del nuevo dispositivo conectado. Cuando desconectamos el dispositivo, la dirección puede ser reutilizada para el próximo dispositivo conectado.

- **Hubs**

Un bus tradicional puede ser dividido en segmentos de bus individuales conectados por puentes. Cada segmento tiene un número limitado de puntos de conexión debido a la limitada energía y la carga del bus. En la arquitectura USB, el número de puntos de conexión de dispositivos se expande añadiendo dispositivos únicos llamados hubs o concentradores. Estos dispositivos expanden la arquitectura del USB de dos formas:

- Incrementando los puntos de conexión a través de puertos adicionales.
- Proporcionando energía a los dispositivos.

Los hubs sirven para simplificar la conectividad USB desde la perspectiva del usuario y proporcionar robustez y complejidad a un precio relativamente bajo. Según esta arquitectura, se puede expandir el bus acoplando hubs adicionales, interconectando dichos hubs mediante enlaces. Cada hub convierte un punto simple de conexión en múltiples puntos, donde estos puntos de conexión se llaman puertos.

Otra de las funciones importantes de los hubs es la de aislar a los puertos de baja velocidad de las transferencias a alta velocidad, proceso sin el cual todos los dispositivos de baja velocidad conectados al bus entrarían en colapso. La protección de los dispositivos lentos de los rápidos ha sido siempre un problema serio dentro de las redes mixtas, como es USB.

El hub está compuesto por dos partes importantes:

- El Controlador del Hub: puede asemejarse a una pequeña CPU de supervisión de las múltiples funciones que debe desempeñar un hub.
- El Repetidor del Hub: tiene la función de analizar, corregir y retransmitir la información que llega al hub hacia los puertos del mismo. Mantiene una memoria consistente en varios registros de interfaz, que le permite sostener diálogos con el host y llevar adelante algunas funciones administrativas, además de las meramente operativas.

- **Funciones**

Una función es, dentro de la terminología USB, todos los dispositivos que pueden ser conectados al bus USB, a excepción de los hubs. Es un dispositivo capaz de transmitir y recibir datos de información de control en un bus USB. Suele conectarse como un dispositivo independiente enlazado por un cable a un puerto del hub, o directamente al sistema anfitrión. El común denominador de todas las funciones USB es su cable y el conector del mismo, diseñado y fabricado atendiendo a las especificaciones del bus, por lo que no hay que preocuparse por la

compatibilidad entre equipos de diferentes fabricantes. Son funciones típicas el ratón, el monitor, módem, etc. La Figura 1.4 las ilustra adecuadamente.

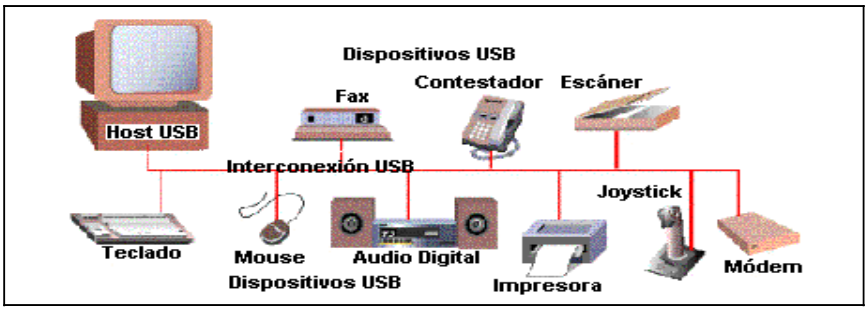


Fig. 4.1: *Funciones USB.*

Un aspecto interesante de las funciones, es que pueden ser a su vez nuevos hubs. De hecho, la figura siguiente muestra un esquema en el que el PC tiene tres puertos, el monitor cuatro, el teclado tres, y un hub provee 4 puertos más. En un esquema tan sencillo, existen 14 puertos disponibles para todo tipo de periféricos.

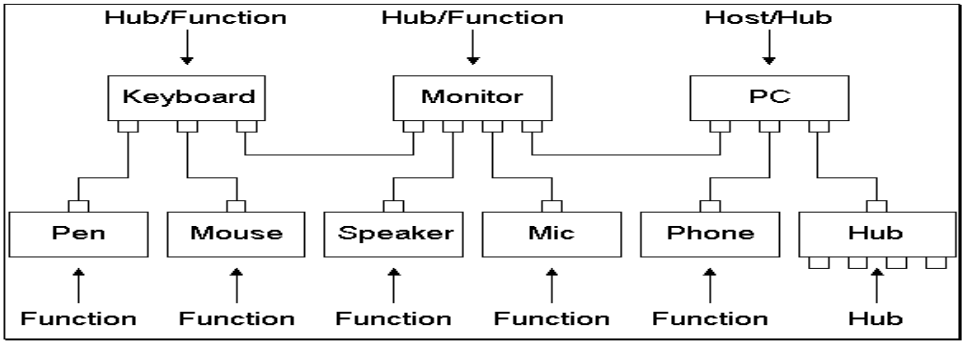


Fig. 4.2*: Esquema de interconexión USB.*

4.3. Flujo de datos

Un dispositivo USB, desde un punto de vista lógico, hay que entenderlo como una serie de endpoints (interfaz del dispositivo). A su vez, los endpoints se agrupan y dan lugar a un conjunto de interfaces (pipe), las cuales permiten controlar la función del dispositivo.

La comunicación entre el host y un dispositivo físico USB se puede dividir en tres niveles o capas. En el nivel mas bajo el controlador de host USB se comunica con la interfaz del bus utilizando el cable USB, mientras que en un nivel superior el software USB del sistema se comunica con el dispositivo lógico utilizando el pipe de control por defecto (*"Default Control*

Pipe"). En lo que al nivel de función se refiere, el software cliente establece la comunicación con las interfaces de la función a través de pipes asociados a endpoints.

4.3.1. Endpoints y direcciones del dispositivo

Cada dispositivo USB está compuesto por una colección de endpoints independientes, y una dirección única asignada por el sistema en el momento de la conexión de forma dinámica. A su vez, cada endpoint dispone de un identificador único dentro del dispositivo al que pertenece, a este identificador se le conoce como número de endpoint y viene asignado de fábrica. Cada endpoint tiene una determinada orientación de flujo de datos. La combinación de dirección, número de endpoint y orientación, permite diferenciar cada endpoint de forma inequívoca. Cada endpoint es por si solo una conexión simple que soporta un flujo de datos en una única dirección, bien de entrada o bien de salida.

Cada endpoint se caracteriza por:

- Frecuencia de acceso al bus requerida.
- Ancho de banda requerido.
- Número de endpoint.
- Tratamiento de errores requerido.
- Máximo tamaño de paquete que el endpoint puede enviar o recibir.
- Tipo de transferencia para el endpoint.
- La orientación en la que se transmiten los datos.

Existen dos endpoints especiales que todos los dispositivos deben tener, los endpoints con número 0 de entrada y de salida, que deben de implementar un método de control por defecto al que se le asocia el pipe de control por defecto. Estos endpoints están siempre accesibles mientras que el resto no lo estarán hasta que no hayan sido configurados por el host.

4.3.2. Pipes

Las vías de comunicación entre las aplicaciones que se ejecutan en el host y los distintos endpoints en los dispositivos USB son denominadas pipes. Cuando un dispositivo USB se conecta a un sistema, y el sistema lo reconoce y lo configura, el dispositivo queda organizado como un cierto conjunto de endpoints de distintos tipos. Entonces, el sistema establece todas los pipes necesarios entre el sistema y cada uno de los endpoints disponibles en dicha configuración. El

dispositivo puede implementar varias posibles configuraciones, con distintos tipos de transferencias en cada una de ellas.

Existen dos tipos de pipes:

- **Stream:** Es un pipe entre el host y un endpoint de tipo Bulk, Interrupción o Isócrono. Si un dispositivo necesita transferencias bidireccionales de un tipo de endpoint concreto, el sistema debe establecer dos pipes, uno de salida (con un endpoint de salida) y otro de entrada (con un endpoint de entrada).
- **Mensaje:** Es un pipe entre el host y los endpoints de Control en un dispositivo USB. Un endpoint es de salida y el otro es de entrada, de forma que se pueda establecer la comunicación bidireccional. Se denomina Pipe de Control por Defecto, y es el único pipe que se puede establecer antes de configurar al dispositivo. A través de este pipe, el sistema puede leer del dispositivo toda la información descriptiva necesaria para enterarse del tipo de dispositivo, posibles configuraciones, protocolos que soporta, número y tipos de endpoints que soporta en cada posible configuración. Esta información son los descriptores.

Además un pipe se caracteriza por:

- Demanda de acceso al bus y uso del ancho de banda.
- Un tipo de transferencia.
- Las características asociadas a los endpoints.

El software cliente normalmente realiza peticiones para transmitir datos a un pipe vía IRP (*I/O Request Packet*, paquete de entrada o salida) y entonces, o bien espera, o bien es notificado de que se ha completado la petición. El software cliente puede causar que un pipe devuelva todas las IRPs pendientes. El cliente es notificado de que una IRP se ha completado cuando todas las transacciones del bus que tiene asociadas se han completado correctamente, o bien porque se han producido errores.

Una IRP puede necesitar de varias tramas para mover los datos del cliente al bus. La cantidad de datos en cada trama será el tamaño máximo de un paquete, excepto el último paquete de datos que contendrá los datos que faltan. De modo que, un paquete recibido por el cliente que no consiga llenar el buffer de datos de la IRP, puede interpretarse de diferentes modos en función de las expectativas del cliente. Si esperaba recibir una cantidad variable de datos, considerará que se trata del último paquete de datos, sirviendo pues como delimitador de final de datos, mientras que si esperaba una cantidad específica de datos, lo tomará como un error.

4.3.3. Tipos de Transferencias

La interpretación de los datos que se transmitan a través de los pipes, independientemente de que se haga siguiendo o no una estructura USB definida, corre a cargo del dispositivo y del software cliente. No obstante, USB proporciona cuatro tipos de transferencia de datos sobre los pipes para optimizar la utilización del bus en función del tipo de servicio que ofrece la función. Estos cuatro tipos son:

- **Transferencias de control**

Es el único tipo de transferencia que utiliza pipes de mensajes, soporta, por lo tanto, comunicaciones de tipo configuración/comando/estado entre el software cliente y su función. Una transferencia de tipo control se compone de una transacción de setup del host a la función, de cero o más transacciones de datos en la dirección indicada en la fase de setup, y por último, de una transacción de estado de la función al host. La transacción de estado devolverá éxito cuando el endpoint haya completado satisfactoriamente la operación que se había solicitado.

Por lo tanto, este tipo de transferencia esta pensado para configurar, obtener información, y, en general, para manipular el estado de los dispositivos. El tamaño máximo de datos que se transmiten por el bus viene determinado por el endpoint. En dispositivos de velocidad media los posibles tamaños máximos son de 8, 16, 32 o 64 bytes; en velocidad baja el tamaño es de 8 bytes; y en velocidad alta de 64 bytes.

El endpoint puede estar ocupado durante la fase de envío de datos y la fase de estado, en esos casos el endpoint indica al host que se encuentra ocupado, invitando al host a intentar la comunicación más tarde. Si el endpoint recibe un mensaje de setup y se encontraba en mitad de una transferencia de control, aborta la transferencia actual y pasa a la nueva que acaba de recibir. Normalmente el host no inicia una nueva transferencia de control con un endpoint hasta que no ha acabado la actual. Pero, debido a problemas de transmisión, el host puede considerar que se han producido errores y pasar a la siguiente. USB proporciona detección y recuperación, vía retransmisión, de errores en las transferencias de control.

- **Transferencias isócronas**

Hacen uso de pipes stream. Garantiza un acceso al bus USB con una latencia limitada. Asegura una transmisión constante de los datos a través del pipe siempre y cuando se suministren datos. Además, en caso de que la entrega falle debido a errores, no se intenta reenviar los datos.

USB limita el máximo tamaño de datos para los endpoints con tipo de transferencia isócrona a 1023 bytes para los endpoints de velocidad media, y a 1024 bytes para velocidad alta. De

hecho, las transferencias isócronas sólo se pueden usar en dispositivos de velocidad alta o media. En función de la cantidad de datos que se estén transmitiendo en un momento dado, en velocidad media el porcentaje de trama utilizado puede variar desde un 1% hasta un 69%, mientras que el porcentaje de microtrama utilizado en velocidad alta varía entre un 1% y un 41%.

- **Transferencias de interrupción**

Hacen uso de pipes stream. Este tipo de transferencia está diseñado para servicios que envían o reciben datos de forma infrecuente. Esta trasferencia garantiza el máximo servicio para el pipe durante el periodo en el que envía. En caso de error al enviar los datos se reenvían en el próximo periodo de envío de datos.

El tamaño de paquete de datos máximo es de 1024 bytes para alta velocidad, de 64 bytes para velocidad media, y de 8 bytes para baja velocidad. En ningún caso se precisa que los paquetes sean de tamaño máximo, es decir, no es necesario rellenar los paquetes que no alcancen el máximo. Cuando en una transferencia de interrupción se necesite transmitir más datos de los que permite el paquete máximo, todos los paquetes, a excepción del último, deben de tener el tamaño máximo. Así, la transmisión de un paquete se ha llevado a cabo cuando se ha recibido la cantidad exacta esperada, o bien, se ha recibido un paquete que no alcanza el tamaño máximo. El porcentaje de microtrama utilizado ronda el 13% en velocidad baja y el 25% en velocidad media, mientras que en velocidad alta, para cantidades similares utilizadas para obtener los anteriores porcentajes, se obtienen resultados del 1%, pero para cantidades muy superiores se puede llegar a una utilización del 42%.

- **Transferencias Bulk**

Hacen uso de pipes stream. Esta diseñado para dispositivos que necesitan transmitir grandes cantidades de datos en un momento determinado, sin importar mucho el ancho de banda disponible en ese momento. Esta transferencia garantiza el acceso al USB con el ancho de banda disponible, además, en caso de error se garantiza el reenvío de los datos. Por lo tanto, este tipo de transferencia garantiza la entrega de los datos, pero no un determinado ancho de banda o latencia.

El tamaño máximo de paquete de datos para velocidad media es de 8, 16, 32 o 64 bytes; mientras que en velocidad alta es de 512 bytes. Los dispositivos de velocidad baja no disponen de endpoints con este tipo de transferencia. No es necesario que los paquetes se rellenen para alcanzar el tamaño máximo. El porcentaje de trama utilizado en velocidad media en función del número de bytes enviados varía entre el 1% y el 5%, mientras que el porcentaje de microtrama en velocidad alta varía entre un 1% y un 5%, eso sí, teniendo en cuenta mayor cantidad de datos.

4.4. Capa de protocolo

Cuando se transmite una secuencia de bytes en USB se realiza en formato *"little-endian"*, es decir, del byte menos significativo al byte más significativo.

En la transmisión se envían y reciben paquetes de datos, cada paquete de datos viene precedido por un campo Sync y acaba con el delimitador EOP, todo esto se envía codificado insertando los bits de relleno necesarios. En este punto, cuando se habla de datos, se refiere a los paquetes sin el campo Sync ni el delimitador EOP, sin codificación, ni bits de relleno.

El primer campo de todo paquete de datos es el campo PID. El PID indica el tipo de paquete y, por lo tanto, el formato del paquete y el tipo de detección de errores aplicado al paquete.

4.4.1. Formato de los Campos

Los paquetes se dividen en campos, a continuación el formato de los diferentes campos.

- **Campo identificador de paquete (PID)**

Es el primer campo que aparece en todo paquete. El PID indica el tipo de paquete, y por tanto, el formato del paquete y el tipo de detección de error aplicado a este. Se utilizan cuatro bits para la codificación del PID, sin embargo, el campo PID son ocho bits, que son los cuatro del PID seguidos del complemento a 1 de esos cuatro bits. Estos últimos cuatro bits sirven de confirmación del PID. Si se recibe un paquete en el que los cuatro últimos bits no son el complemento a 1 del PID, o el PID es desconocido, se considera que el paquete está corrupto y es ignorado por el receptor.

- **Campo dirección**

Este campo indica la función, a través de la dirección, que envía o es receptora del paquete de datos. Se utilizan siete bits, de lo cual se deduce que hay un máximo de 128 direcciones.

- **Campo endpoint**

Se compone de cuatro bits e indica el número de enpoint al que se quiere acceder dentro de una función. Como es lógico este campo siempre sigue al campo dirección.

- **Campo número de frame**

Es un campo de 11 bits que es incrementado por el host en cada microtrama en una unidad. El máximo valor que puede alcanzar es el 7FFH, si se vuelve a incrementar pasa a cero.

- **Campo de datos**

Los campos de datos pueden variar de 0 a 1024 bytes.

- **Campo CRC (*Cyclic Redundancy Checks*)**

El CRC se usa para proteger todos los campos no PID de los paquetes de tipo token y de datos. Siempre es el último campo y se genera a partir del resto de campos del paquete, a excepción del campo PID. El receptor al recibir el paquete comprueba si se ha generado de acuerdo a los campos del paquete, si no es así, se considera que alguno o mas de un campo están corruptos, en ese caso se ignora el paquete. El CRC utilizado detecta todos lo errores de un bit o de dos bits.

4.4.2. Formato de los Paquetes

A continuación se describen los diferentes paquetes y sus formatos en función de los campos que lo forman.

- **Paquetes de tipo token**

Un token está compuesto por un PID que indica si es de tipo IN, OUT o SETUP. El paquete especial de tipo PING también tiene la misma estructura que token. Después del campo PID, sigue un campo dirección y un campo endpoint. Por último, hay un campo CRC de 5 bits.

En los paquetes OUT y SETUP esos campos identifican al endpoint que va a recibir el paquete de datos que va a continuación. En los paquetes IN indican el endpoint que debe transmitir un paquete de datos. En el caso de los paquetes PING hacen referencia al endpoint que debe responder con un paquete handshake.

- **Paquete inicio de trama (SOF)**

Estos paquetes son generados por el host cada 1 milisegundo en buses de velocidad media y cada 125 microsegundos en velocidad alta. Este paquete está compuesto por un campo número de trama y un campo de CRC.

- **Paquete de datos**

Este paquete está compuesto por cero o más bytes de datos seguido de un campo de CRC de 16 bits. Existen cuatro tipos de paquetes de datos: DATA0, DATA1, DATA2 y MDATA. El número máximo de bytes de datos en velocidad baja es de ocho bytes, en media de 1023 bytes y en alta de 1024 bytes.

- **Paquetes Handshake**

Los paquetes handshake se utilizan para saber el estado de una transferencia de datos, indicar la correcta recepción de datos, aceptar o rechazar comandos, control de flujo, y condiciones de parada. El único campo que contiene un paquete de este tipo es el campo PID.

Existen cinco paquetes de tipo handshake:

- **ACK:** Indica que el paquete de datos ha sido recibido y decodificado correctamente. ACK sólo es devuelto por el host en las transferencias IN y por una función en las transferencias OUT, SETUP o PING.
- **NAK:** Indica que una función no puede aceptar datos del host (OUT) o que no puede transmitir datos al host (IN). También puede enviarlo una función durante algunas fases de transferencias IN, OUT o PING. Por último se puede utilizar en el control de flujo indicando disponibilidad. El host nunca puede enviar este paquete.
- **STALL:** Puede ser devuelto por una función en transacciones que intervienen paquetes de tipo IN, OUT o PING. Indica que una función es incapaz de transmitir o enviar datos, o que una petición a un pipe de control no está soportada. El host no puede enviar bajo ninguna condición paquetes STALL.
- **NYET:** Sólo disponible en alta velocidad. Es devuelto como respuesta bajo dos circunstancias: como parte del protocolo PING; o como respuesta de un hub a una transacción SPLIT, indicando que la transacción de velocidad media o baja aún no ha terminado, o que el hub no está aún habilitado para realizar la transacción.
- **ERR:** Sólo disponible en alta velocidad. Forma parte del protocolo PING, permite a un hub de alta velocidad indicar que se ha producido un error en un bus de media o baja velocidad.

4.4.3. Transacciones

Los paquetes de tipo token tienen como objetivo indicar el inicio de una transacción de datos de una determinada forma.

- **Transacción IN**

La transacción empieza con el envío de un paquete de tipo IN por parte del host a un determinado endpoint en una función.

- **Transacción OUT**

El host envía un paquete OUT a un determinado endpoint y seguidamente envía el paquete de datos. Suponiendo que el endpoint ha decodificado correctamente el token, espera a recibir un paquete de datos, en caso contrario, se ignora el token y los datos.

- **Transacción SETUP**

La transacción SETUP es una transacción especial, tiene las mismas fases que una

transacción OUT. Sólo se puede utilizar con endpoints de tipo control y su finalidad es indicar el inicio de la fase setup.

4.4.4. Split

El token especial SPLIT es utilizado para poder soportar transacciones en varias tramas, entre un hub que trabaja a velocidad alta y dispositivos de velocidad media o baja. Se definen nuevas transacciones que utilizan el token SPLIT, en concreto dos transacciones con sus respectivos paquetes: "start-split transaction (SSPLIT)" y "complete-split transaction (CSPLIT)".

Transacciones Split

La transacción split es utilizada solamente por el host y un hub cuando se quiere comunicar con un dispositivo de velocidad media o baja conectado con el hub. El host puede iniciar una transacción por partes a través del paquete SSPLIT, y puede completarla, recibiendo las respuestas de la función de velocidad baja o media, a través del paquete CSPLIT. Esto permite al host realizar otras transacciones a velocidad alta sin tener que esperar a que acabe la transacción.

5. Protocolo TCP/IP

Como se ha comentado en el capítulo de introducción el microcontrolador se comunicará con el PC vía Ethernet mediante el protocolo TCP/IP. En este capítulo se hará una breve introducción al protocolo para comprender su funcionamiento y entender los pasos llevados a cabo para conseguir esta comunicación que se explicarán en detalle en el apartado 7.3.

5.1. Introducción a TCP/IP

5.1.1. Antecedentes

TCP/IP es la denominación que recibe una familia de protocolos diseñados para la interconexión de ordenadores, independientemente de su arquitectura y el sistema operativo que ejecuten. Son un estándar de facto debido a la expansión de Internet, la Red que conecta millones de máquinas por todo el mundo.

Internet deriva de una red inicial denominada ARPANET (*Advanced Research Projects Agency NET*), promovida por el Departamento de Defensa de los Estados Unidos. ARPANET unía edificios de universidades, organismos de investigación e instituciones militares. La red fue dividida a principios de 1984 en dos partes: ARPANET, dedicada a proyectos de investigación y MILNET (*Military Network*) para usos militares. La familia de protocolos TCP/IP nació en el seno de esta conjunto de máquinas interconectadas con una serie de objetivos de diseño.

5.1.2. Estructura Interna

Aunque esta familia de protocolos es conocida como TCP/IP, existe un mayor numero de elementos que la conforman. En la *Figura 5.1* podemos observar la relación existente entre ellos así como los niveles en los que se encuadran.

***Fig. 5.1**: Torre de Protocolos TCP/IP (Fuente: [10]).*

- **TCP** *(Transmission Control Protocol)*: Protocolo de Control de Transmisión. Sus características fundamentales se resumen en que es orientado a conexión y que proporciona mecanismos que nos ofrecen seguridad acerca de la entrega de los paquetes en destino, así como su capacidad de ordenación y no duplicación.
- **UDP** (*User Datagram Protocol*): Protocolo de Datagramas de Usuario. Es un protocolo no orientado a conexión. No garantiza que los datagramas sean entregados en destino.
- **ICMP** (*Internet Control Message Protocol*): Protocolo de Mensajes de Control en Internet. Utilizado para gestionar la comunicación de mensajes de error entre distintos puntos de la red.
- **IP** (*Internet Protocol. Prtocolo Internet*): Es el protocolo que proporciona el servicio de envío de paquetes para los protocolos soportados TCP, UDP e ICMP.
- **ARP** (*Address Resolution Protocol*): Protocolo de Resolución de Direcciones. Permite mantener asignaciones de pares formados por direcciones IP y direcciones físicas de dispositivos de comunicación.
- **RARP** (*Reverse Address Resolution Protocol*): Inversamente al anterior, permite mantener asignaciones de direcciones físicas - direcciones IP.

5.2. Nivel de Red: IP

5.2.1. Introducción

El nivel IP del conjunto de protocolos TCP/IP aporta un interfaz de comunicación no

orientado a conexión. Cada datagrama IP que se transmite por la red es independiente de los otros. Por lo tanto, cualquier asociación entre los datagramas para formar un mensaje completo debe ser aportado por los niveles superiores. El interfaz que aporta se denomina también no fiable, esto se traduce en que no garantiza que los datagramas IP vayan a ser recibidos correctamente. Al igual que la característica anterior, deberá ser aportada por los niveles superiores que lo requieran en sus especificaciones.

El protocolo aporta un sistema mínimo de detección de errores, en concreto se trata de un *checksum* que incluye la cabecera del datagrama. Si se encuentra un error la única acción que se genera es descartar el datagrama, suponiendo que un nivel superior (un protocolo que se apoye en IP) retransmitirá el paquete dañado.

El nivel IP es el encargado de realizar las tareas de encaminamiento y fragmentación de los paquetes. Si un nodo que actúe como pasarela recibe un datagrama IP demasiado extenso para transmitir a través de la red, el nivel IP segmenta el mensaje y envía cada segmento en varios paquetes IP. Estos paquetes son reensamblados en destino, sólo si lo alcanzan todos y cada uno de ellos. Si se produce alguna pérdida, todo el datagrama original es descartado.

5.2.2. Encaminamiento

En redes de tipo TCP/IP es necesario tomar una decisión cada vez que se necesita enviar un datagrama. Por ello dos datagramas que tengan el mismo origen y destino y sean consecutivos pueden seguir diferentes rutas hasta alcanzar el nodo receptor.

Podemos encontrar diferentes algoritmos de *encaminado de paquetes.* Básicamente distinguiremos dos clases:

- Encaminado estático: escoge una ruta para el datagrama basándose en información recogida en tablas diseñadas en un primer momento y que no son modificadas, salvo en casos de cambio de topología de la red, adición de nuevos nodos, etc.
- Encaminado dinámico: tiene en cuenta la carga actual de las distintas partes de la red. El sistema decide basándose en información actual. Esta información puede ser únicamente local, centralizada o distribuida a lo largo de los nodos, con la posibilidad de que éstos intercambien información para ajustar sus tablas de encaminamiento dinámico de forma precisa.

5.2.3. Pasarelas

Una pasarela es un sistema que interconecta dos o más *redes,* entendiendo como red un conjunto de nodos conectados mediante una LAN (*Red de Área Local*), y cuya función es transferir envíos de mensajes de una red a otra.

Si las dos redes interconectadas utilizan el mismo conjunto de protocolos, la pasarela únicamente necesita enviar los mensajes recibidos por uno de sus interfaces (conexión a una de las redes) a través de su otro interfaz. No necesita realizar un tratamiento del mensaje, ya que el nivel de red de la red destino sabrá encaminar adecuadamente el mensaje, puesto que proviene de un nivel de red de la misma familia de protocolos.

Sin embargo, este intercambio se complica si las redes interconectadas utilizan diferentes familias de protocolos. Se necesita un tratamiento del mensaje, que normalmente se realiza en los niveles de acceso a la red.

5.2.4. Resolución de direcciones

Si disponemos de una Red de Área Local (*LAN Local Área Network*) con tecnología Ethernet, y estamos usando protocolos de tipo TCP/IP en la comunicación de los nodos, disponemos de dos tipos de direcciones en nuestra red: direcciones IP (32 bits) y direcciones Ethernet (48 bits).

Al utilizar Ethernet como Interfaz Hardware de comunicación encontramos dos tipos de resolución de direcciones necesarias:

- Si conocemos la dirección IP de un nodo con el que queremos comunicar, ¿Cómo puede el protocolo IP determinar su dirección Ethernet?. Esta relación se conoce como resolución de dirección.
- Por otro lado, puede ser necesario establecer la relación contraria, es decir, determinar la dirección IP de un nodo conociendo únicamente su dirección Ethernet: resolución inversa de dirección.

Para la resolución se utilizan los protocolos nombrados en la introducción: *ARP* y *RARP:*

El protocolo ARP permite que el nodo emita un paquete de tipo broadcast a la red local. Esto significa que el paquete es recibido por cualquier máquina conectada a la red Ethernet. El paquete lleva la dirección IP de un nodo con el que queremos comunicar; únicamente este nodo contestará cuando reciba la petición ARP**.** La respuesta permite al nodo origen enviar a partir de

aquí paquetes directamente al nodo deseado, ya que conoce su dirección Ethernet. Además se establecerán sistemas de almacenamiento que guarden los pares Dirección IP - Dirección Ethernet para posteriores comunicaciones, evitando así tener que repetir el broadcast a toda la red de área local.

El protocolo RARP está diseñado para una red de área local que disponga de nodos sin sistema de almacenamiento. Dentro de esta LAN al menos uno de los nodos actuará de servidor RARP almacenando pares de relaciones Direcciones Ethernet-Direcciones IP. Cuando se inicializa un nodo sin disco en la Red de área local, éste conoce su dirección Ethernet (a través de su hardware de red) y envía un mensaje de tipo broadcast conteniendo su dirección Ethernet y preguntando por su IP equivalente. Únicamente el servidor RARP responde a este mensaje, indicando en la respuesta la dirección IP correspondiente.

5.3. Nivel de Transporte: TCP

5.3.1. Introducción

Normalmente los procesos que se ejecutan en la máquina no acceden directamente al Nivel de Red (IP), por el contrario, interactuan con un nivel superior de abstracción: El nivel de transporte. Este nivel, dentro del sistema de protocolos TCP/IP, permite a los procesos enviar y recibir datos a través de los protocolos TCP y UDP. Básicamente, TCP ofrece un servicio fiable y orientado a conexión, mientras que UDP ofrece un servicio no fiable y no orientado a conexión.

Al basarse en IP, podemos deducir a partir de estas especificaciones que el protocolo TCP debe aportar todo el servicio relacionado con la conexión y el cierre de la comunicación, así como la fiabilidad del servicio.

El protocolo UDP únicamente aporta dos características nuevas a IP:

- Número de puerto (multiplexación).
- Un *checksum* opcional. Un *checksum* es básicamente un campo introducido en el mensaje que permite verificar en destino que los datos no se han corrompido en la comunicación, aunque no es capaz de detectar el 100% de los fallos de transmisión.

TCP aporta las siguientes características:

- **Orientado a conexión.** Antes de transferir los datos, dos procesos de nivel de aplicación deben negociar formalmente una conexión TCP, utilizando el proceso de establecimiento de

conexión adecuado. Las conexiones de TCP se cierran formalmente empleando el proceso de desconexión TCP.

- **Full Duplex.** Para cada extremo de una conexión TCP, la conexión consta de dos enlaces lógicos, uno de salida y otro de entrada. Con la tecnología apropiada en el nivel de red, los datos pueden fluir simultáneamente en ambos sentidos. La cabecera TCP contiene tanto el número de secuencia de los datos de salida como el reconocimiento de los datos de entrada.
- **Fiable.** Los datos que se envían por una conexión TCP se numeran en secuencia y se espera un reconocimiento positivo por parte del receptor. Si no se recibe este reconocimiento, el segmento se retransmite. En el receptor, los segmentos duplicados se descartan y los segmentos que llegan fuera de secuencia se colocan en su posición dentro de la secuencia. Todo segmento transmitido va protegido frente a errores mediante un código detector (CRC), que verifica la integridad de la información recibida.
- **Flujo de bytes**. Para TCP los datos que se envían por los enlaces lógicos de entrada y salida se consideran un flujo continuo de bytes. El número de secuencia y de reconocimiento que se envían en cada cabecera TCP definen puntos concretos de este flujo de bytes. TCP no tiene en consideración otras divisiones dentro del flujo de datos, siendo el protocolo de aplicación el que establezca las divisiones lógicas adecuadas (por ejemplo, fin de registro ó de campo en bases de datos, fin de orden, etc.)
- **Control de flujo en ambos extremos.** Para evitar la transmisión de excesivos datos simultáneos, que podría causar problemas de congestión en los *routers*, TCP implementa un control de flujo en el emisor que regula la cantidad de datos que se envían. Para evitar que el emisor transmita datos que el receptor no es capaz de almacenar, TCP también implementa control de flujo en el receptor, indicando cuánto espacio se encuentra disponible en los *buffers* del receptor.
- **Segmentación de datos de aplicación**. TCP segmentará los datos obtenidos del proceso de aplicación para que se ajusten al tamaño de los paquetes IP. Ambos extremos TCP pueden negociar el tamaño máximo de segmento, existiendo además la posibilidad de ejecutar un algoritmo de descubrimiento del tamaño máximo en la ruta (PMTU).
- **Transmisión uno a uno**. Las conexiones TCP son un circuito lógico punto a punto entre dos procesos de nivel de aplicación. TCP no proporciona servicios de difusión. TCP emplea el mecanismo de los **puertos** para el acceso a diferentes destinos dentro de un *Host*. Al igual que UDP, la utilización de los puertos permite que un cliente seleccione el servidor

correspondiente a la aplicación deseada dentro del computador destino.

5.3.2. Fragmentación

La mayor parte de las niveles de acceso a una red tienen definido un tamaño máximo de información que pueden manejar, conocido con el acrónimo de MTU (*Maximun Transmission Unit*). El estándar **Ethernet,** que puede ser utilizado como nivel físico en la Torre de protocolos TCP/IP, establece que el campo de datos no puede exceder de 1500 bytes en la trama.

La fragmentación es la división de una trama en otras más pequeñas, de forma que éstas últimas puedan ser enviadas indivisiblemente a través del nivel físico utilizado. En los protocolos TCP/IP es el nivel IP quien se encarga de la fragmentación, cuando es necesaria, y del reensamblado de las sub-tramas.

En concreto, al enviar un paquete UDP si la longitud del datagrama IP generado (formado por los datos, la cabecera UDP y la cabecera IP) es mayor que el MTU de la capa de acceso a la red utilizada, será necesario que el protocolo IP realice una fragmentación del datagrama antes de enviarlo al nivel inferior. La capa IP que recibe las sub-tramas en el otro nodo deberá hacerse cargo del reensamblado de las sub-tramas en una única antes de transferirla al nivel superior.

TCP actúa de la misma manera, es decir, la fragmentación y el reensamblado de las tramas es responsabilidad del nivel de red IP. Sin embargo, las implementaciones del nivel TCP intentan evitar la fragmentación en el nivel IP por motivos de eficiencia.

5.3.3. Fases de TCP

Para garantizar que no se pierden bloques de datos, una sesión TCP consta de tres fases:

- **Establecimiento de conexión**: Durante esta fase los *hosts* origen y destino determinan unos parámetros necesarios para el intercambio de datos, como son el número inicial de secuencia, los tamaños de *buffer* necesarios, etc., creando un marco para el intercambio fiable de información.

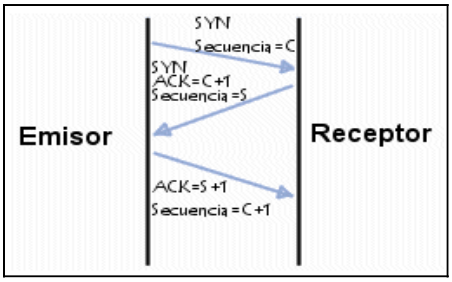


Fig. 5.2: *Inicio de conexión TCP (Fuente: [11])*

- **Transferencia de datos**: La información viaja en ambos sentidos fraccionada en segmentos de datos. En esta fase entran en juego los mecanismos de detección y corrección de errores, control de flujo y control de la congestión.
- **Cierre de conexión**: Tras el intercambio de información, y a propuesta de uno de los extremos, se intenta una desconexión negociada, donde no queden datos sin entregar por ninguna de las dos partes.

El flujo de datos es tratado como una secuencia de bytes *(stream)*, siendo responsabilidad del protocolo la decisión de cómo dividir (o agrupar) las unidades de datos de la aplicación a la hora de transferirlos.

5.3.4. Formato de segmentos TCP

A continuación se muestra el formato de un segmento TCP:

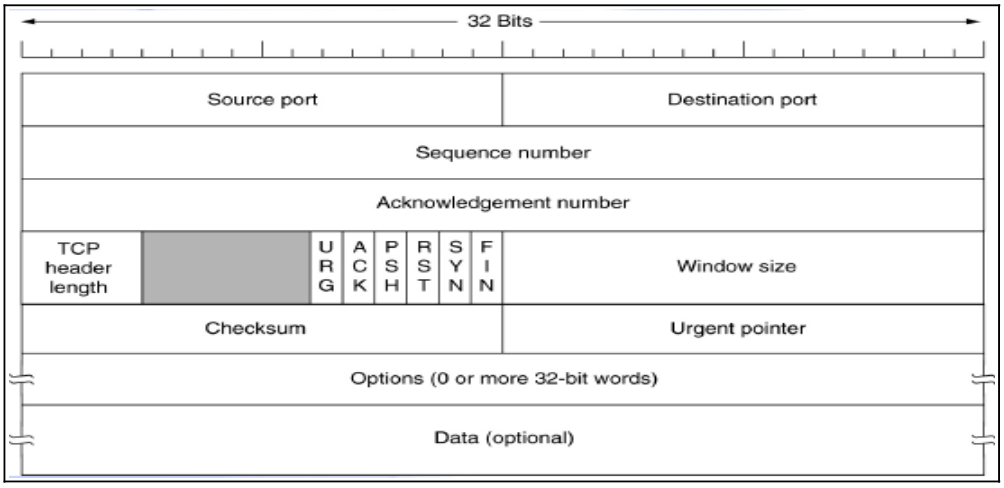


Fig. 5.3: *Formato de un segmento TCP (Fuente: [11])*

Los campos **Puerto Origen** (*Source Port*) y **Puerto Destino** (*Destination Port*) identifican, junto a las direcciones origen y destino del paquete IP, la conexión a la que pertenece el segmento.

El campo **Número de Secuencia** (*Sequence number*) indica el orden, dentro de la transmisión, del primer byte de datos contenido en el segmento. Junto al tamaño total del paquete IP, que permite conocer cuántos bytes de datos se reciben, es posible determinar cuál será el número de secuencia del siguiente segmento.

El campo **Número de Reconocimiento** (*Acknowledgement number*) indica cuál es el byte que el emisor del segmento espera recibir como número de secuencia. Esto implica que todos los bytes transmitidos en segmentos anteriores han llegado de forma satisfactoria. Sólo es significativo cuando el bit **ACK** del campo **Código** está a **1.**

El campo **Longitud de la Cabecera** (*TCP header length)*, indica cuántas palabras de 32 bits (4 bytes) componen la cabecera TCP. Como se comentará más adelante, en esta cabecera TCP pueden existir campos de tamaño variable. El tamaño mínimo del campo es 5 (la cabecera ocupa 5 * 4=20 bytes) y el máximo será de 15 (15 * 4 = 60 bytes). Tras el mismo existen unos bits **reservados**, que por omisión deben estar a cero.

El campo **código** consta de seis bits, teniendo cada uno de ellos un significado independiente del resto, tal y como se muestra en *Fig. 5.4*.

Bit	Significado si está a uno
URG	El puntero a datos urgentes es válido
ACK	El campo de reconocimiento es válido
PSH	Este segmento solicita un PUSH
RST	Reiniciar la conexión
SYN	Establecimiento de la conexión
FIN	El emisor llegó al final de su secuencia de datos

Fig. 5.4: *Campo código de un segmento TCP.*

El campo **Ventana** (*Window Size*) indica cuál es el tamaño de la ventana de recepción del emisor de segmento, es decir, el espacio en memoria disponible en el receptor. Este campo limita la ventana de transmisión del receptor del segmento, limitando el número de bytes que puede transmitir sin recibir reconocimientos.

El campo **Checksum** contiene un código de detección de errores.

El campo **Puntero a Datos Urgentes** *(Urgent Pointer)* permite, junto al bit **URG**, la

transmisión de datos urgentes no sujetos al control de flujo.

Finalmente, pueden existir una o varias **Opciones** *(Options)*, que permiten gestionar aspectos como el tamaño máximo de segmento, los reconocimientos selectivos, etc. Por este campo es por lo que existe el campo Longitud de Cabecera, ya que su tamaño es variable.

6. Diseño del Sistema

En este capítulo se hace una descripción general de los dispositivos utilizados en el sistema final. Además de los dispositivos propios del sistema, como son la placa de desarrollo y los lectores, descritos en el primer apartado de este capítulo, se ha tenido que utilizar un analizador de protocolo USB, que se describirá en el segundo apartado, en el que se hará uso de los conocimientos adquiridos del capítulo Protocolo USB y del capítulo Tarjetas Inteligentes.

6.1. Descripción de los dispositivos

En este apartado se describen las características generales de los dispositivos utilizados en el sistema final. Primero se comentarán las características de la placa de desarrollo con la que se ha trabajado, después las de los lectores de tarjetas, que se conectarán a la placa vía USB, y las del sensor de huella, que se conectará por SPI; y finalmente las de la pantalla táctil, que se conectará a la placa vía SPI.

6.1.1. Placa de Desarrollo: MCB2470

Como ya se comentó en el capítulo de introducción, vamos a necesitar un microcontrolador que sea capaz de comunicarse por varias interfaces, tales como USB, SPI, y Ethernet. Para facilitar el trabajo se ha elegido la placa de desarrollo MCB2470, que se puede ver en la figura 6.1, que tiene incorporados el módulo de USB, el módulo de Ethernet, la interfaz SPI, y la pantalla táctil.

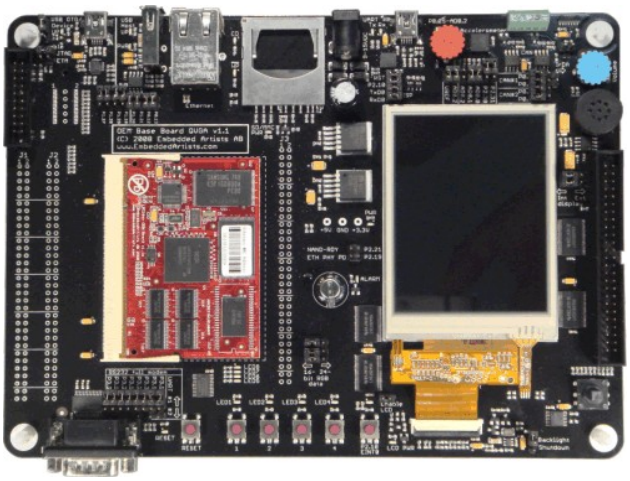


***Fig. 6.1**: Placa de Desarrollo MCB2470 (Fuente: [14]).*

La placa de desarrollo MCB2470 OEM, fabricada por Embedded Artists, está basada en un microcontrolador de NXP, el LPC2478. Este microcontrolador tiene un procesador ARM7TDMI-S con velocidades de hasta 72 Mhz, una memoria FLASH interna de 512 Kbytes, y una memoria SRAM interna de 98 KBytes. Ésta última está repartida de la siguiente manera: 64 KBytes sobre el bus local para acceso de alto rendimiento de la CPU, 16 KBytes para la interfaz Ethernet, y 16 KBytes para la interfaz USB.

El LPC2478 incorpora, además, lo siguiente:

- Un cristal RTC de 32.768 Khz.
- 160 pines GPIO.
- Periférico Ethernet MAC interno de 100/10Mb con interfaz MII/RMII y asociado al controlador DMA.
- 256 Kbit de memoria no volátil accesible vía interfaz I2C.
- Una interfaz SPI.
- Controlador USB 2.0 en modo device, host y OTG, con PHY on-chip y asociado al controlador DMA.
- Cuatro puertos serie de tipo UART. Uno soporta IrDA, otro con control I/O, y todos con FIFO.
- Un controlador de LCD que soporta pantallas de tipo STN y de tipo TFT. Acepta resoluciones de hasta 1024 x 768 píxeles y soporta hasta RGB-24.
- Un conversor ADC y otro DAC.

- Controlador CAN con dos canales.
- Cuatro timers con ocho entradas de captura y diez salidas de comparación.
- Dos PWMs.

La placa de desarrollo dispone de dos memorias FLASH externas, una NAND de 128 MBytes y otra NOR de 4 Mbytes; y una memoria SDRAM externa de 32 Mbytes, con un bus de datos de 16 o 32 bits.

La placa de desarrollo añade una pantalla QVGA TFT de 3,2 pulgadas con interfaz táctil, un conector para Ethernet (RJ45), un conector de expansión para LCDs externos, un conector e interfaz CAN, un conector e interfaz MMC/SD, un conector ETM, un conector e interfaz USB OTG, un conector e interfaz USB Host, una interfaz IrDA y un altavoz.

Por lo tanto, con esta placa de desarrollo tenemos la posibilidad de conectarnos por red para llevar a cabo la comunicación con un PC, de conectar dispositivos vía USB para obtener las credenciales de los usuarios y de interactuar con los usuarios a través de la pantalla táctil.

6.1.2. Lector de Tarjetas Inteligentes: SDI010

Fig. 6.2: *Lector de tarjetas SDI010 (Fuente: [17])*

SDI010 es un lector de tarjetas inteligentes de SCM Microsystems con interfaz dual, es decir, soporta tanto tarjetas inteligentes con contacto como sin contacto.

Además de otras características importantes, en la figura 6.3 se puede ver que este lector tiene posibilidad de comunicarse vía USB, con una velocidad de 12Mbps. Por esto, y por la interfaz dual, se eligió este lector para la obtención de las credenciales de las tarjetas inteligentes.

Host Interface	▪ Full speed USB (12 Mbps) ▪ High Bus powered device ▪ CCID compliant
Smart Card Interface	▪ T=0, T=1 protocol support ▪ Communication speed up to 344,105 bps (PPS, FI parameter) ▪ Frequency up to 8 MHz (PPS, DI parameter) ▪ Support ISO 7816 Class A and AB smart card
Smart Card Connector	▪ 8 contacts (ISO position) ▪ 100,000 insertions ▪ Sliding contact
Contactless	▪ ISO 14443 A and B (13.56 MHz) ▪ Support ISO 14443 Part 1 to 4 ▪ Communication speed: up to 848 Kbit/s
Power	▪ through USB Bus
Operating Temperature	▪ 0° to +50° Celsius
OS	▪ Windows® 2000, XP and Server 2003, Vista
API	▪ PC/SC
Approvals	▪ USB ▪ Microsoft® WHQL 2000, XP, Server 2003
Environmental	▪ RoHS ▪ WEEE

Fig. 6.3: *Datos técnicos del lector SDI010 (Fuente: [17])*

Dado que ha sido imposible obtener documentación sobre el protocolo a bajo nivel de este lector, y esto es necesario para poder comunicarnos con él y con las tarjetas inteligentes desde el microcontrolador, se ha tenido que hacer un análisis de la comunicación que se explicará con detalle en el apartado 6.2.

6.1.3. Lector de Tarjetas Inteligentes: SCL010

Fig. 6.4: *Lector de tarjetas SCL010 (Fuente: [18])*

SCL010 es un lector de tarjetas inteligentes de SCM Microsystems que soporta tarjetas inteligentes sin contacto. Además, este lector tiene posibilidad de comunicarse vía USB.

Al igual que en el caso del lector de tarjetas inteligentes SDI010 ha sido imposible obtener documentación sobre el protocolo a bajo nivel de este lector. Por ello se ha tenido que hacer un

análisis de la comunicación que se explicará con detalle en el apartado 6.2. En ese apartado se explicarán, además, los motivos que han llevado a tener que utilizar este lector para comunicarnos con las tarjetas inteligentes sin contacto.

6.1.4. Pantalla Táctil

Fig. 6.5: *Pantalla táctil*

El módulo QVGA incluido con la placa MCB2470 es una pantalla TFT-LCD de 3,2 pulgadas con interfaz táctil vía SPI. Soporta formatos de hasta RGB-24.

Como se puede ver en la figura 6.6 la pantalla tiene tres tipos de interfaces, paralelo, serie o RGB. Tiene una resolución de 240 x 320 píxeles, tiene backlight de tipo LED y un voltaje de entrada de 2,8V.

En este proyecto se usará la interfaz RGB con 24 bits para mostrar imágenes en la pantalla y la interfaz SPI para el control de la interfaz táctil.

Item	Contents	Unit
LCD type	TFT TRANSMISSIVE	/
Viewing direction	9:00	O' Clock
Glass area (W × H)	57.54 × 79.2	mm^2
Viewing area (W×H)	50.6 × 66.8	mm^2
Active area (W×H)	48.6 × 64.8	mm^2
Number of Dots	240 (RGB) × 320	/
Pixel size(W×H)	0.2025 × 0.2025	mm^2
Driver IC	SSD1289	/
Backlight Type	LED	/
Module Power consumption	360	mw
Interface Type	System parallel interface /Serial Bus System Interface / RGB interface	/
Input voltage	2.8	V

Fig. 6.6: *Características de la pantalla táctil.*

6.1.5. Sensor de Huella

UPEK tiene diferentes tipos de sensores de huella, cuatro de ellos se pueden ver en la figura 6.7. Estos sensores tienen un conector de 20 pines flexible. Cada píxel de la imagen que capturan es de 8 bits. Las diferencias entre ellos son las siguientes:

- TCS1x:
  - Captura la huella en un array de píxeles de 256 x 360.
  - El área del sensor es de 18.0mm x 12.8mm.
- TCS2x:
  - Captura la huella en un array de píxeles de 208 x 288.
  - El área del sensor es de 14.4mm x 10.4mm.
- TCSxC:
  - Tiene recubrimiento de polímero resistente.
- TCSxS:
  - Tiene recubrimiento de polímero más resistente que el TCSxC.

Fig. 6.7: *Sensores de Huella de UPEK*

En la figura 6.8 se puede observar el chip TCD50D de UPEK, tiene arquitectura RISC de 32 bits. Este chip transforma la interfaz en paralelo de los sensores de huella a SPI, UART o USB.

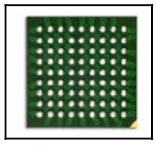


Fig. 6.8: *Chip TCD50D de UPEK*

A partir de los sensores de huella y del chip ya comentados, existen módulos que unen estos dos dispositivos para poder utilizar directamente las interfaces USB, SPI o UART. En la figura

6.9 se pueden observar los módulos que soportan UART y SPI. En nuestro sistema integraremos el módulo TCESC1.

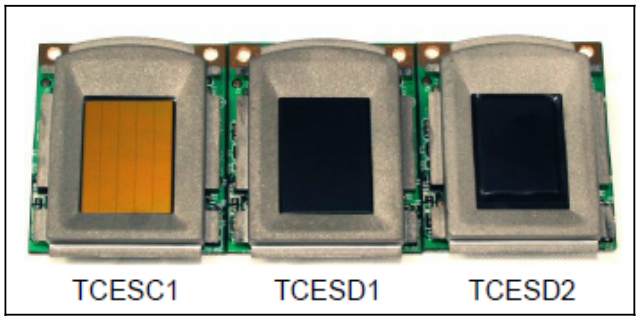


Fig. 6.9: *Módulo SPI de UPEK (TCEFC1)*

En la figura 6.10 se muestra la placa de desarrollo para los sensores de huella. Ésta placa se alimenta con un cable USB. Conectaremos el sensor con una cable flexible de 12 pines al conector 'SERIAL'. Así, tendremos acceso a las señales necesarias para establecer la comunicación vía SPI a través de los pines 'TESTPOINTS'.

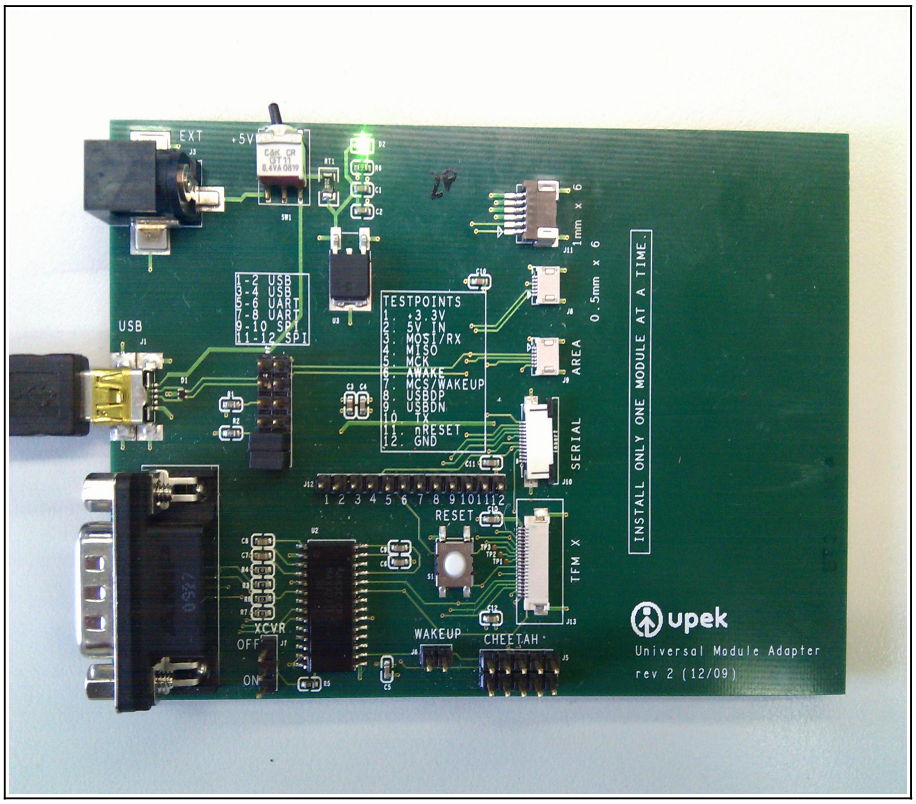


Fig. 6.10: *Placa de desarrollo de UPEK*

Las señales necesarias para la comunicación SPI son MOSI, MISO, MCK, AWAKE y MCS. Conectaremos estos pines a nuestra placa MCB2470 y así seremos capaces de comunicarnos con el sensor de huella TCESC1.

6.2. Análisis del protocolo de los dispositivos

Para poder comunicarnos con los lectores de tarjetas inteligentes se ha tenido que analizar la comunicación de estos con un ordenador. Así, se pueden enviar todas las tramas necesarias para obtener las credenciales de los usuarios desde el microcontrolador. Se ha usado el analizador de protocolo USB de LeCroy que se muestra en la Fig.6.11.

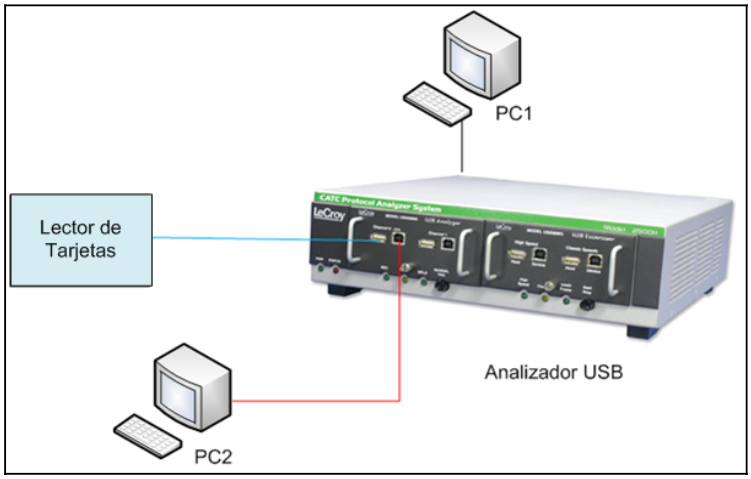


Fig. 6.11: *Esquema de conexionado.*

El procedimiento es el siguiente:

- Tendremos un PC (PC1) en el que se instalará el software del analizador para monitorizar la comunicación. Estará conectado al analizador vía USB.
- Tendremos otro PC (PC2) dónde se ejecutarán las aplicaciones necesarias para comunicarse con los lectores. Éste estará conectado al analizador vía USB.
- El lector que vayamos a analizar irá conectado al analizador vía USB en vez de directamente al PC (PC2). De esta forma tendremos el analizador espiando la comunicación.
- Así conseguimos monitorizar en PC1 la comunicación entre los lectores y PC2.

6.2.1. Introducción

En el siguiente capítulo vamos a mostrar imágenes con las tramas USB que captura el analizador. Para comprender las imágenes mejor haremos una introducción a partir de la figura 6.12.

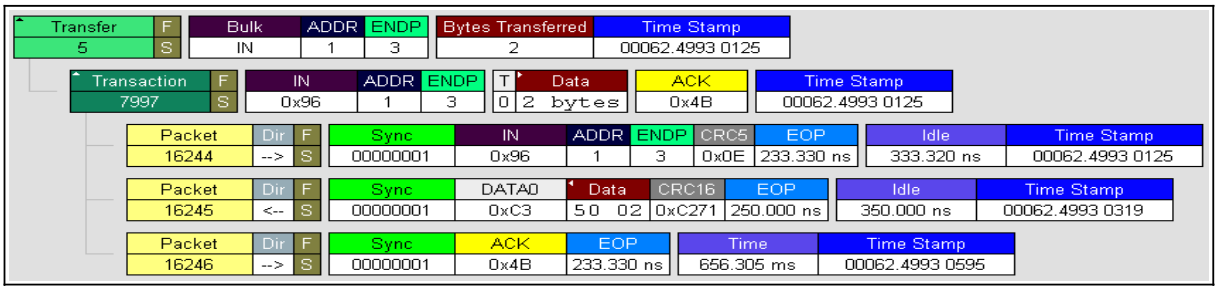


Fig. 6.12: *Intercambio de tramas USB para avisar de que no hay tarjeta insertada en el lector de tarjetas.*

En la primera línea de la figura se observa que la transferencia es de tipo Bulk, que es IN (el PC espera recibir datos) y que se transfieren 2 bytes. Además de esto vemos que el dispositivo tiene asignada la dirección 1.

En la tercera línea de la figura, en el campo Dir, observamos como la petición de los datos la hace el PC (-->). En la cuarta línea, como es una transacción de tipo DATA0 y el campo Dir es <--, vemos que el lector contesta con datos, los cuales se pueden ver en el campo Data. Finalmente, en la quinta línea observamos como el PC responde con un ACK al lector.

6.2.2. Análisis del Lector de Tarjetas Inteligentes SDI010

En este apartado se describe el análisis llevado a cabo del protocolo del lector de tarjetas SDI010 y de las tarjetas inteligentes con contacto y sin contacto.

Para poder iniciar una comunicación con el lector de tarjetas inteligentes, primero tendremos que estudiar las tramas de control intercambiadas entre el Host y el dispositivo. Con ellas se establecerá la configuración necesaria para establecer la comunicación. En la figura 6.13 vemos este intercambio de tramas:

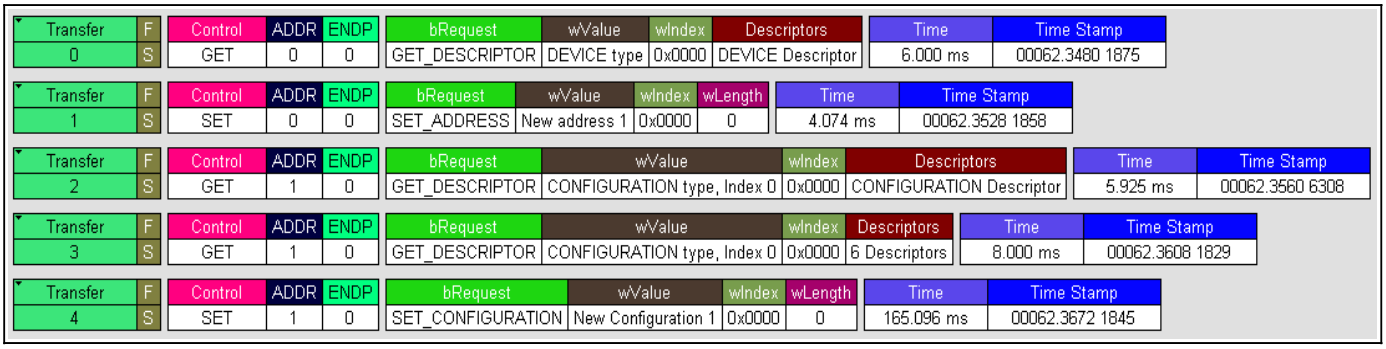


Fig. 6.13: *Intercambio de tramas USB para configurar el dispositivo.*

En la primera línea observamos que sobre la dirección 0 (utilizada hasta que el host le asigna una dirección libre al dispositivo) y el endpoint 0 (utilizado para las transferencias de control) el lector informa de que tipo de dispositivo es.

En la segunda línea el host asigna al dispositivo la dirección con la que le identificará a partir de ahora cuando quiera comunicarse con él.

En la tercera y cuarta línea observamos que, utilizando ya la dirección asignada (1), el dispositivo informa de la configuración que necesita para poder establecer comunicación con él y con las tarjetas inteligentes.

Por último, en la quinta línea, el host establece la configuración. A partir de aquí podemos intercambiar cualquier tipo de transferencia con el lector.

Una vez tenemos configurado el dispositivo, para poder obtener datos de las tarjetas inteligentes con contacto, tendremos que averiguar las tramas que manda el lector cuando hay insertada una tarjeta, y cuando no. Estudiando los intercambios de tramas llegamos a las siguientes conclusiones:

- Cuando no hay ninguna tarjeta con contacto en el lector, éste manda una trama USB de tipo Bulk con los datos: 0x5002, tal y como se muestra en la figura 6.14.

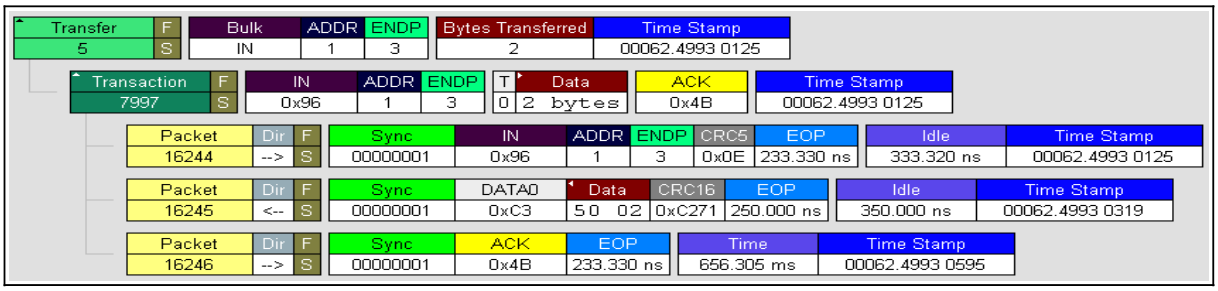


Fig. 6.14: *Intercambio de tramas USB para avisar de que no hay tarjeta insertada en el lector de tarjetas.*

- Cuando hay una tarjeta con contacto en el lector, éste manda una trama USB de tipo Bulk con los datos: 0x5003, tal y como se muestra en la figura 6.15.

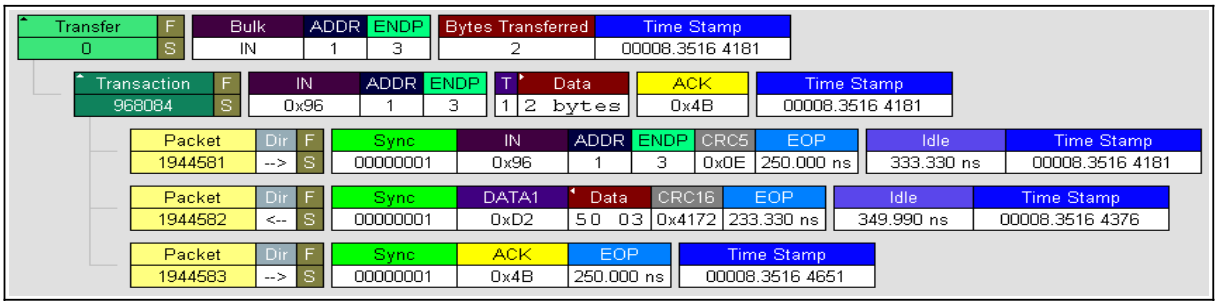


Fig. 6.15: *Intercambio de tramas USB para avisar de que si hay tarjeta insertada en el lector de tarjetas.*

Así, ya podríamos comunicarnos con las tarjetas inteligentes con contacto.

Antes de iniciar cualquier comunicación con una tarjeta inteligente, el host tiene que mandar el comando RESET a ésta, al que la tarjeta responde con el ATR (contiene información sobre el tipo de comunicación tarjeta-lectora, estructura de los datos intercambiados, protocolo de transmisión...). En la figura 6.16 observamos este intercambio de tramas cuando espiamos la comunicación del host con el lector:

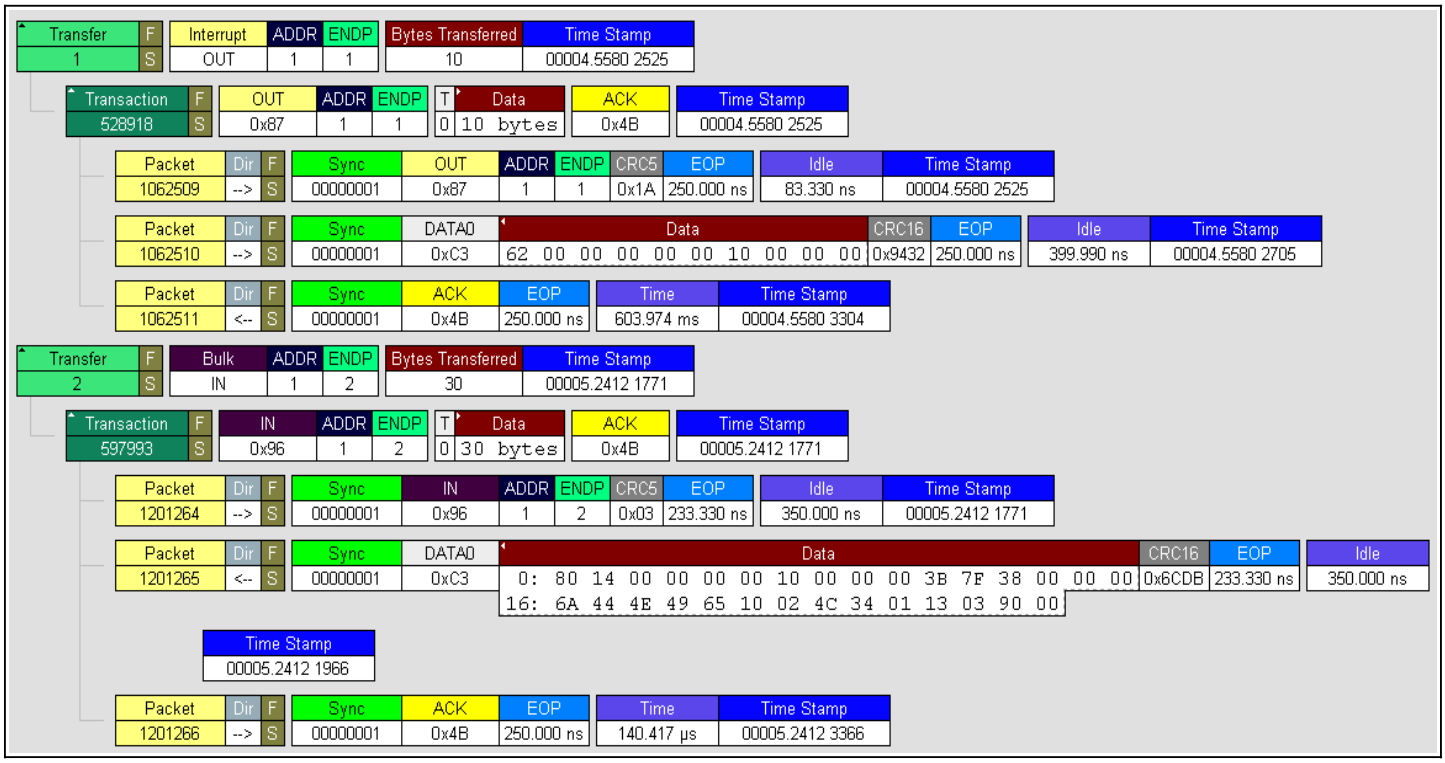


Fig. 6.16: *Intercambio de tramas USB para envío de comando RESET.*

Ahora, ya podemos establecer una comunicación con la tarjeta inteligente con contacto.

En este proyecto se ha elegido como credencial de las tarjetas inteligentes el número de serie. Después de estudiar las tramas intercambiadas para la obtención de éste, obtenemos la información que se muestra en la figura 6.17.

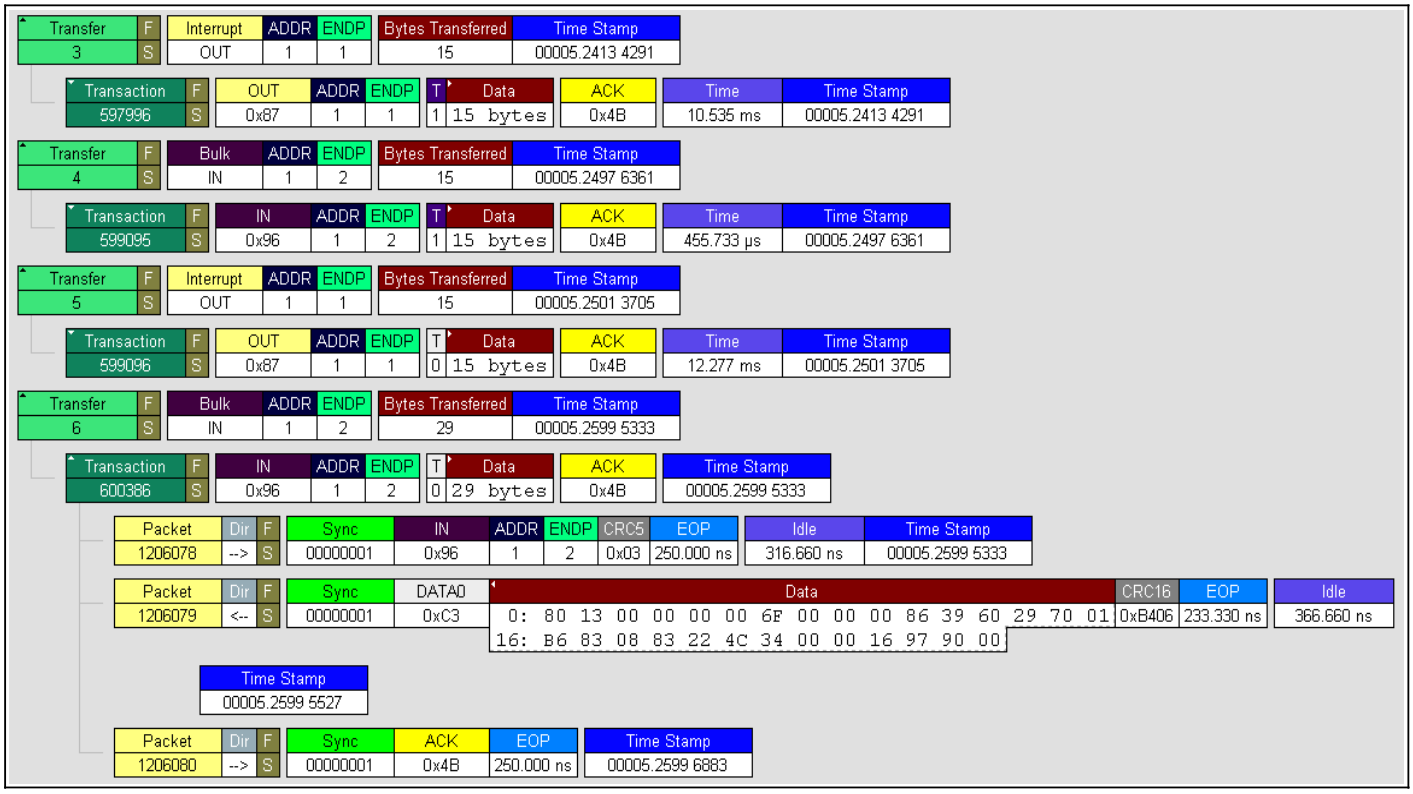


Fig. 6.17: *Intercambio de tramas USB para petición del número de serie de la tarjeta.*

El host manda al lector un comando, y el dispositivo responde con el número de serie como se observa en el campo Data de la última transferencia de la figura 6.16.

Así, ya tenemos toda la información necesaria para desarrollar el código del microcontrolador que se comunica con el lector de tarjetas y las tarjetas inteligentes con contacto.

Aun faltaría obtener el protocolo a bajo nivel de la comunicación del host con tarjetas inteligentes sin contacto. Después de un estudio riguroso de ésta comunicación, ha sido imposible obtener información clara del protocolo con el analizador. Por ello, se ha tenido que estudiar el protocolo de comunicación con otro lector de tarjetas inteligentes. Se ha elegido un lector de la misma familia que el utilizado para las tarjetas con contacto por comodidad. En el siguiente apartado se describe el protocolo.

6.2.3. Análisis del Lector de Tarjetas Inteligentes SCL010

Al estudiar el protocolo de comunicación del lector de tarjetas inteligentes con el host, se han obtenido los mismos intercambios de tramas para todos los procesos estudiados con el otro lector, excepto que en la configuración del dispositivo éste manda su información.

Así, ya tenemos, también, toda la información necesaria para desarrollar el código del microcontrolador que se comunica con las tarjetas inteligentes sin contacto.

7. Desarrollo del Sistema

Una vez descritos los distintos dispositivos que se van a utilizar en el sistema y sus protocolos de comunicación, en este capítulo describiremos la implementación de los distintos módulos del microcontrolador, como son el módulo USB, el módulo Ethernet y el módulo de la pantalla táctil. Por último, serán integrados en el sistema principal para conseguir las funcionalidades requeridas en este proyecto.

7.1. Lector de Tarjetas Inteligentes vía USB

En este apartado explicaremos en detalle el módulo USB del microcontrolador. Comenzaremos describiendo las librerías con las que se ha contado para el desarrollo de éste módulo. Seguiremos describiendo los métodos que se han desarrollado para comunicarnos con los lectores de tarjetas y las tarjetas inteligentes, tanto con contacto como sin contacto. Finalmente, se describirán los métodos principales mediante diagramas de flujo.

7.1.1. Introducción a la conexión USB

Para el desarrollo del módulo USB se estudiaron dos librerías, una de ellas facilitada por el propio fabricante de la placa de desarrollo, y la otra facilitada por NXP. Estas librerías disponen de métodos para configurar la interfaz USB y para enviar y recibir tramas.

Con la librería del fabricante, después de arreglar todos los errores de compilación que tenía, fue imposible poner en funcionamiento la interfaz USB. Por ello se probó la librería de NXP, con la cual se consiguió configurar la interfaz USB.

A continuación se describirán las variables y los métodos de la librería de NXP de los que partiremos para desarrollar el módulo USB. Además, se comentaran los cambios hechos para conseguir la funcionalidad necesaria de este sistema.

Para definir endpoints se facilita la siguiente estructura:

- ***struct*** *hcEd {* ***volatile unsigned int*** *Control;* ***volatile unsigned int*** *TailTd;* ***volatile unsigned***

int *HeadTd;* ***volatile unsigned int*** *Next } HCED*: define la estructura de los endpoint.

En esta librería sólo estaban implementados los endpoints de tipo Bulk (*EDBulkIn* y *EDBulkOut*), que se describen a continuación:

- ***HCED*** * *EDBulkIn:* endpoint para transferencias de entrada de tipo Bulk.
- ***HCED*** * *EDBulkOut:* endpoint para transferencias de salida de tipo Bulk.

Como serán necesarios endpoints de tipo Interrupt para comunicarnos con los lectores de tarjetas, se ha implementado este tipo de endpoint:

- ***HCED*** * *EDInterr*: endpoint para transferencias de entrada de tipo Interrupt.

Para almacenar las tramas intercambiadas entre el microcontrolador y los lectores de tarjetas se facilitan los siguientes buffers:

- ***unsigned char*** * *TDBuffer:* Buffer donde se almacenan las tramas intercambiadas en las transferencias de control.
- ***unsigned char*** * *UserBuffer:* Buffer donde se almacenan las tramas intercambiadas en las transferencias de datos.

El método con el que se consiguió configurar la interfaz USB proporcionado por NXP es el siguiente:

- ***void*** *Host_Init (****void****):* Configura la interfaz USB para poder iniciar una comunicación en modo Host.

El siguiente método implementado por NXP sólo configura la comunicación con memorias USB y sólo inicializa el dispositivo para tranferencias de tipo Bulk, por tanto se tuvo que modificar para configurar todo tipo de dispositivos y para aceptar todo tipo de transferencias.

- ***signed int*** *Host_EnumDev (****void****)*: se encarga de la comunicación de control inicial con los dispositivos que se encuentren conectados a la interfaz USB, asignándoles una dirección y configurándoles para poder iniciar una comunicación con ellos.

En el siguiente método, que se encarga de realizar transferencias de todo tipo, sólo estaban implementadas las transferencias Bulk, luego se tuvo que añadir el código necesario para realizar transferencias de tipo Interrupt, necesarias para la comunicación con los lectores de tarjetas.

- ***signed int*** *Host_ProcessTD (****volatile*** *HCED *ed,* ***volatile unsigned int*** *token,* ***volatile unsigned char*** ** buffer,* ***unsigned int*** *buffer_len):* realiza una transferencia con el endpoint *ed* de tipo *token* usando el buffer *buffer* de longitud *buffer_len*.

Y por último, los siguientes métodos realizan las trasnferencias de control:

- ***signed int*** *Host_CtrlRecv (****unsigned char*** *bm_request_type,* ***unsigned char*** *b_request,* ***unsigned short*** *w_value,* ***unsigned short*** *w_index,* ***unsigned short*** *w_length,* ***volatile unsigned char*** ** buffer):* realiza la transacción completa necesaria para recibir una trama de control de tipo *b_request*.
- ***signed int*** *Host_ CtrlSend (****unsigned char*** *bm_request_type,* ***unsigned char*** *b_request,* ***unsigned short*** *w_value,* ***unsigned short*** *w_index,* ***unsigned short*** *w_length,* ***volatile unsigned char*** ** buffer):* realiza la transacción completa necesaria para enviar una trama de control de tipo *b_request*.

7.1.2. Comunicación con el Lector de Tarjetas Inteligentes

A partir de los métodos descritos en el apartado anterior se han implementado las variables y los métodos necesarios para comunicarnos con el lector de tarjetas y con las tarjetas inteligentes.

Como se ha comentado en el capítulo de introducción, la credencial que se va a obtener de las tarjetas inteligentes es el número de serie de ésta. Para ello tendremos una variable global que almacenará este número:

- ***unsigned char*** *numSerie [17]:* almacenará el número de serie de la tarjeta inteligente una vez hecha la petición.

Para saber en todo momento si hay una tarjeta inteligente insertada (con contacto) o posada (sin contacto), se define otra variable global:

- ***unsigned char*** *introTarjeta:* Su valor será 0 cuando no haya ninguna tarjeta en el lector y 1 cuando si haya.

A continuación se listan los métodos desarrollados para la comunicación con el lector de tarjetas y las tarjetas inteligentes. Estos métodos implementan los intercambios de tramas estudiados en el capítulo Diseño del Sistema en el apartado de Análisis del Protocolo de Dispositivos:

- ***unsigned char*** *comprobarTarjeta (**void**):* Si se ha recibido una trama USB de tipo Interrupt del lector de tarjetas y los datos de ésta son 0x5003, se actualizará el valor de introTarjeta a 1; en cambio, si los datos son 0x5002, se actualizará a 0. Devuelve un 1 si hay tarjeta y un 0 si no hay tarjeta.
- ***unsigned char*** *getATR (**void**):* Antes de iniciar cualquier comunicación con la tarjeta se tiene que mandar el comando RESET a ésta, al que la tarjeta responde con el ATR (contiene información sobre el tipo de comunicación tarjeta-lectora, estructura de los datos intercambiados, protocolo de transmisión...). Si el intercambio de tramas se realiza correctamente se devuelve un 1, si no, se devuelve un 0. En el apartado 4.2.4. se explica con más detalle mediante un diagrama de flujo.
- ***unsigned char*** *getNumSerie (**void**):* Se hará una petición a la tarjeta del número de serie y su valor se almacenará en numSerie. En el apartado 4.2.4. se explica con más detalle mediante un diagrama de flujo.

7.1.3. Diagramas de Flujo de las Principales Funciones

En este apartado se explican, mediante diagramas de flujo, los principales métodos desarrollados para el módulo USB.

En la figura 7.1. podemos observar el método getATR(), y en la figura 7.2. el método getNumSerie().

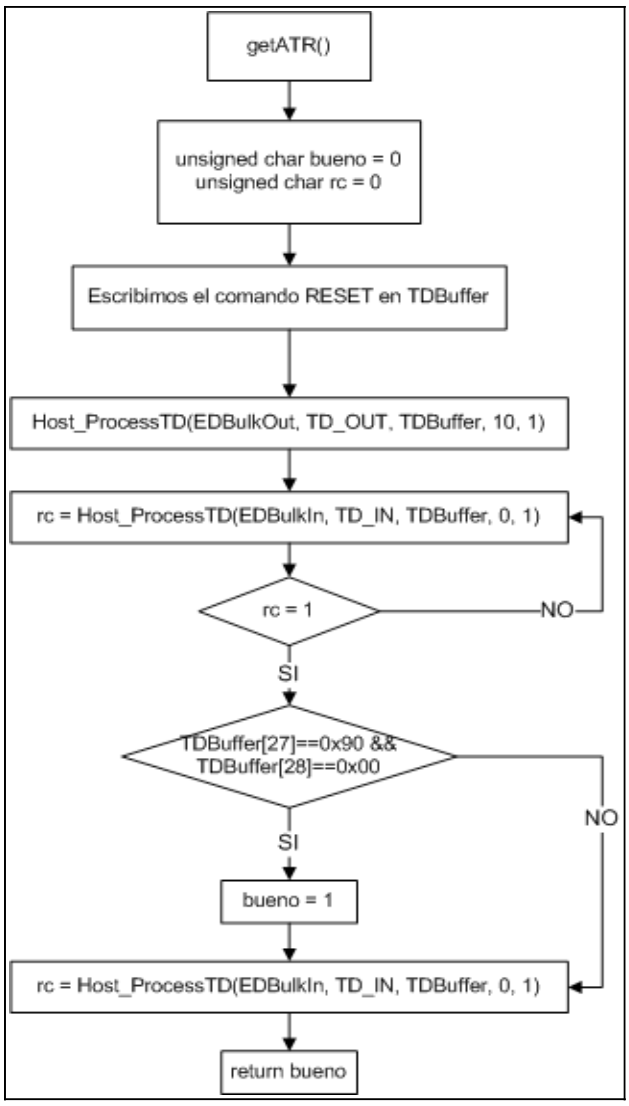


Fig. 7.1: *Diagrama de flujo del Método getATR()*

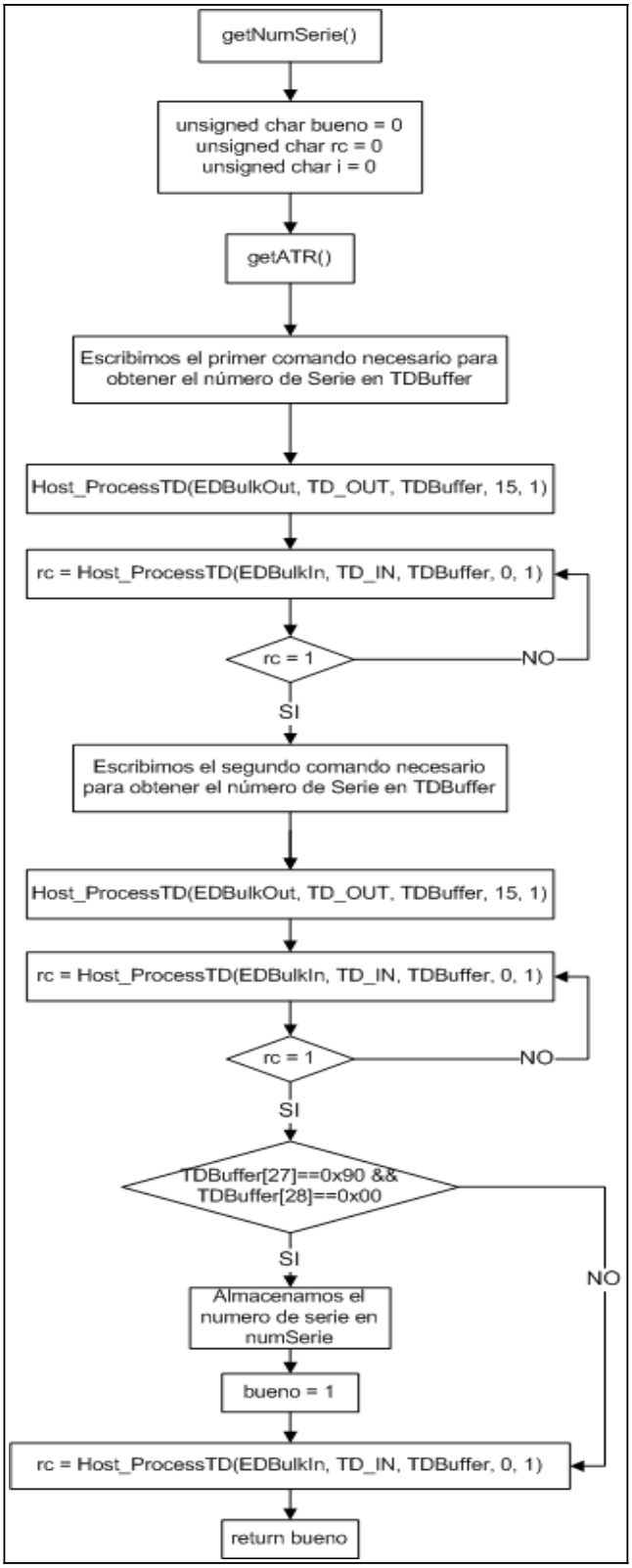


Fig. 7.2: *Diagrama de flujo del Método getNumSerie()*

7.2. Pantalla Táctil vía SPI

En este apartado explicaremos en detalle el módulo de la pantalla táctil del microcontrolador. En primer lugar describiremos las librerías con las que se ha contado para el desarrollo de éste módulo. En segundo lugar mostraremos las imágenes utilizadas en nuestro sistema junto con los métodos desarrollados para ello. Y, finalmente, se describirán las variables implementadas para interactuar con el usuario.

7.2.1. Introducción a la conexión SPI

Para el desarrollo del módulo de la pantalla táctil se estudiaron dos librerías, una de ellas facilitada por el propio fabricante de la placa de desarrollo, y la otra facilitada por NXP. Estas librerías disponen de métodos para configurar la interfaz SPI y métodos para la interfaz gráfica.

Con la librería de NXP no se consiguió poner en funcionamiento la interfaz SPI ni mostrar imágenes en la pantalla. Por ello se probó la librería de Embedded Artists, con la cual se consiguió configurar la interfaz SPI y mostrar imágenes en la pantalla, después de corregir todos los errores de compilación que tenía su librería.

A continuación se describirán las variables y los métodos de la librería de Embedded Artists de los que partiremos para desarrollar el módulo de la pantalla táctil.

Empezaremos describiendo los métodos facilitados para configurar la interfaz SPI y para comunicarse con ella. Estos métodos serán utilizados también para establecer comunicación con el sensor de huella, ya que utiliza la interfaz SPI. La comunicación con el sensor de huella se explicará con más detalle en el apartado 7.3.

- ***static void*** *spiInit (**void**):* Configura la interfaz SPI con los parámetros necesarios para la comunicación con la capa táctil de la pantalla, es decir, para que tome los datos de 16 bits en 16 bits, para que los datos se tomen en el segundo flanco del reloj, y para que el reloj sea activo a nivel bajo (modo 3).
- ***void*** *touch_cs_low (**unsigned char** tipo):* Si tipo es 0 pone a nivel bajo la señal CS de la pantalla táctil.
- ***void*** *touch_cs_high (**unsigned char** tipo):* Si tipo es 0 pone a nivel bajo la señal CS de la pantalla táctil.
- ***void*** *tp_xfer (**unsigned char** *out, **unsigned char** *in, **int** bytes, **unsigned char** tipo:* envía por la interfaz SPI los datos contenidos en *out* y almacena los datos recibidos en *in*, ambos de

longitud *bytes*. La variable *tipo* especifica si el método es llamado por la pantalla táctil o por el sensor de huella. Para la pantalla esta variable será 0, así, SPI funcionará en modo 4.

A continuación se describen los métodos utilizados para obtener las coordenadas del punto que se ha pulsado de la pantalla:

- ***static unsigned short*** *spi_read_tp (**unsigned char** command)*: Pone a nivel bajo la señal CS de la pantalla mediante el método *touch_cs_low(0),* es decir, hace que ésta escuche por la interfaz SPI. Lee los datos que envía la pantalla cuando se le manda un comando mediante el método *tp_xfer* con el parámetro tipo de éste a 0. Pone a nivel alto la señal CS de la pantalla mediante el método *touch_cs_high(0),* es decir, hace que ésta no escuche por la interfaz SPI.
- ***void*** *touch_xyz (**signed int** * x, **signed int** * y, **signed int** * z):* Obtiene las coordenadas (x,y,z) del punto que se ha pulsado en la pantalla táctil mediante el método *spi_read_tp*. Si z es nulo significa que no se ha pulsado la pantalla.

Se facilitan las siguientes dos constantes para tener acceso a ellas en cualquier momento:

- ancho: su valor será 240.
- alto: su valor será 320.

Para poder mostrar una imagen de tamaño 320x240 píxeles o menor Embedded Artists ha implementado el siguiente método, el cual sólo admite imágenes de hasta RGB-16.

- ***void*** *lcd_picture (**unsigned short** x, **unsigned short** y, **unsigned short** width, **unsigned short** height, **unsigned short** * pPicture):* muestra la imagen *pPicture* desde las coordenadas (x, y) con ancho *width* y alto *height*.

A parte de este método con el que se muestran por pantalla imágenes fijas, se facilita una librería de funciones más simples que se listan a continuación:

- ***void*** *lcd_point (**unsigned short** x, **unsigned short** y, **unsigned short** color):* muestra el píxel de la coordenada (x, y) del color *color.*
- ***unsigned char*** *lcd_putChar (**unsigned short** x, **unsigned short** y, **unsigned char** ch, **unsigned short** c)*: muestra el carácter *ch* del color *c* en la coordenada (x, y).

- ***void*** *lcd_putString (**unsigned short** x, **unsigned short** y, **unsigned char** * pStr, **unsigned short** color):* muestra la cadena de caracteres *pStr* de color *color* desde la coordenada (x, y).
- ***static void*** *hLine (**unsigned short** x0, **unsigned short** y0, **unsigned short** x1, **unsigned short** color):* muestra una línea horizontal de color *color* en la coordenada y0 desde x0 a x1.
- ***static void*** *vLine (**unsigned short** x0, **unsigned short** y0, **unsigned short** y1, **unsigned short** color):* muestra una línea vertical del color color en la coordenada x0 desde y0 a y1.
- ***void*** *lcd_drawRect (**unsigned short** x0, **unsigned short** y0, **unsigned short** x1, **unsigned short** y1, **unsigned short** color):* muestra un rectángulo sin relleno de color color de la coordenada (x0, y0) a la (x1, y1).
- ***void*** *lcd_fillRect (**unsigned short** x0, **unsigned short** y0, **unsigned short** x1, **unsigned short** y1, **unsigned short** color):* muestra un rectángulo con relleno de color color de la coordenada (x0, y0) a la (x1, y1).
- ***void*** *lcd_line (**unsigned short** x0, **unsigned short** y0, **unsigned short** x1, **unsigned short** y1, **unsigned short** color):* muestra una línea de color *color* de la coordenada (x0, y0) a la (x1, y1).

7.2.2. Pantallas definidas para el Sistema Desarrollado

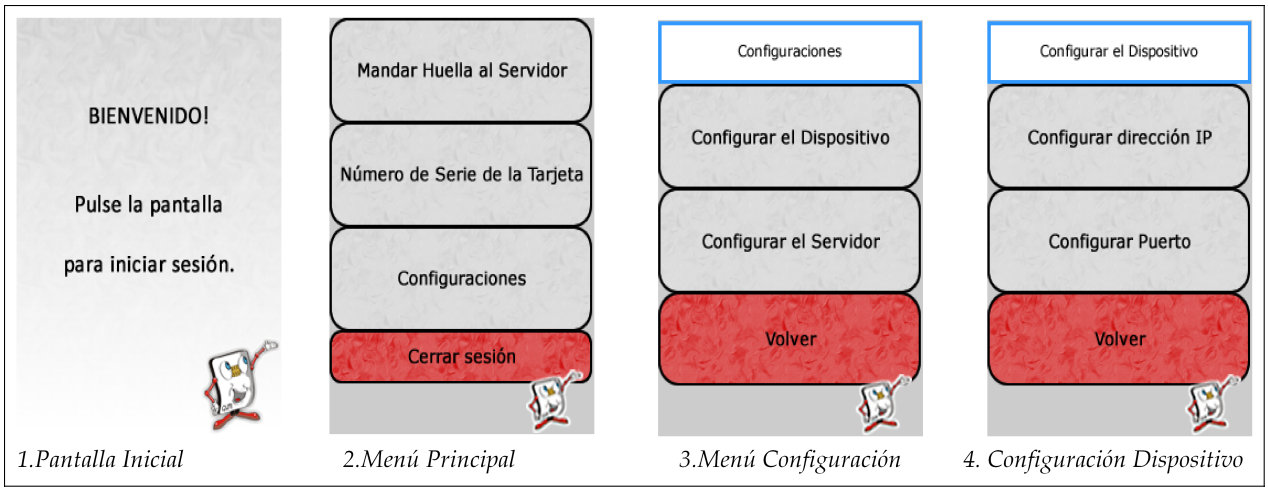
Como se comentó en el capítulo de introducción, necesitaremos interactuar con el usuario para obtener las credenciales de las tarjetas inteligentes y para configurar las direcciones IP y puertos que usarán para la comunicación el microcontrolador y el PC. Para ello se han creado unas imágenes fijas con las diferentes opciones posibles.

Como el mejor formato que admite la pantalla es RGB-16 según la documentación facilitada por Embedded Artists, el siguiente paso era diseñar la interfaz gráfica con algún programa de diseño gráfico que genere las imágenes en este formato, para luego mostrarlas en la pantalla con el método *lcd_picture*. Al no encontrar ningún programa que haga esto, se desarrolló una aplicación en Visual C# que transforma imágenes en RGB-24 a RGB-16. Esto se hace con una simple regla de tres que pasa los 8 bits del rojo y el azul a 5 bits, y los 8 bits del verde a 6 bits. Al mostrar en la pantalla las imágenes conseguidas algunos píxeles no muestran el color real, ya que al truncar los valores de los colores se pierde precisión. Por ello se decidió indagar más a fondo en las características de la pantalla.

Finalmente se consiguieron otros documentos de la pantalla facilitados por el fabricante de la misma. Estudiándolos se descubrió que ésta admite también RGB-24, por lo que se tuvo que modificar el método de Embedded Artists para mostrar imágenes en este formato:

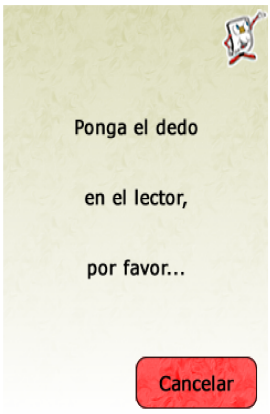
- El método *lcd_picture* tiene el parámetro ***unsigned short*** * *pPicture* que es un array que contiene un píxel de la imagen en cada posición, siendo el tipo de datos short (16 bits) ya que usa RGB-16. Para utilizar formato RGB-24 cada posición del array tendrá que tener al menos 24 bits para cada píxel, luego cambiaremos ese parámetro a ***unsigned int*** * *pPicture.* Además, en el código del método, se tendrán que hacer las modificaciones oportunas. El método inicialmente copiaba el array de la imagen al buffer de la pantalla de 16 bits en 16 bits. Para RGB-24 tendremos que copiar la imagen al buffer de 32 bits en 32 bits. Como cada píxel de las imágenes es de 24 bits, tendremos que copiar la imagen en el buffer añadiendo el byte 0x00 al principio de cada píxel para completar los 32 bits.

La pantalla táctil se ha configurado finalmente para trabajar en RGB-24. Las imágenes que se van a mostrar en nuestro sistema se han diseñado en formato BMP-24bits. Para que nuestro sistema las pueda interpretar, tendremos que almacenar los bytes de la imagen en un array de enteros quitando la cabecera BMP. A continuación se muestran las imágenes diseñadas:

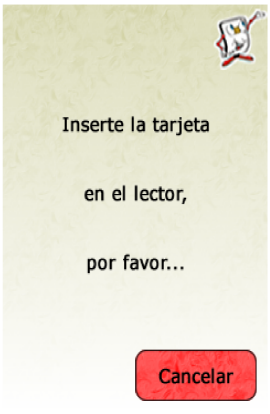


1.Pantalla Inicial *2.Menú Principal* *3.Menú Configuración* *4. Configuración Dispositivo*

5. Configuración Servidor

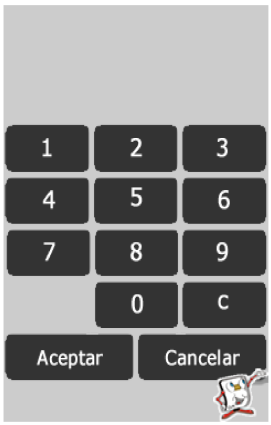


6. Introducir Huella

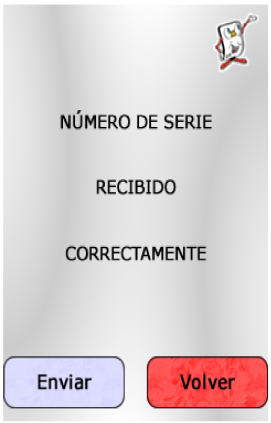


7. Insertar Tarjeta

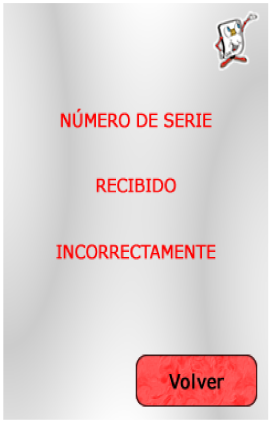
8. Teclado 1



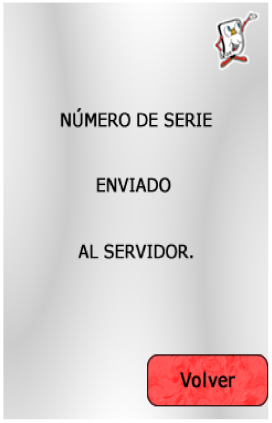
9. Teclado 2



10. Número Serie Correcto

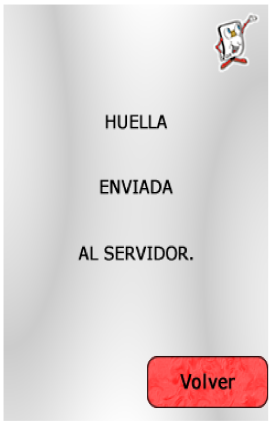


11. Número Serie Incorrecto



12. Número Serie Enviado

13. Número Serie No Enviado



14. Huella Enviada

15. Huella No Enviada

A continuación se listan los métodos utilizados para mostrar las imágenes anteriores en la pantalla:

- ***void*** *paintCentrado (**char** * pMsg, **unsigned short** x, **unsigned short** y, **unsigned short** color):* muestra la cadena de caracteres *pMsg* de color *color* centrada en la coordenada (x, y).
- ***void*** *mostrarInicio (**void**):* muestra la imagen número 1 mediante el método *lcd_picture(...)*.
- ***void*** *mostrarMenuCuatro (**unsigned char** tipo):* si tipo es igual a 0 mostramos la imagen número 2 mediante el método *cd_picture(...)*.
- ***void*** *mostrarMenuConf (**void**):* muestra la imagen número 3 mediante el método *lcd_picture(...)*.
- ***void*** *mostrarConfDisp (**void**):* muestra la imagen número 4 mediante el método *lcd_picture(...)*.
- ***void*** *mostrarConfServ (**void**):* muestra la imagen número 5 mediante el método l*cd_picture(...)*.
- ***void*** *mostrarTeclado (**unsigned char** tipo):* muestra la imagen número 8 si tipo es igual a 0 y la imagen 9 si es igual a 1; ambas mediante el método *lcd_picture(...)*.
- ***void*** *mostrarIP (**unsigned char** estado):* muestra la dirección IP del servidor (si estado es igual a 6) o del dispositivo (si estado es igual a 5) actual y el teclado mediante el método *mostrarTeclado(0).*
- ***void*** *mostrarPuerto (**unsigned char** estado):* muestra el puerto del servidor (si estado es igual a 8) o del dispositivo (si estado es igual a 7) actual y el teclado mediante el método *mostrarTeclado(1).*
- ***void*** *mostrarPIN (**void**):* muestra "INTRODUZCA EL PIN:" mediante el método *paintCentrado(...)* y el teclado mediante el método *mostrarTeclado(1).*
- ***void*** *mostrarEdicionIP (**char** ip [16]):* muestra "IP NUEVA: " seguido de la dirección IP pasada por parámetro mediante el método *paintCentrado(...)*.
- ***void*** *mostrarEdicionPuerto (**char** puerto [6]):* muestra "PUERTO NUEVO: " seguido del puerto pasado por parámetro mediante el método *paintCentrado(...)*.
- **void** mostrarEdicionPIN (**unsigned char** numero): muestra tantos "*" como indique *numero* mediante el método *paintCentrado(...)*.
- ***void*** *mostrarEspera (**unsigned char** tipo):* si tipo es igual a 0 mostramos la imagen número 7, si es igual a 1 mostramos la imagen 6, y si es igual a 2 mostramos la imagen. Todas

éstas imágenes mediante *lcd_picture(...)*.

- ***void** mostrarEnvio (**unsigned char tipo**):* si tipo es igual a 0 mostramos la imagen número 13, si es igual a 1 mostramos la imagen 12, si es igual a 2 mostramos la imagen 15, y si es igual a 3 mostramos la imagen 14, todas mediante el método *lcd_picture(...)*.

  ***void** mostrarNumeroSerie (**unsigned char** tipo):* si tipo es igual a 1 mostramos la imagen número 10, y si es igual a 0 mostramos la número 11; ambas mediante el método *lcd_picture(...)*.

7.2.3. Relación entre elementos de las pantallas y acciones a realizar

A partir de las imágenes del apartado anterior interactuaremos con el usuario, para ello necesitaremos tener un control de la pantalla actual que está viendo el usuario, de los botones que tiene esa pantalla y de las acciones a realizar cuando se pulse ésta. En este apartado describiremos las variables que se han definido para este control, y mostraremos una tabla resumen de todas las acciones de los botones de cada pantalla.

A continuación se listan las variables definidas para controlar las acciones requeridas:

- ***unsigned char** estadoPantalla:* informa de la imagen actual que se está mostrando al usuario. Relación del valor de la variable con la imagen:
  - 0: Pantalla Inicial
  - 1: Menú Principal
  - 2: Menú Configuración
  - 3: Menú Configuración Dispositivo
  - 4: Menú Configuración Servidor
  - 5: IP Dispositivo
  - 6: IP Servidor
  - 7: Puerto Dispositivo
  - 8: Puerto Servidor
  - 9: Introducir Huella
  - 10: Introducir Tarjeta
  - 11: Recibir del Lector Número Serie
  - 12: Número de Serie enviado bien
  - 13: Numero de Serie enviado mal

  - 14: Huella enviada bien
  - 15: Huella enviada mal
- ***char*** *ipASCII [16]:* Cuando se esté editando una nueva dirección IP, esta variable irá almacenando en formato ASCII (para mostrar por pantalla) el valor actual que esté pulsando el usuario.
- ***unsigned char*** *ip [4]:* Una vez comprobado que la dirección IP editada por el usuario es correcta, almacenaremos en esta variable el valor en decimal de la dirección IP para después actualizarlo en nuestro sistema.
- ***char*** *puertoASCII [6]:* Cuando se esté editando un nuevo puerto, esta variable irá almacenando en formato ASCII (para mostrar por pantalla) el valor actual que esté pulsando el usuario.
- ***unsigned int*** *puerto:* Una vez comprobado que el puerto editado por el usuario es correcto, almacenaremos en esta variable el valor en decimal del puerto para después actualizarlo en nuestro sistema.
- ***unsigned char*** *i:* Variable usada para tener control de cuantos caracteres de las direcciones IP o de los puertos ha editado por el momento el usuario.
- ***unsigned char*** *envio:* Cuando su valor es 1 significa que en la pantalla actual hay un botón ENVIAR.

A continuación mostramos la tabla resumen con las acciones que se realizan según se pulse la pantalla:

IMAGEN	BOTONES	ACCIÓN SI PULSADO
Pantalla Inicial		Menú principal
Menú Principal	Mandar Huella al Servidor	Introducir Huella.
	Número de Serie de la Tarjeta	Si hay tarjeta mostramos número de serie. Si no hay tarjeta Petición de Tarjeta.
	Configuraciones	Menú Configuración
	Salir	Pantalla Inicial
	Configuración del Dispositivo	Menú Configuración Dispositivo

Menú Configuración	Configuración del Servidor	Menú Configuración Servidor
	Volver	Menú Principal
Introducir Huella	Cancelar	Menú Principal
Introducir Tarjeta	Cancelar	Menú Principal
Numero Serie Correcto	Enviar	Si se envía al servidor correctamente se muestra: Numero Serie Enviado. Si no, se muestra: Numero Serie No enviado.
	Volver	Menú Principal
Numero Serie Incorrecto	Volver	Menú Principal
Número Serie Enviado	Volver	Menú Principal
Número Serie No Enviado	Volver	Menú Principal
Huella Enviada	Volver	Menú Principal
Huella No Enviada	Volver	Menú Principal
Menú Configuración Dispositivo	Configurar Dirección IP	Teclado + Edición IP
	Configurar Puerto	Teclado + Edición Puerto
	Volver	Menú Configuración
Menú Configuración Servidor	Configurar Dirección IP	Teclado + Edición IP
	Configurar Puerto	Teclado + Edición Puerto
	Volver	Menú Configuración
Teclado 1	Números	Actualizar variable de edición
	Aceptar	Comprobar si la dirección IP es correcta y mostrar un mensaje por pantalla informando.
	Volver	Vuelve al menú de configuración del dispositivo o del servidor, dependiendo de en que pantalla estábamos antes.
	Números	Actualizar variable de edición

Teclado 2	Aceptar	Comprobar si el puerto es correcto y mostrar un mensaje por pantalla informando.
	Volver	Vuelve al menú de configuración del dispositivo o del servidor, dependiendo de en que pantalla estábamos antes.

7.3. Sensor de Huella vía SPI

En este apartado explicaremos en detalle el módulo del sensor de huella del microcontrolador. Comenzaremos describiendo los métodos utilizados para la configuración de la interfaz SPI y para la comunicación con ella. Y seguiremos describiendo las librerías utilizadas para establecer conexión y poder comunicarnos con el sensor de huella.

7.3.1. Introducción a la conexión SPI

Haciendo algunas modificaciones en los métodos de inicialización de la interfaz SPI facilitados por Embedded Artists, que se describieron en el apartado de la pantalla táctil, estableceremos comunicación con el sensor de huella. A continuación se especifican:

- ***void*** *touch_cs_low (****unsigned char*** *tipo):* Si tipo es 1 pone a nivel bajo la señal CS del sensor de huella.
- ***void*** *touch_cs_high (****unsigned char*** *tipo):* Si tipo es 1 pone a nivel bajo la señal CS del sensor de huella
- ***void*** *tp_xfer (****unsigned char*** **out*, ***unsigned char*** **in*, ***int*** *bytes*, ***unsigned char*** *tipo:* envía por la interfaz SPI los datos contenidos en *out* y almacena los datos recibidos en *in*, ambos de longitud *bytes*. La variable *tipo* especifica si el método es llamado por la pantalla táctil o por el sensor de huella. Para el sensor de huella *tipo* será 1, así los datos se tomarán de 8 bits en 8 bits y la interfaz SPI funcionará en modo 0, es decir, los datos se tomarán en el primer flanco del reloj, que será activo a nivel alto.

7.3.2. Comunicación con el Sensor de Huella

A continuación se describirá la librería facilitada por el fabricante de los sensores de huella con la que se establecerá la comunicación:

- **unsigned int** PTOpen(): Establece una conexión con el sensor de huella, es decir, comprueba si el sensor de huella está activo, si no es así, le manda el comando WAKEUP para activarlo; y cuando esté activo le manda un comando RESET. Si ha conseguido realizar todas las acciones devuelve 0, si no devuelve un código de error.
- ***unsigned int*** *PTDetectFingerEx():* Comprueba si se ha posado un dedo en el sensor de huella, si es así devuelve un 0, si no devuelve un código de error.
- ***unsigned int*** *PTGrabPart (**unsigned char** * huella):* Captura la imagen de la huella y la almacena en el puntero *huella.* Hay que realizar varias interacciones, en cada llamada almacena 1.008 bytes de un total de 51.840 bytes de la imagen de la huella. Si se realiza con éxito la operación devuelve un 0, si no devuelve un código de error.
- ***unsigned int*** *PTClose():* Cierra la conexión con el sensor de huella, es decir, le manda el comando CLOSE, espera un ACK, y le desactiva mandando el comando SLEEP. Si ha conseguido realizar todas las acciones devuelve un 0, si no devuelve un código de error.

El método desarrollado para obtener la huella al completo es el siguiente:

- ***unsigned int*** *GrabImage():* mediante varias llamadas al método *PTGrabPart()* se obtiene una imagen de 51.840 bytes, es decir, de 192 x 270 píxeles en escala de grises (1 byte por píxel).

7.4. Conexión Ethernet

En este apartado explicaremos en detalle el módulo Ethernet del microcontrolador. Comenzaremos describiendo las librerías con las que se ha contado para el desarrollo de éste módulo. Seguiremos describiendo los métodos que usaremos para comunicarnos con un PC mediante el protocolo TCP/IP. Finalmente, se describirán los métodos principales mediante diagramas de flujo.

7.4.1. Introducción a la Conexión de Red

Para el desarrollo del módulo Ethernet se estudiaron dos librerías, una de ellas facilitada por el propio fabricante de la placa de desarrollo, y la otra facilitada por NXP. Estas librerías disponen de métodos para configurar la interfaz de red y métodos para enviar y recibir tramas.

Con la librería de NXP no se consiguió poner en funcionamiento la interfaz de red. Por ello se probó la librería de Embedded Artists, con la cual se consiguió configurar la interfaz de red y

enviar y recibir tramas, después de corregir todos los errores de compilación que tenía su librería.

A continuación se describirán las variables y los métodos de la librería de Embedded Artists de los que partiremos para desarrollar el módulo Ethernet.

Para almacenar las tramas recibidas y las tramas a enviar la librería define un buffer:

- ***unsigned char*** *outBuf [1514]:* almacena los datos recibidos desde el ordenador por red y los que se van a enviar al ordenador.

Para configurar la interfaz de red utilizaremos el siguiente método de la librería:

- ***void*** *ethIf_init (**unsigned char** * pEthAddr):* configura la interfaz con la MAC definida en *pEthAddr* para poder iniciar la comunicación por red.

Para calcular el checksum de IP la librería nos facilita el siguiente método:

- ***static unsigned short*** *checksum (**unsigned char** * pStartAddress, **unsigned short** len, **unsigned short** initialValue):* Se suman, con aritmética complemento a uno, los datos que contiene *pStartAddress* con longitud *len,* como palabras de 16 bits. Finalmente se calcula el complemento a uno del resultado obtenido en la operación anterior.

Para almacenar las tramas recibidas, si se ha recibido alguna, la librería nos ofrece los siguientes métodos:

- ***static unsigned short*** *getPacket (**unsigned char** * pBuf, **unsigned short** bufLen):* almacena los datos recibidos por red en la variable *pBuf. S*i la longitud de estos es mayor que la longitud del buffer (*buflen*), los trunca a *bufLen*. Devuelve la longitud final.
- ***unsigned short*** *ethIf_poll (**unsigned char** * pBuf, **unsigned short** len):* comprueba si se han recibido datos en la interfaz de red. Si es así llama al método getPacket(), devolviendo la longitud de los datos recibidos. Si no es así, devuelve el valor 0.

Para procesar las tramas recibidas y actuar en consecuencia se facilita el método *ethInput*. Éste método se tuvo que modificar ya que sólo se procesaba tramas de tipo IP y en este proyecto se tendrán que procesar también tramas de tipo ARP.

- ***void*** *ethInput (**unsigned char** * pBuf, **unsigned short** len): pBuf* contiene la cabecera Ethernet de la trama recibida por red junto con los datos. En este método se comprueba que la trama recibida es correcta a nivel Ethernet, es decir, la longitud tiene que ser mayor que 14 bytes

(tamaño de la cabecera Ethernet). Con esto, si el protocolo de la trama es IP se llama al método *ipInput()*, y si el protocolo es ARP se llama a *arpInput()*. Éste último método se explicará en detalle en el siguiente apartado.

Para procesar las tramas de tipo IP se facilita el método *ipInput*. Éste método se tuvo que modificar ya que sólo se procesaba tramas de tipo ICMP y en este proyecto se tendrán que procesar también tramas de tipo TCP.

- ***static void*** *ipInput (****unsigned char*** ** pBuf,* ***unsigned short*** *len): pBuf* contiene la cabecera IP de la trama recibida por red junto con los datos. En este método se comprueba que la trama recibida es correcta a nivel IP, es decir, la longitud tiene que ser mayor que 20 bytes (tamaño de la cabecera IP), el checksum tiene que ser correcto, la IP de origen tiene que ser la del servidor, y la IP de destino la del microcontrolador. Con todo esto, si el protocolo de la trama es TCP se llama al método *tcpInput()*. Éste método se explica en detalle en el siguiente apartado.
- ***void*** *ethIf_send (****unsigned char*** ** pData,* ***unsigned short*** *len):* envía por la interfaz de red los datos contenidos en *pData* de longitud *len*.

7.4.2. Comunicación vía Ethernet

A partir de los métodos descritos en el apartado anterior se han implementado las variables y los métodos necesarios para comunicarnos vía Ethernet.

Para poder desarrollar toda la funcionalidad del protocolo TCP/IP se han definido las siguientes variables:

- ***unsigned short*** *seqNuestro:* número de secuencia TCP del microcontrolador.
- ***unsigned char*** *estado:* Si su valor es 0 no hay conexión TCP establecida; si es 1, se ha enviado una trama de tipo SYN y se está esperando una de tipo SYN-ACK; si es 2, la conexión TCP está establecida; y si es 3, se ha recibido una petición para finalizar la conexión.
- ***unsigned char*** *cambiadaIP:* su valor es 1 cuando se ha modificado el valor de la dirección IP del servidor, con lo cual para la siguiente conexión TCP tendremos que obtener su dirección MAC.
- ***unsigned char*** *seudoCab [1514]:* para almacenar la seudo-cabecera necesaria para calcular el checksum de un paquete TCP.

- ***static unsigned short*** *portServer:* almacena el puerto TCP del servidor con el que se va a establecer la comunicación.
- ***static unsigned short*** *portMicro:* almacena el puerto TCP del microcontrolador con el que se va a establecer la comunicación.
- ***static unsigned char*** *localIP[4]:* almacena la dirección IP del microcontrolador.
- ***static unsigned char*** *serverIP[4]:* almacena la dirección IP del servidor.
- ***static unsigned char*** *mac[6]*: almacena la dirección MAC del microcontrolador.
- ***static unsigned char*** *macServidor[6]:* almacena la dirección MAC del servidor.

Para tener acceso en todo momento a las direcciones IP y puertos, tanto del microcontrolador como del PC con el que se establezca la comunicación, desarrollamos los siguientes métodos:

- ***unsigned char*** * *getIPMicro (**void**):* obtiene el valor de la dirección IP del microcontrolador.
- ***void*** *setIPMicro (**unsigned char** IPMicro [4]):* establece la dirección IP del microcontrolador a *IPMicro.*
- ***unsigned char*** * *getIPServer (**void**):* obtiene el valor de la dirección IP del servidor.
- ***void*** *setIPServer (**unsigned char** IPServer [4]):* establece la dirección IP del servidor a *IPServer.*
- ***unsigned short*** *getPuertoMicro (**void**):* obtiene el valor del puerto del microcontrolador.
- ***void*** *setPuertoMicro (unsigned **short** puertoMicro):* establece el puerto del microcontrolador a *puertoMicro.*
- ***unsigned short*** *getPuertoServer (**void**):* obtiene el valor del puerto del PC.
- ***void*** *setPuertoServer (unsigned **short** puertoServer):* establece el puerto del PC a *puertoServer.*

Para poder comunicarse por red según el protocolo TCP/IP se han desarrollado los siguientes métodos:

- ***static void*** *tcpInput (**unsigned char** * pBuf, **unsigned short** len, **unsigned int** ipDest): pBuf* contiene la cabecera TCP de la trama recibida por red junto con los datos. Se comprueba que la trama recibida es correcta a nivel TCP, es decir, la longitud tiene que ser mayor o igual que 20 bytes (tamaño de la cabecera TCP), el checksum tiene que ser correcto, y,

además, si hay una conexión establecida, los números de secuencia tienen que coincidir. Con todo esto, se encarga de analizar las tramas recibidas y actuar en consecuencia. En el apartado 4.3.3. se explica con más detalle mediante un diagrama de flujo.

- ***static void*** *arpInput (****unsigned char*** ** pBuf,* ***unsigned short*** *len): pBuf* contiene la cabecera ARP de la trama recibida por red junto con los datos. En este método se comprueba que la trama recibida es correcta a nivel ARP, es decir, la longitud tiene que ser mayor que 28 bytes (tamaño de la cabecera IP). Con esto, si es una petición ARP a la dirección IP del microcontrolador enviamos una repuesta ARP informando de nuestra dirección MAC a la IP que hizo la petición. Si es una respuesta ARP enviada desde el servidor, y le habíamos enviado con anterioridad una petición ARP, almacenamos la dirección MAC recibida en esta trama en *macServidor*.
- ***static void*** *enviarARP (****unsigned char*** ** pBuf):* se envía una petición ARP al servidor mediante el método *ethIf_send*. En la imagen 7.3 se muestra el intercambio de tramas llevado a cabo:

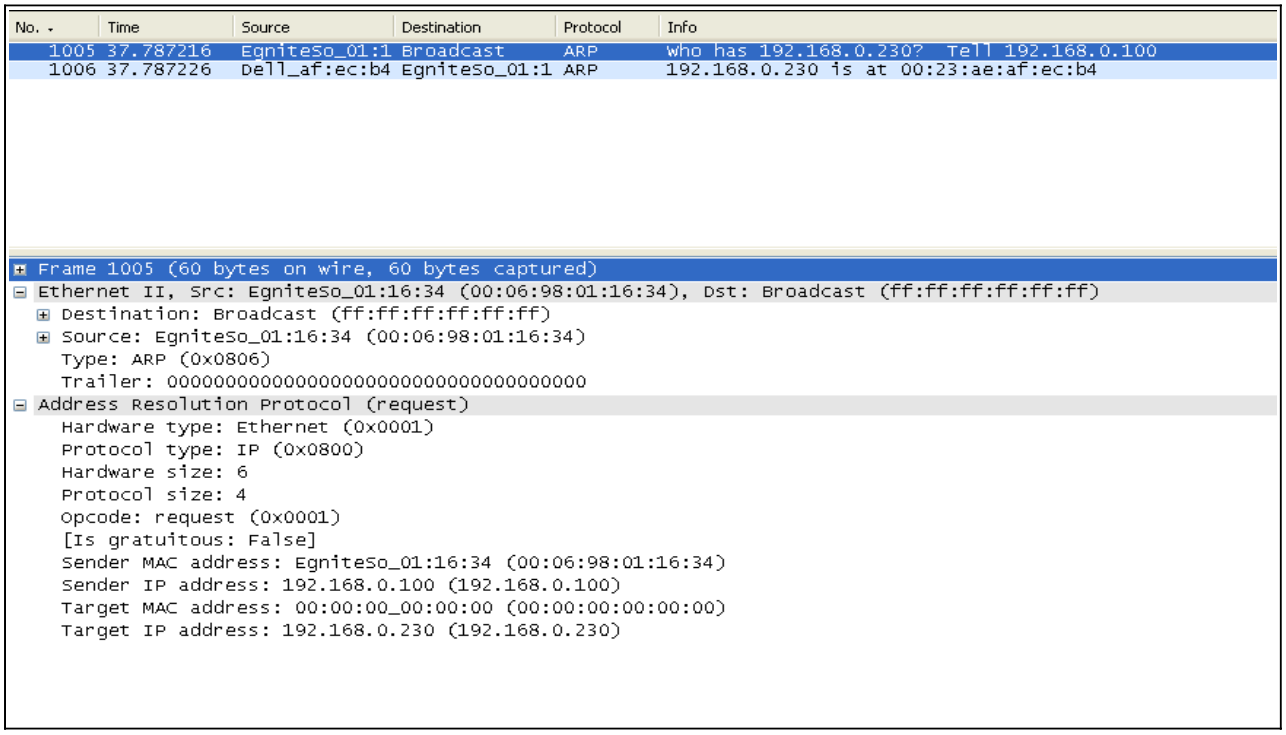


Fig. 7.3: *Intercambio de tramas TCP para petición ARP.*

- ***unsigned char*** *abrirTCP (****unsigned char*** ** pBuf):* Si no tenemos almacenada la dirección MAC del servidor (es decir, si la variable cambiadaIP es igual a 1), se envía una petición ARP mediante el método *enviarARP.* Una vez tengamos la dirección MAC del servidor, se envía

a éste una trama de tipo SYN mediante el método *ethIf_send* y se espera a recibir una trama del servidor de tipo SYN ACK. Una vez recibida, se contesta con una trama de tipo ACK para confirmar que la conexión TCP está establecida. En el caso de que no se recibiera la trama SYN ACK, se volvería a enviar la trama SYN, esperando de nuevo el SYN ACK. Habrá un máximo de cinco intentos para establecer la conexión. Si la conexión se ha establecido con éxito se devuelve un 1, si no, se devuelve un 0. En la imagen 7.4 se muestra el intercambio de tramas llevado a cabo:

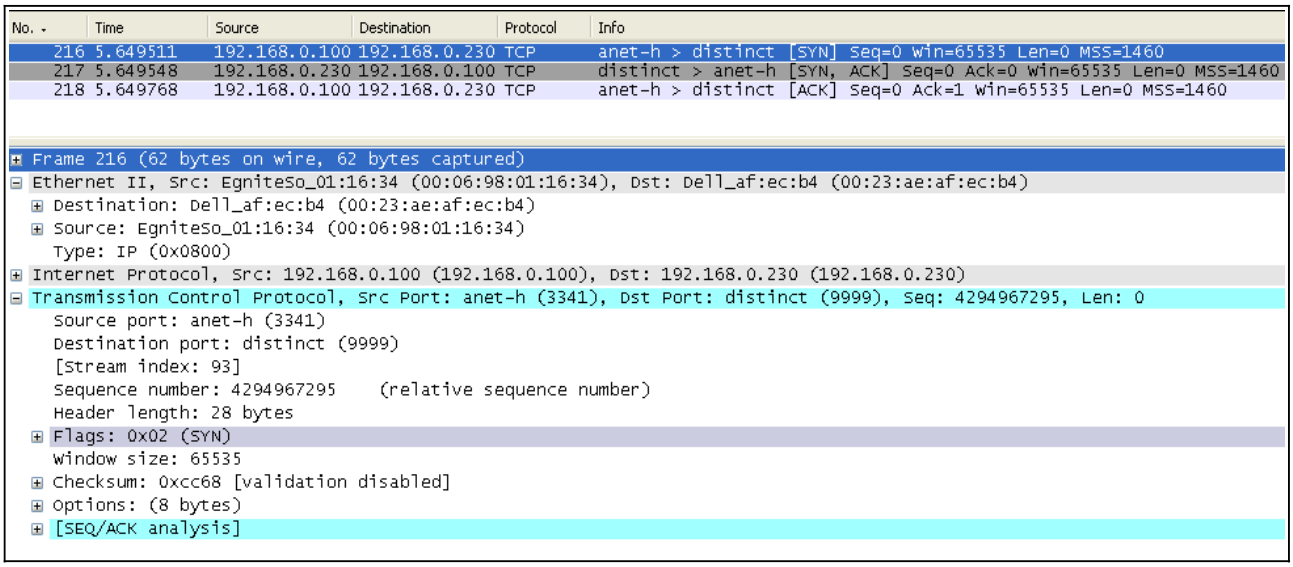


Fig. 7.4: *Intercambio de tramas TCP para establecer conexión TCP.*

- ***unsigned char*** *rellenaDatosTCP (****unsigned char*** ** pBuf,* ***unsigned char*** ** datos,* ***unsigned short*** *lon):* se envía una trama de tipo PSH ACK al servidor con los datos que contiene *datos* mediante el método *ethIf_send* y se espera un trama de tipo ACK del servidor para confirmar que le ha llegado la trama correctamente. En el caso de que no recibiera la trama ACK, se volvería a enviar la trama de datos PSH ACK, esperando de nuevo el ACK. Habrá un máximo de cinco intentos para enviar la trama. Si se agotan los intentos y sigue sin recibirse el ACK, se finalizará la conexión. Si la trama se ha enviado correctamente se devuelve un 1, si no, se devuelve un 0. En la imagen 7.5 se muestra el intercambio de tramas llevado a cabo:

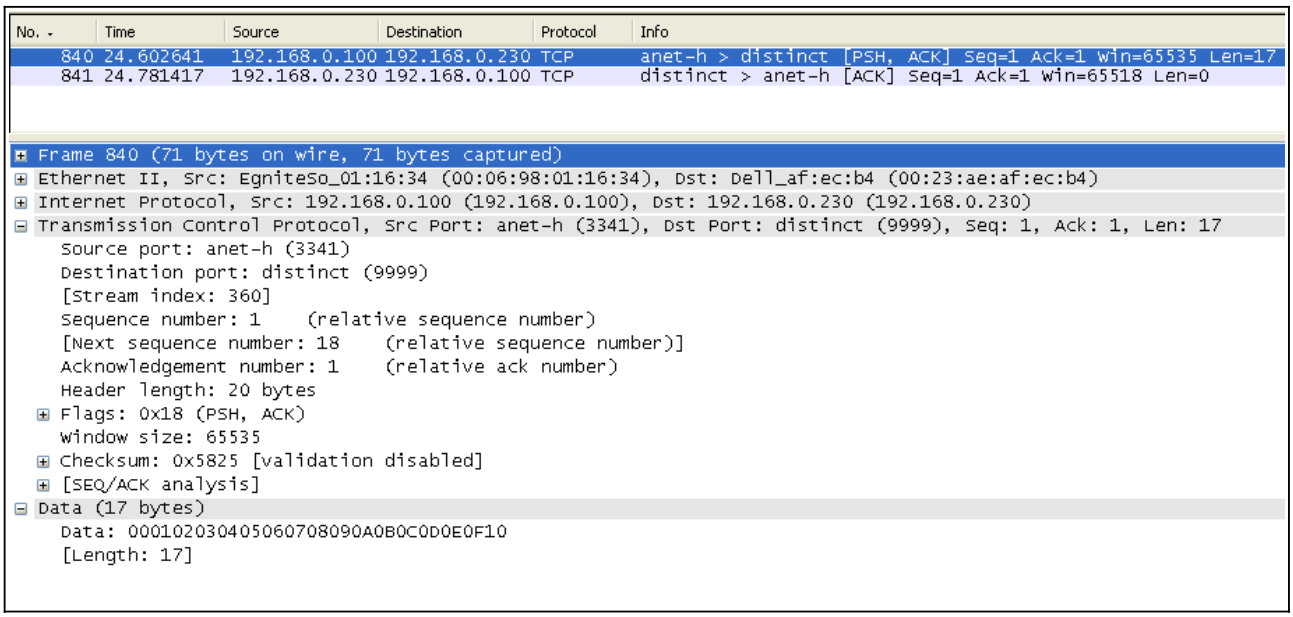


Fig. 7.5: *Intercambio de tramas TCP para mandar datos al servidor.*

7.4.3. Diagramas de Flujo de las Principales Funciones

En este apartado se explica, mediante diagramas de flujo, el principal método desarrollado para establecer una comunicación mediante el protocolo TCP. Éste se observa en las figuras 7.6. y 7.7.

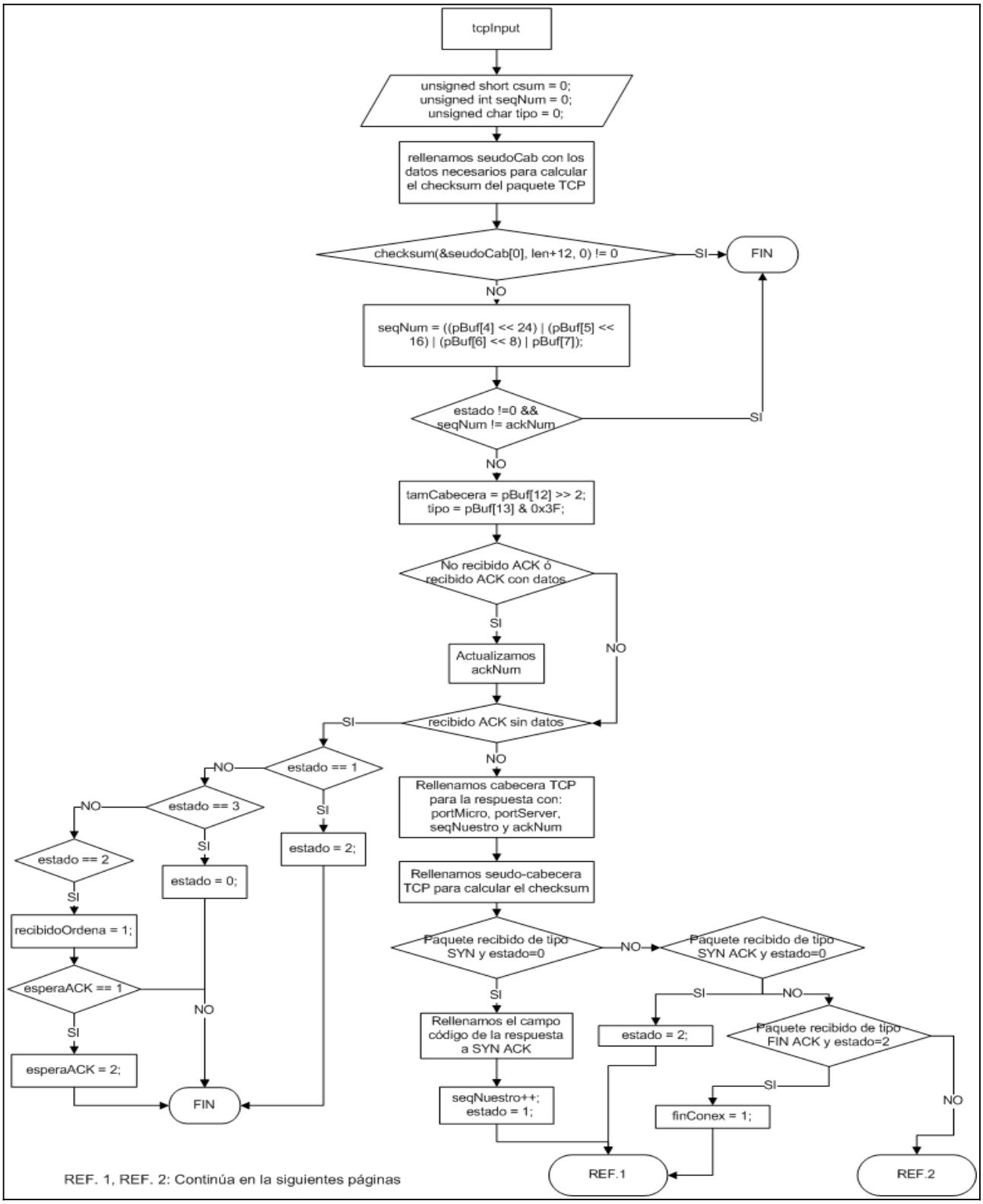


Fig. 7.6: *Diagrama de Flujo de tcpInput(unsigned char* pBuf, unsigned short len, unsigned int ipDest).*

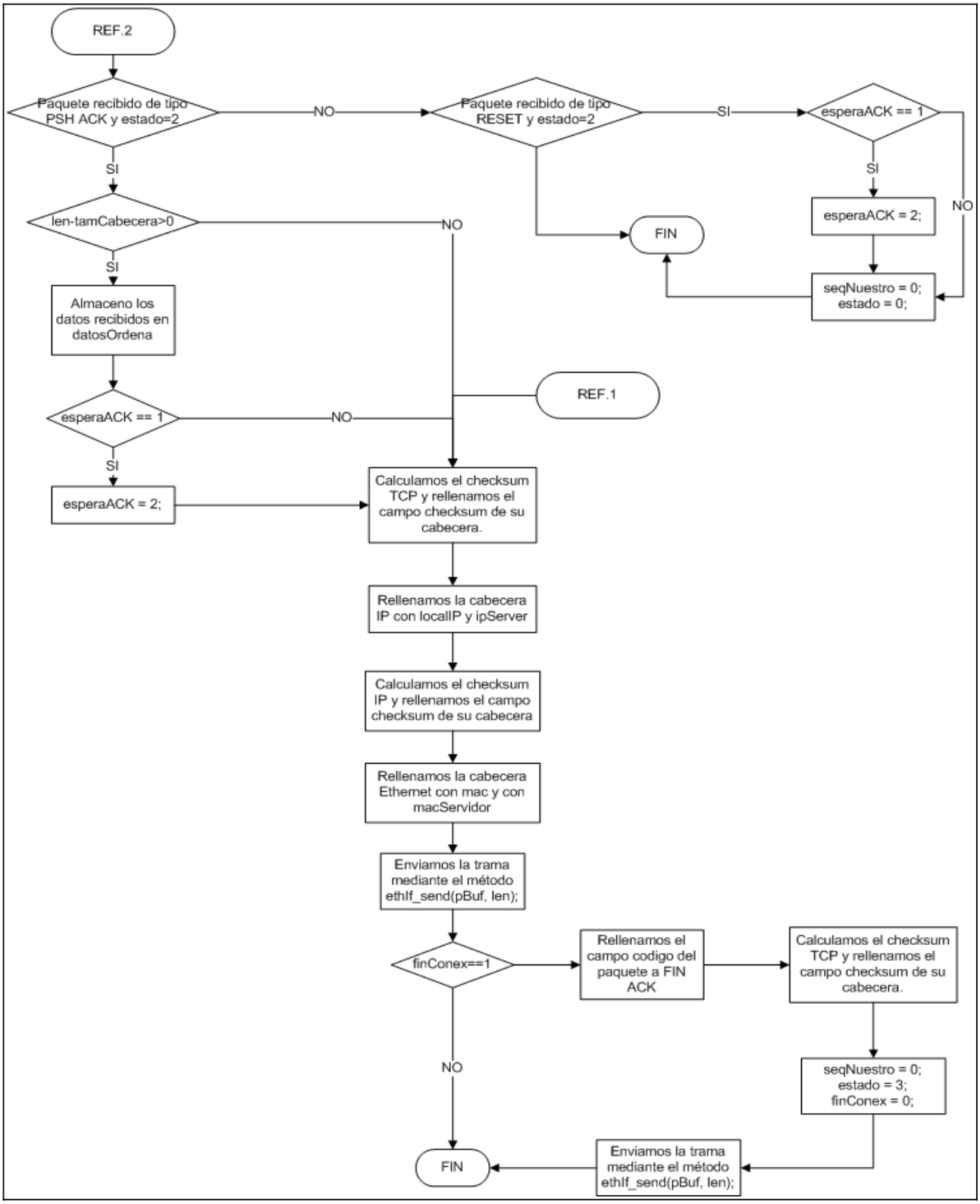


Fig. 7.7: *Cont. Diagrama de Flujo de tcpInput(unsigned char*pBuf, unsigned short len, unsigned int ipDest)*

7.5. Aplicación de escucha del PC

Como se ha comentado en el capítulo de introducción, el microcontrolador tendrá que comunicarse por red con un PC mediante el protocolo TCP/IP, para mandarle las credenciales de los usuarios. Por ello, necesitaremos desarrollar una aplicación de escucha en un puerto TCP que se ejecutará en un PC. Ésta aplicación se ha programado en Visual C#, y se explica con más detalle a en los siguientes apartados.

7.5.1. Introducción

Esta aplicación se hará cargo de responder a las peticiones que envíe el microcontrolador vía Ethernet. El microcontrolador iniciará siempre la comunicación, por lo que ésta aplicación se quedará escuchando en el puerto TCP definido, a la espera de una petición de conexión por parte del microcontrolador.

Fig. 7.8: *Pantalla Principal de la aplicación de escucha del PC.*

Como se observa en la figura anterior (*Fig.7.8.*), existe la posibilidad de cambiar el puerto TCP de escucha del PC, y de mostrar al usuario el puerto TCP definido actualmente. Estas dos opciones sólo estarán disponibles si no hay ninguna conexión TCP establecida con el microcontrolador.

En la figura 7.9 se observa la pantalla de configuración del puerto del PC que se mostrará cuando el usuario pulse el botón Configurar Puerto de Escucha de la pantalla principal. Al pulsar el botón Volver se mostrará la pantalla principal. Al pulsar el botón Aceptar, si el puerto que se

introduce es correcto, es decir, es un número comprendido entre 1 y 65535, se mostrará la pantalla principal y se dará el valor introducido por el usuario a una variable global que hay definida en el proyecto para interactuar con el puerto del servidor.

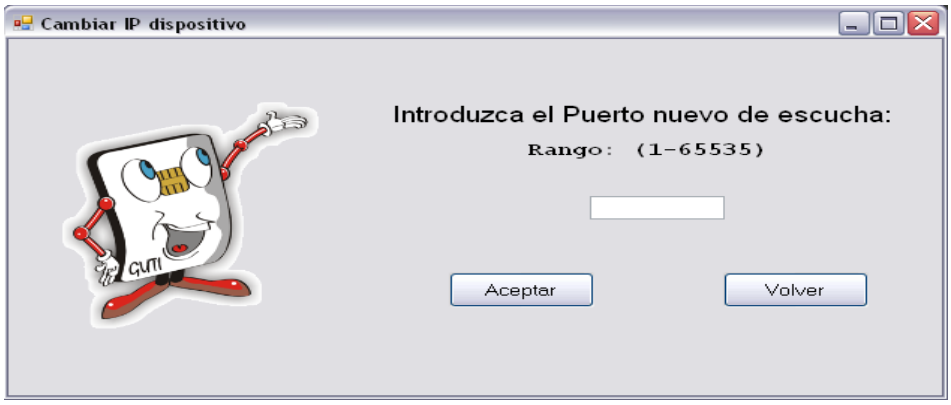


Fig. 7.9: *Pantalla de Configuración del Puerto de la aplicación de escucha del PC.*

En la figura 7.10 se observa la pantalla de vista del puerto actual del PC que se mostrará cuando el usuario pulse el botón Ver Puerto Actual de Escucha de la pantalla principal. Al cargar ésta pantalla se mostrará el valor de la variable global que almacena el puerto actual del servidor. Al pulsar el botón Volver se mostrará la pantalla principal.

Fig. 7.10: *Pantalla de Vista del Puerto de la aplicación de escucha del PC.*

En la figura 7.11 se observa que al pulsar el botón Iniciar Aplicación de la pantalla principal se desactivarán las opciones comentadas anteriormente, y se habilitará la opción de finalizar la aplicación. El PC se quedará a la espera de un petición de inicio de conexión por parte del microcontrolador. Éste proceso se explica con más detalle en el siguiente apartado mediante un diagrama de flujo.

Fig. 7.11: *Pantalla de Espera de conexión entrante de la aplicación de escucha del PC.*

El microcontrolador iniciará una conexión TCP con el PC cuando el usuario quiera enviar una de las credenciales. Cuando esto ocurra, la aplicación obtendrá los datos recibidos por red, cerrará la conexión, activando y desactivando las acciones pertinentes, y mostrará los datos. Éste proceso se explica con más detalle en el siguiente apartado mediante un diagrama de flujo.

Fig. 7.12: *Pantalla de Recepción del Número de Serie de la Tarjeta Inteligente del usuario.*

7.5.2. Diagramas de flujo

En este apartado se explica en detalle mediante diagramas de flujo como se ha desarrollado la aplicación principal.

En el proyecto se tendrán las siguientes variables globales:

- ***public static int*** *puerto = 9999*
- ***public static string*** *numSerie = “”*
- ***public static bool*** *conexionEstablecida = false*
- ***public static bool*** *esperaConexion = false*
- ***public static bool*** *primera = false*

En la figura 7.13 observamos el diagrama de flujo del método al que se llama cuando se pulsa el botón Iniciar Aplicación de la pantalla principal. Y en la figura 7.14 vemos el método que se ejecutará cuando se haga el Start del hilo (Hilo).

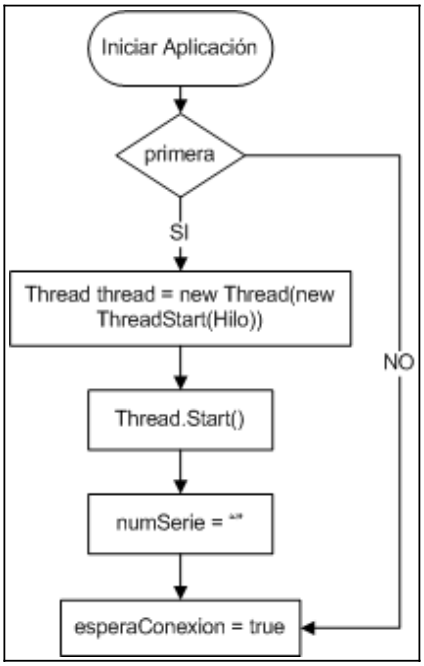


Fig. 7.13: *Diagrama de Flujo del botón Iniciar Aplicación de la pantalla de principal.*

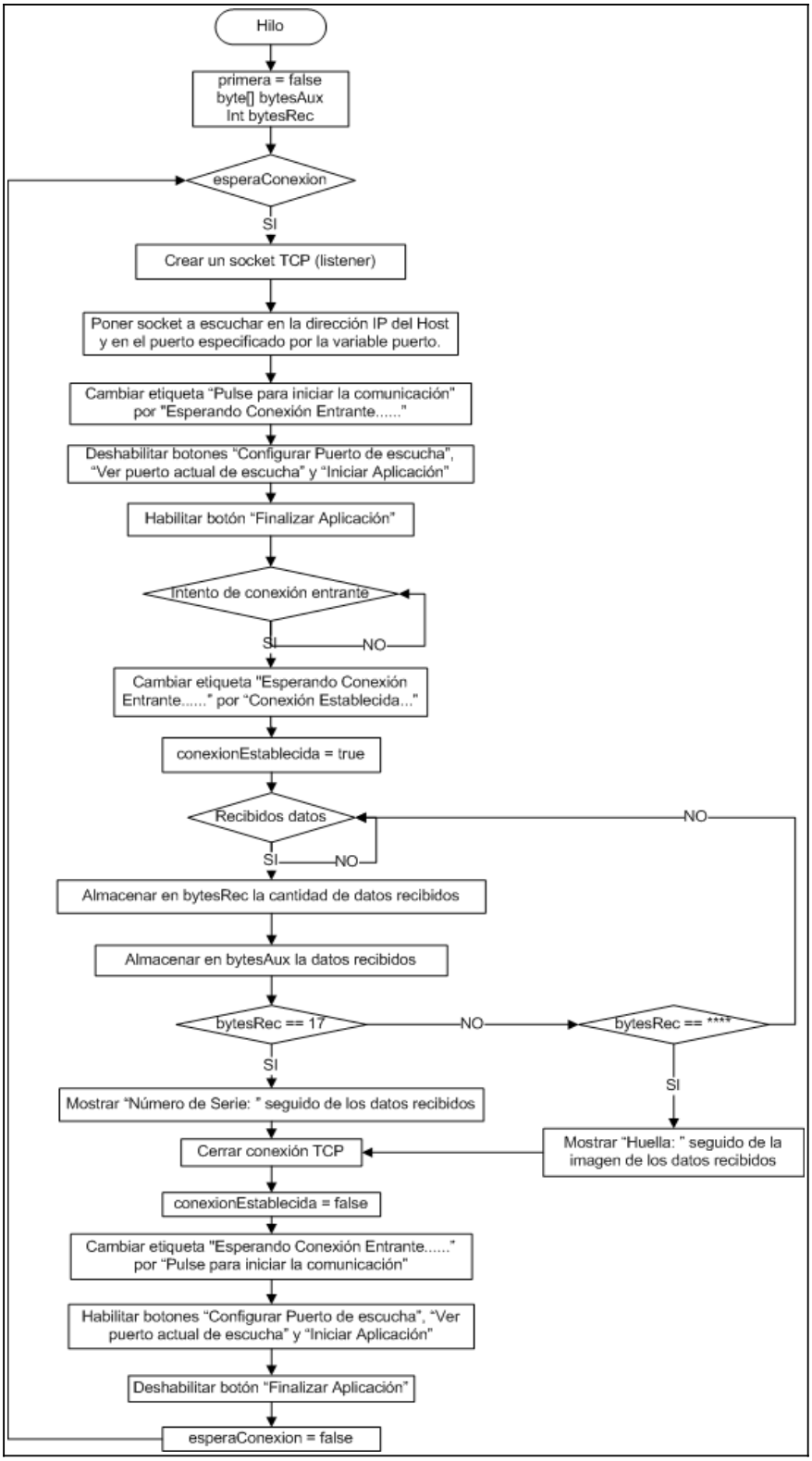


Fig. 7.14: *Diagrama del método Hilo().*

En la figura 7.15 observamos el diagrama de flujo del método al que se llama cuando se pulsa el botón Finalizar Aplicación de la pantalla principal.

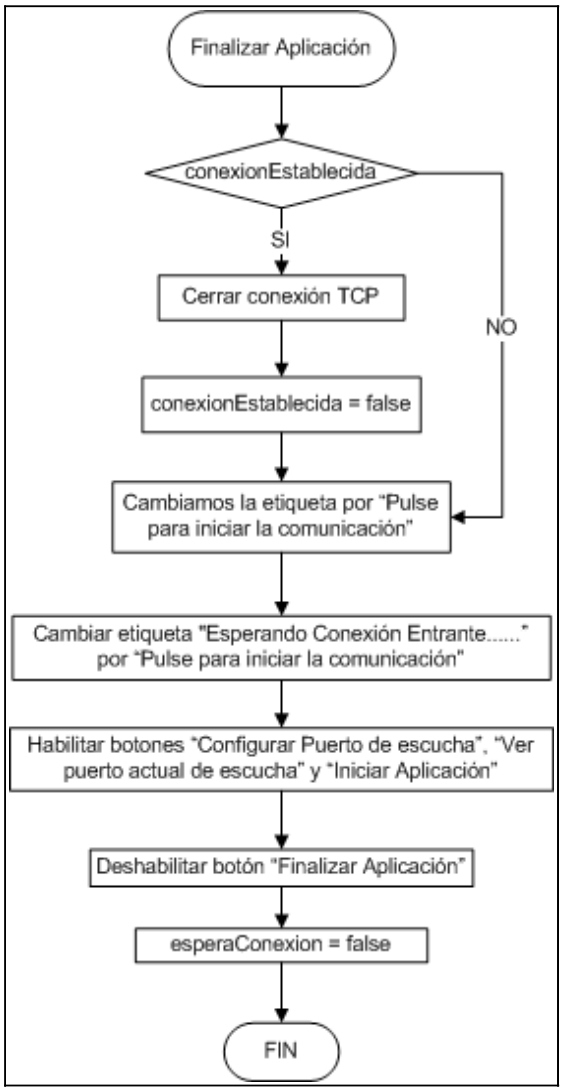


Fig. 7.15: *Diagrama de Flujo del botón Finalizar Aplicación de la pantalla de principal.*

7.6. Sistema Principal

Una vez se han desarrollado todos los módulos necesarios del microcontrolador para comunicarnos con las diferentes interfaces, y programada la aplicación de escucha necesaria para comunicarnos por red con un PC, procederemos a unir los distintos módulos. En primer lugar se han desarrollado unos métodos auxiliares que se explicarán en el primer y segundo apartado, incluyendo sus diagramas de flujo. Por último, se describe el método principal que lleva a cabo todas las funciones requeridas en este proyecto.

7.6.1. Métodos auxiliares para el Sistema Principal

En este apartado se describen los métodos auxiliares que utilizará el método principal para realizar todas las funciones necesarias en el sistema:

- ***unsigned char*** *comprobarIP (**void**):* Comprobamos si la dirección IP editada por el usuario (*ipASCII*) es correcta, es decir:
  - Se tienen que introducir cuatro números separados por tres puntos.
  - Cada número introducido tendrá que ser mayor o igual que 0 y menor o igual que 255.
  - El primer número y el último no podrán ser 0.

  Si es correcto, se devuelve un 1, si no, se devuelve un 0. En el apartado 7.5.2. se explica con más detalle mediante un diagrama de flujo.
- ***unsigned char*** *comprobarPuerto (**void**):* Comprobamos si el puerto editado por el usuario (*puerto*) es correcto, es decir, que sea mayor que 0 y que sea menor que 65536. Si es correcto, se devuelve un 1; si no, se devuelve un 0. En el apartado 7.5.2. se explica con más detalle mediante un diagrama de flujo.
- ***void*** *comprobarPantalla (**void**):* Comprobamos si el usuario ha pulsado la pantalla. Si es así, teniendo en cuenta la imagen actual (*estadoPantalla*) y las coordenadas del punto de la pantalla pulsado, se realizarán las acciones necesarias. En el apartado 7.5.2. se explica con más detalle mediante un diagrama de flujo.
- ***unsigned int*** *leerHuella()*: establece una conexión con el sensor de huella mediante el método *PTOpen()* y se queda a la espera de que se pose un dedo (*PTDetectFingerEx()*) si la operación no es cancelada por el usuario a través de la pantalla táctil. Si se ha posado un dedo almacena la imagen de la huella mediante el método *GrabImage()*. Finalmente se cierra la conexión con el sensor mediante el método *PTClose()*. Si se ha producido algún error en

el proceso, es decir, en *PTOpen, GrabImage, PTDetectFingerEx o PTClose;* o se canceló la operación por el usuario se devuelve un código de error. Si no se devuelve un 0. En el apartado 7.5.2. se explica con más detalle mediante un diagrama de flujo.

7.6.2. Diagramas de Flujo de los Métodos Auxiliares

En este apartado se explican, mediante diagramas de flujo, los métodos auxiliares desarrollados para el correcto funcionamiento del sistema principal.

En la figura 7.16 observamos el método comprobarPuerto(), en la figura 7.13 el métodos comprobarIP(), y en las figuras 7.14, 7.15,7.16,7.17 el método comrpobarPantalla():

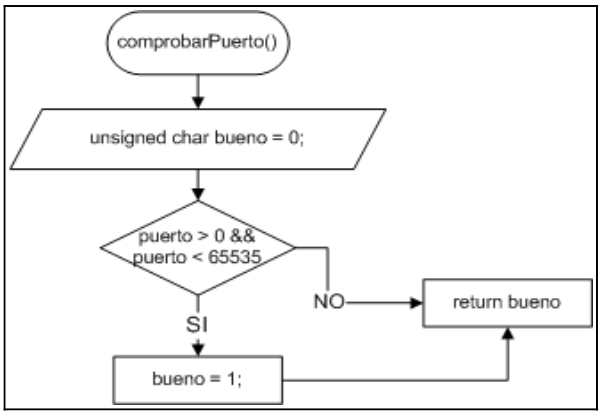


Fig. 7.16: *Diagrama de Flujo del Método comprobarPuerto()*

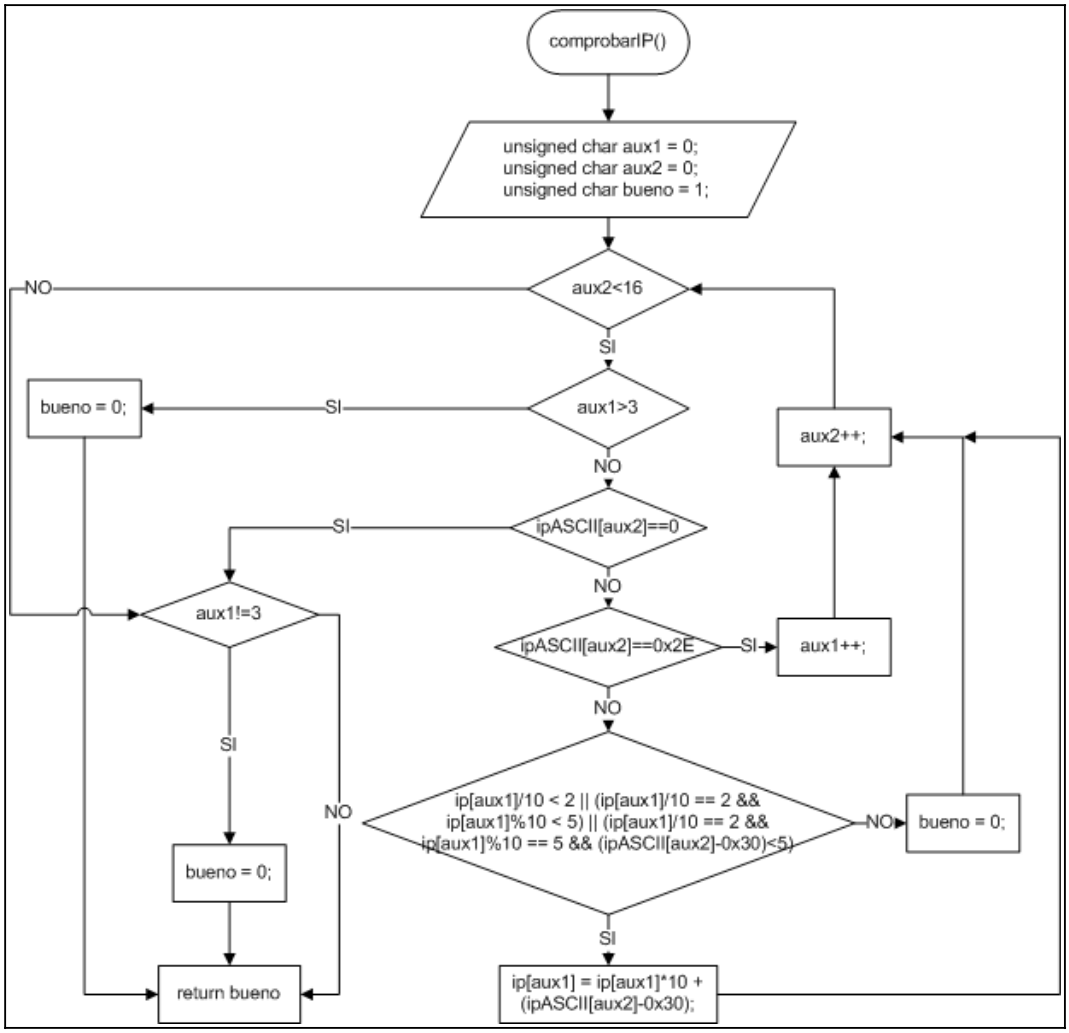


Fig. 7.17: *Diagrama de Flujo del Método comprobarIP()*

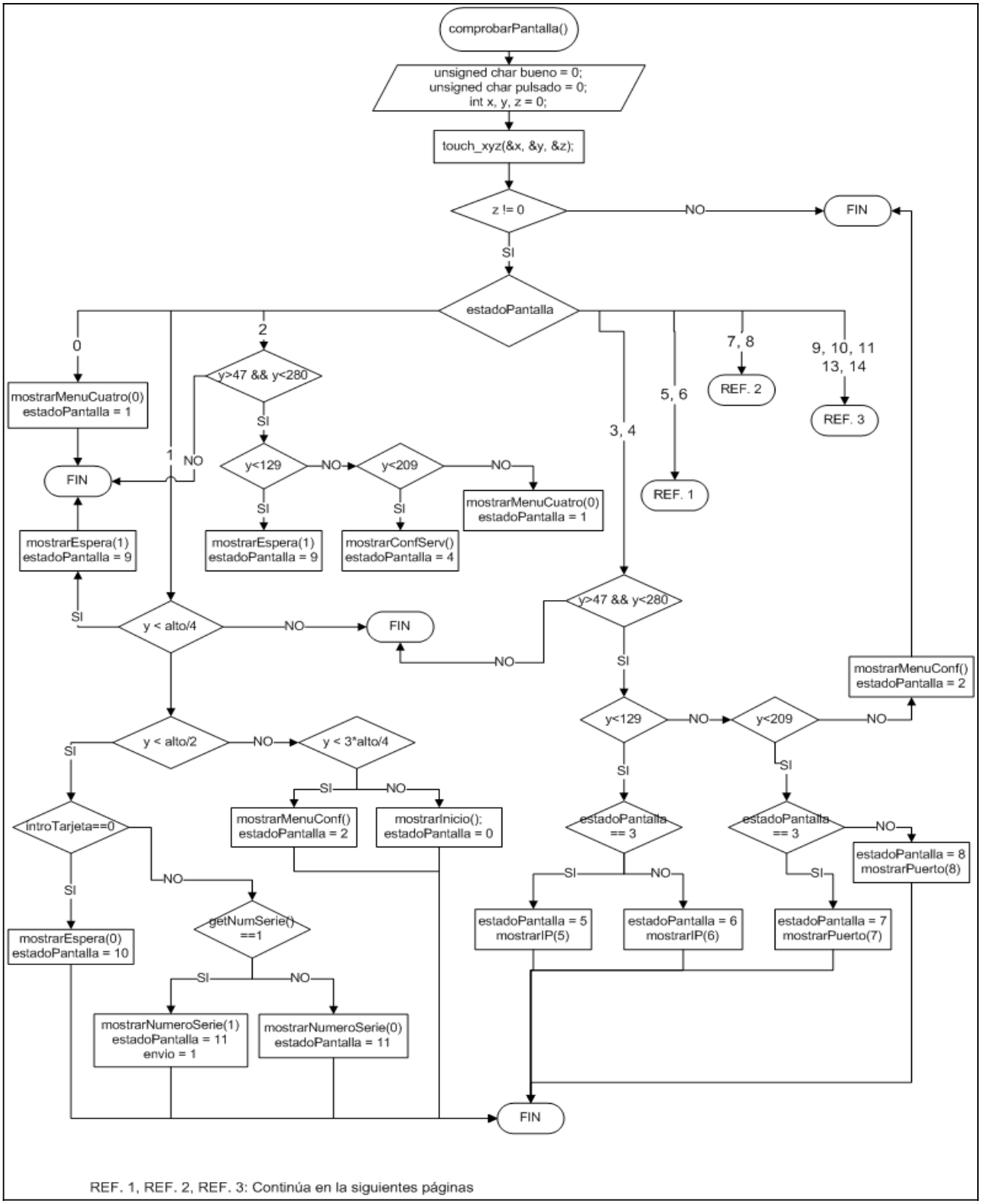


Fig. 7.18: *Diagrama de Flujo del Método comprobarPantalla()*

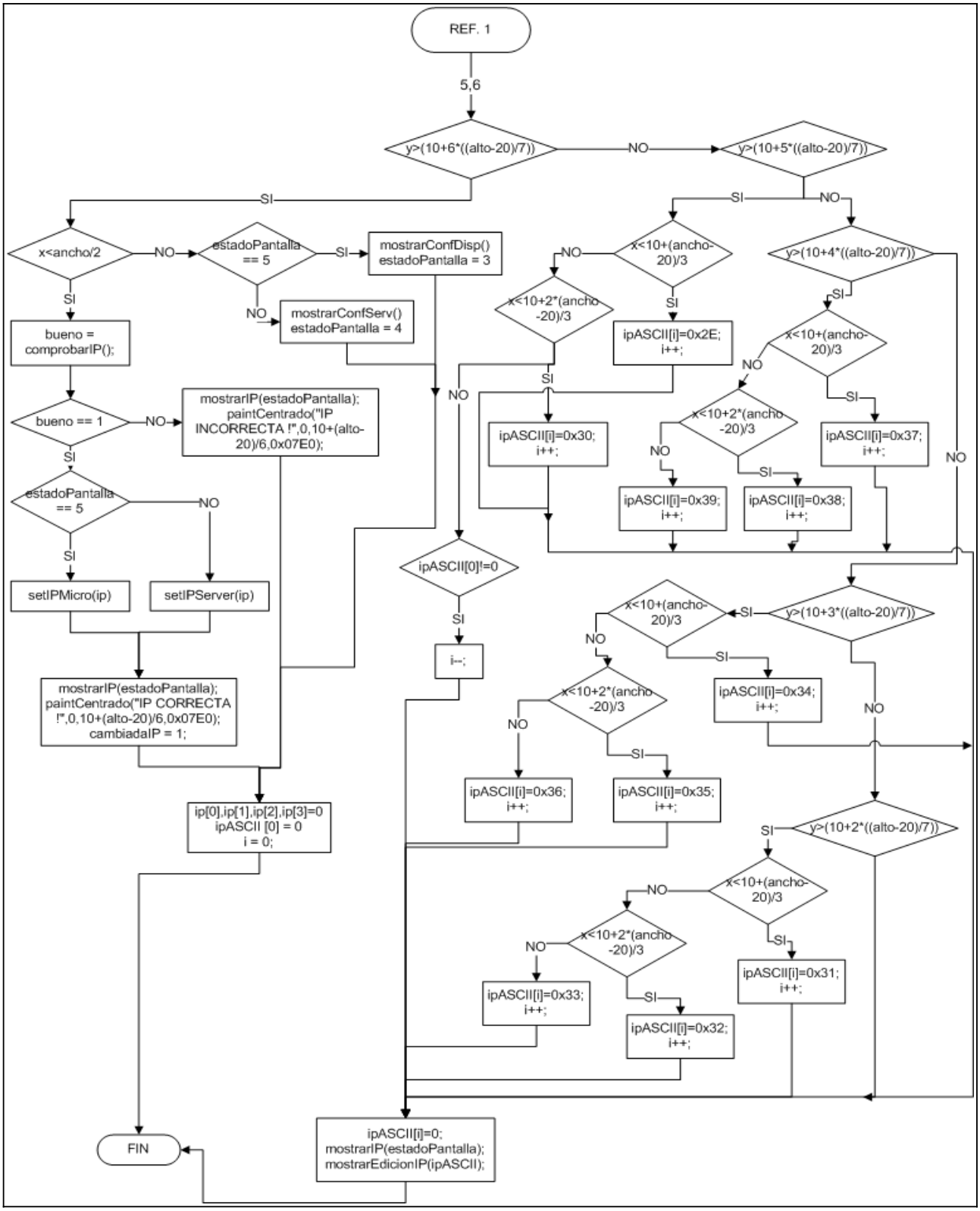


Fig. 7.19: *Diagrama de Flujo del Método comprobarPantalla() (Continuación)*

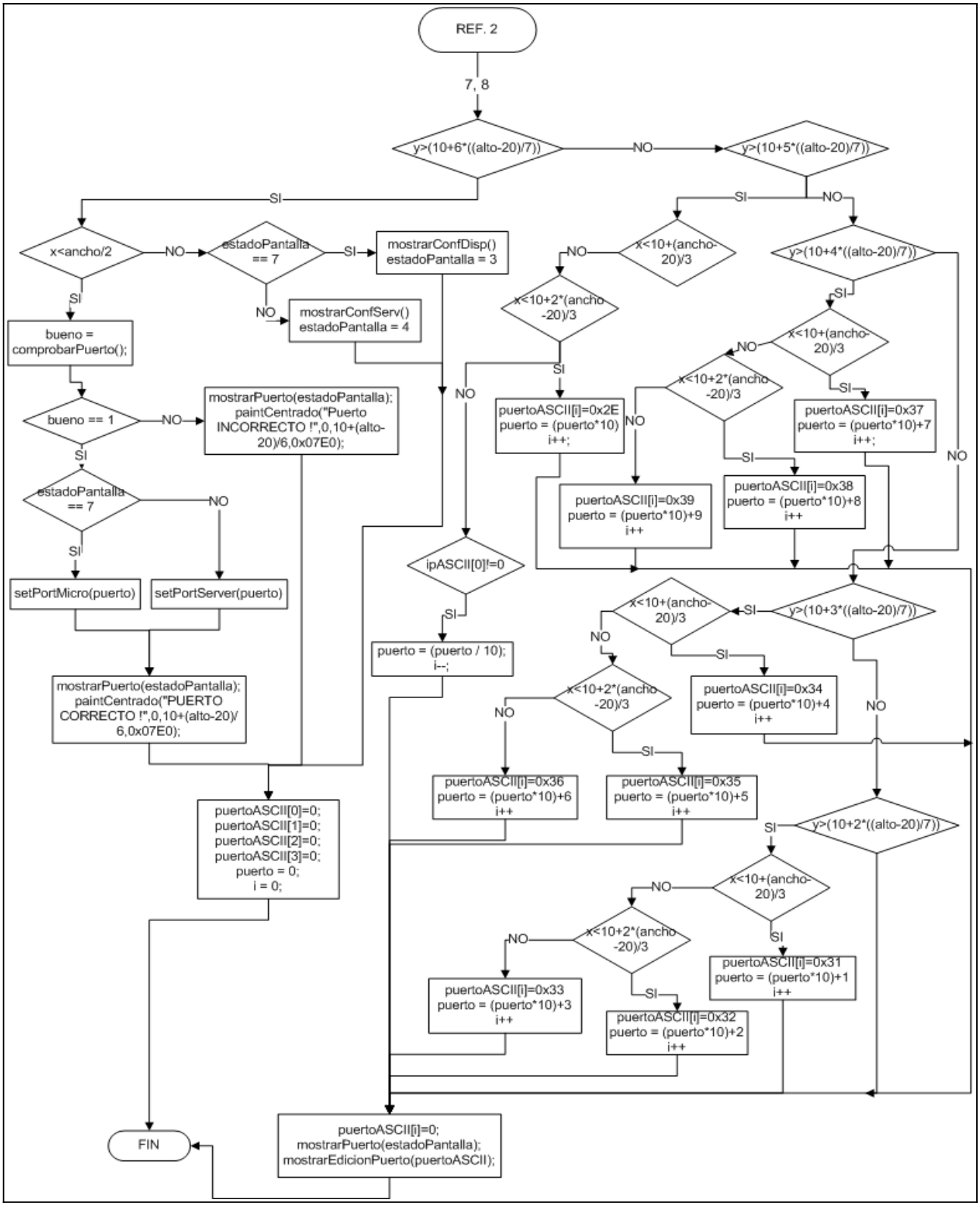


Fig. 7.20: *Diagrama de Flujo del Método comprobarPantalla() (Continuación)*

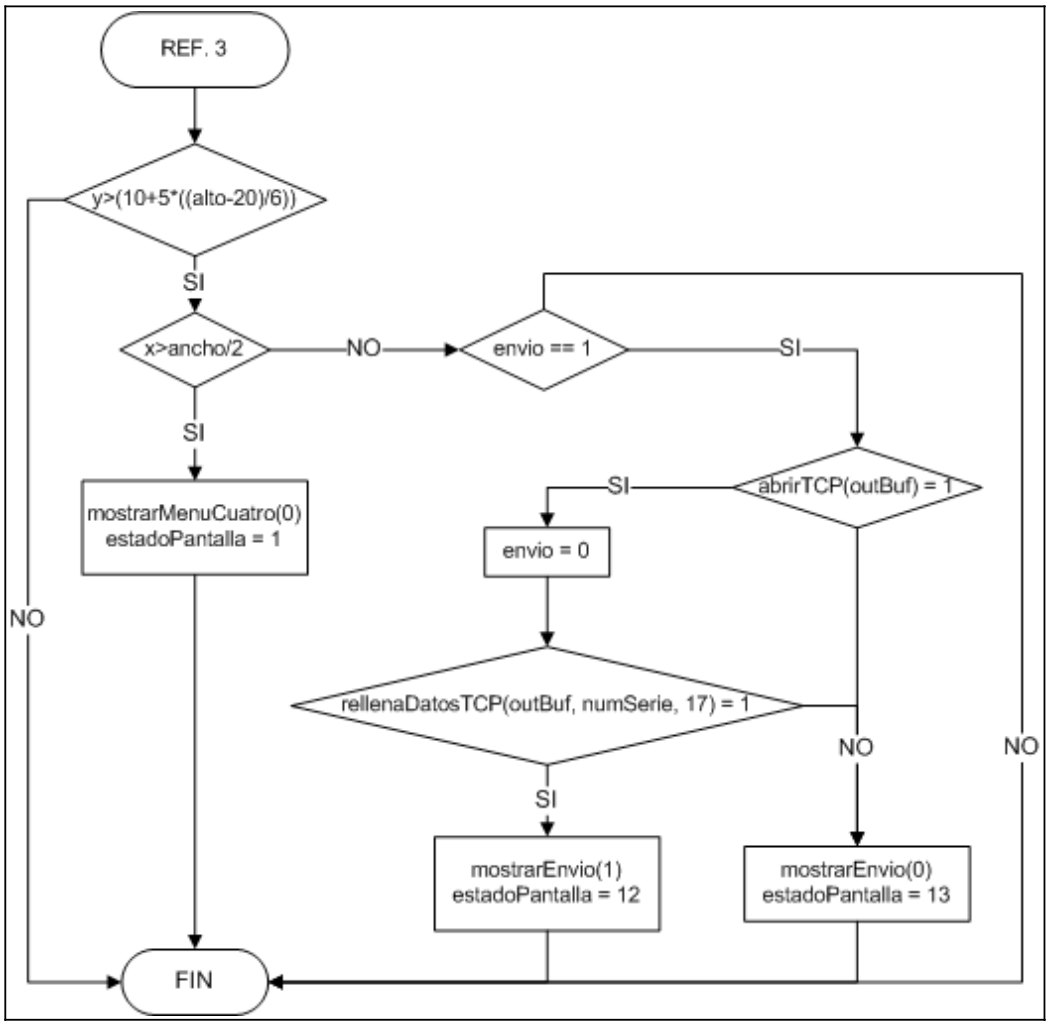


Fig. 7.21: *Diagrama de Flujo del Método comprobarPantalla() (Continuación)*

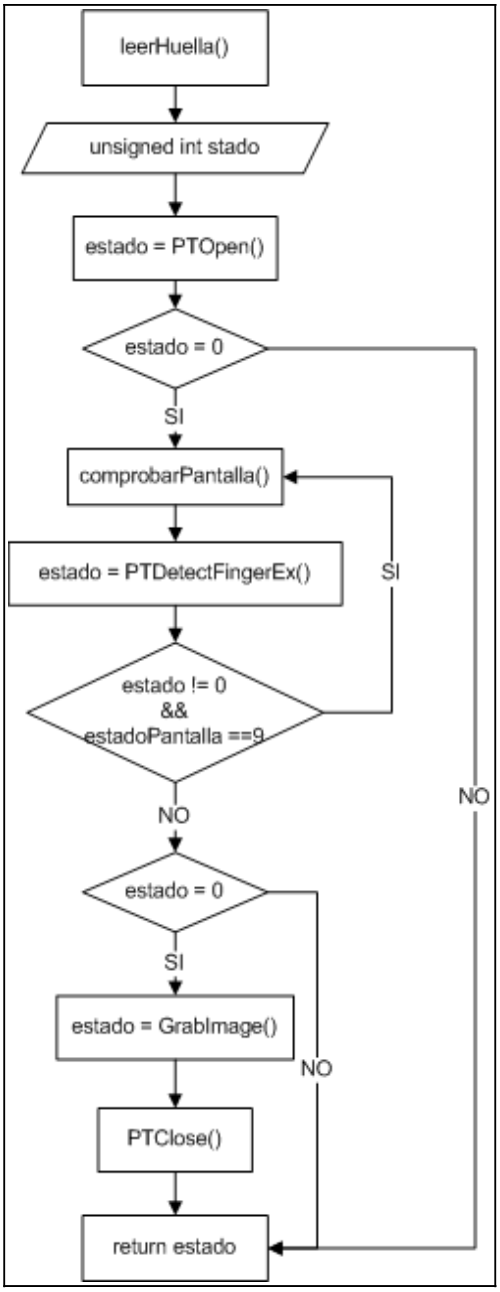


Fig. 7.22: *Diagrama de Flujo del Método leerHuella()*

7.6.3. Diagrama de Flujo del Método Principal

El método principal se hará cargo en primer lugar de configurar las interfaces USB, Ethernet y SPI. Y, en segundo y último lugar, tendrá que comprobar si se ha insertado una tarjeta inteligente en el lector, tendrá que comprobar si el usuario interactúa con la pantalla, y tendrá que comprobar si se recibe algo vía red. Éste método se explica con más detalle en la figura 7.23 mediante un diagrama de flujo.

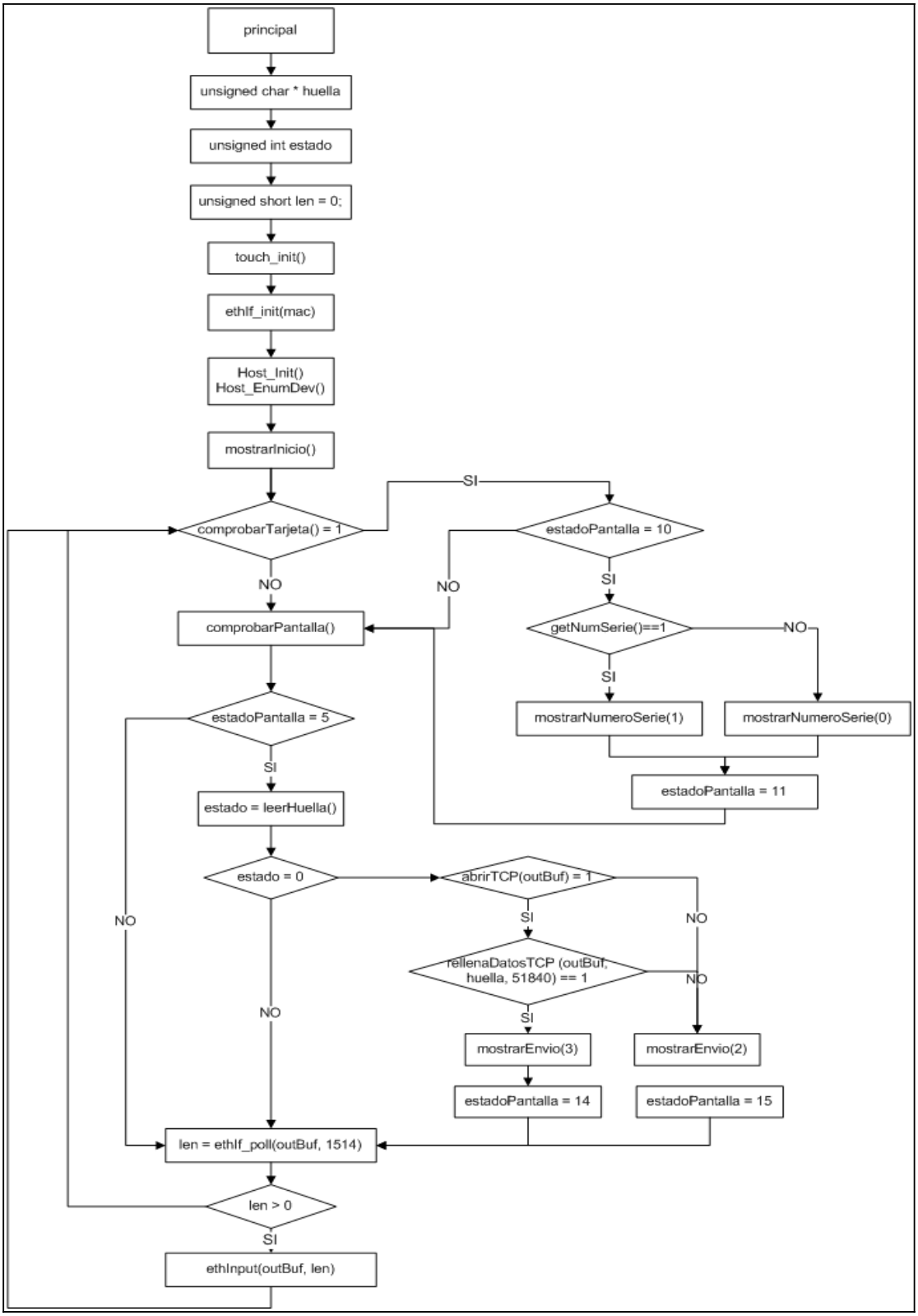


Fig. 7.23: *Diagrama de Flujo del Método Principal*

8. Presupuesto

En el presente capítulo se presenta un cálculo aproximado del coste de este proyecto. Para la evaluación de dicho coste es necesario un desglose del proyecto en tareas, a través de la planificación del mismo y la evolución temporal seguida. Cada tarea será cuantificada en función de su duración y los recursos consumidos, proporcionando de manera global un presupuesto aproximado del trabajo desarrollado.

8.1. Descomposición de actividades

A continuación se exponen las cinco tareas llevadas a cabo:

- Fase 1: Documentación para realizar el proyecto y estudio del estado del arte.
- Fase 2: Aprendizaje del entorno de desarrollo.
- Fase 3: Implementación de la aplicación
- Fase 4: Pruebas y depuración del código obtenido.
- Fase 5: Documentación y memoria.

Las siguientes tablas recogen el presupuesto ofrecido por el desarrollo del PFC, se destaca que el precio final sería el precio que ofrecería una consultoría, no el precio real gastado en el desarrollo.

PRESUPUESTO DEL PROYECTO

1. Autor: **Sandra Cid Pantoja**
2. Departamento: **Dpto. De Tecnología Electrónica**

3. Descripción del Proyecto:

Se implementarán distintas interfaces de dispositivos para un sistema empotrado basado en un microcontrolador ARM7. El microcontrolador se comunicará con un lector de tarjetas, un sensor de huella y una pantalla táctil.

- Título: **Implementación de Interfaces de Dispositivos para Sistemas Empotrados basados en Microcontroladores ARM7.**
- Duración(meses): **10**

  Tasas de costes Indirectos: **20%**

4. Presupuesto total del Proyecto (valores en Euros):

43540,12 **Euros**

5. Desglose presupuestario (costes directos):

PERSONAL

Apellidos y nombre	N.I.F. (no rellenar – sólo a título informativo)	Categoría	Dedicación (hombres mes)	Coste hombre mes	Coste (Euro)
Sánchez Reíllo, Rául		Ingeniero Senior	2	4.289,54	8.579,08
Cid Pantoja, Sandra		Ingeniero	10	2.694,39	26.943,90
				Total	**35.522,98**

EQUIPOS Y SOFTWARE

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable
Placa de desarrollo MCB2470	250,00	100	8	60	33,33
Lector de Tarjetas SDI010	55,53	100	8	60	7,40
Lector de Tarjetas SCL010	60,42	100	8	60	8,06
Sensor de Huella + SDK de UPEK	2.500,00	100	4	60	166,67
Analizador de Protocolo USB	2.850,00	100	2	60	95,00
PC de mesa con Windows XP	900,00	100	10	60	150,00
Licencia Keil	1.800,00	100	10	60	300,00
				Total	**760,46**

Fórmula de Cálculo de la Amortización:

A/B x C x D

A = nº de meses desde la fecha de facturación en que el equipo es utilizado

B = periodo de depreciación (60 meses)

C = coste del equipo (sin IVA)

D = % del uso que se dedica al proyecto (habitualmente 100%)

6. Resumen de costes:

Presupuesto Costes Totales	Presupuesto Costes Totales
Personal	35.522,98
Amortización	760,46
Costes Indirectos	7.256,68
Total	**43.540,12**

El presupuesto total de este proyecto asciende a la cantidad de CUARENTA Y TRES MIL QUINIENTOS CUARENTA EUROS CON DOCE CÉNTIMOS.

Leganés, Septiembre de 2010

El ingeniero proyectista

Fdo. Sandra Cid Pantoja

9. Conclusiones y Líneas de Futuro

9.1. Conclusiones

La realización de este proyecto ha permitido profundizar los conocimientos personales sobre Microcontroladores ARM7 y el entorno de desarrollo utilizado para su programación (µVision4 de Keil).

El proyecto también ha servido para adquirir experiencia en el uso de un entorno de programación visual, Microsoft Visual Studio 2008. Puesto que se ha diseñado la aplicación de escucha del PC se han adquirido los conocimientos necesarios para programar aplicaciones gráficas sencillas y Sockets TCP en un entorno Windows, utilizando el lenguaje de programación Visual C#.

En el presente Proyecto Fin de Carrera se detallan todos los pasos llevados a cabo para el correcto funcionamiento del sistema final. Para ello ha sido necesario un estudio concienzudo de las diferentes interfaces de los dispositivos del sistema y de sus protocolos de comunicación.

La realización de este proyecto también ha permitido ampliar conocimientos sobre biometría y tarjetas inteligentes. Anteriormente a su desarrollo se realizó un estudio de las diferentes técnicas de identificación biométrica, y también de tarjetas, analizándose los bloques funcionales de una tarjeta inteligente con contactos.

A partir de este estudio se han conseguido configurar todas las interfaces necesarias en el sistema desarrollado para poder iniciar una comunicación con ellas.

En el caso de USB, no se tenían conocimientos avanzados sobre el protocolo. Se tuvo que realizar un estudio exhaustivo sobre éste. Además, se tuvo el inconveniente de tener que descubrir el protocolo a bajo nivel para comunicarse con el lector de tarjetas inteligentes y las tarjetas, tanto con contactos, como sin contactos. A pesar de estas dificultades, se ha conseguido, finalmente, establecer esta comunicación para conseguir las credenciales de los usuarios vía USB.

En el caso Ethernet, se tenían conocimientos avanzados sobre el protocolo TCP/IP, pero

siendo un protocolo tan complejo, ha sido complicado desarrollar todas sus funcionalidades para un microcontrolador. Paralelo a esto, se ha tenido que desarrollar la aplicación de escuche para el PC, la cual se ha programado en Visual C#, un lenguaje de programación que no se había utilizado hasta ahora. Finalmente, se ha conseguido establecer comunicación vía red entre el microcontrolador y el PC para poder enviar las credenciales obtenidas vía USB anteriormente.

En el caso de la pantalla táctil, debido a la poca documentación encontrada, ha sido difícil obtener la coordenadas del punto pulsado por el usuario, pero, finalmente, se encontró una relación entre ellas para conseguir las coordenadas en píxeles exactas del punto. Además, como se han utilizado imágenes fijas para el desarrollo de nuestro interfaz gráfico del microcontrolador, se ha ocupado prácticamente toda la memoria ROM interna del microprocesador. Lo que es un problema, ya que no podríamos desarrollar más funcionalidades para el microcontrolador a partir de éste sistema.

En el caso del sensor de huella, ya que se facilitaba una librería desarrollada a nivel bajo para un microcontrolador distinto al nuestro, ha sido difícil adaptarla a nuestra placa. Pero finalmente, gracias a la documentación facilitada por el fabricante, se ha conseguido establecer comunicación con el sensor y capturar la imagen de la huella.

9.2. Líneas de Futuro

Como se ha comentado en el apartado anterior, el tamaño de la memoria ROM interna del microprocesador nos limita mucho en cuanto a desarrollar sistemas más complejos que el expuesto en este proyecto. Para solucionar esto, se tendría que configurar la memoria ROM externa de la que dispone el microprocesador para poder comunicarnos con ella. Además, se debería de conseguir que el programa que va a ejecutar el microcontrolador se repartiera entre la dos memorias, la interna y la externa. Si esto se consiguiera, tendríamos mucho margen para poder desarrollar sistemas complejos para el microcontrolador.

Sería interesante, ya que el microcontrolador es capaz de comunicarse por red, poder configurar opciones de él desde el PC. Algunas de ellas serían: las direcciones IP y puertos que utiliza para la comunicación por red, tanto de él, como del PC; las imágenes que se mostrarán en la pantalla; o, incluso, poder cargar el programa que va a ejecutar el microcontrolador vía red.

Este sistema podría llegar a utilizarse para multitud de aplicaciones, como son, un sistema de autenticación, un sistema de control de acceso, un sistema que informe a los usuarios de

noticias en tiempo real, ó un sistema de reclutamiento.

Simultáneamente a este proyecto se ha estado implementando un sistema más complejo, el cual sigue desarrollándose en la actualidad. Debido a que este sistema forma parte de un proyecto I+D con políticas de confidencialidad, no se puede entrar en detalles sobre en que consiste.

10. Bibliografía

[1] ***Identificación Biométrica y su unión con las Tarjetas Inteligentes***. Raúl Sánchez Reíllo. Ágora SIC Divulgación. Abril 2000.

[2] ***Identificación biométrica***. Joaquín González Rodríguez. 14/07/2000. **http://www.instisec.com/publico/verarticulo.asp?id=20**

[3] ***Biometría: ¿Realidad o ficción?*** Orestes Sánchez. 22/12/2005. **http://sociedaddelainformacion.telefonica.es/jsp/articulos/detalle.jsp?elem=1744**

[4] ***Técnicas de seguridad biométricas.*** Network World. 01/09/2004. **http://www.idg.es/comunicaciones/articulo.asp?id=161095**

[5] ***Identificación biométrica, la llave del futuro.*** Elvira Fernández. 2000. **http://www.cienciadigital.es/hemeroteca/reportaje.php?id=83**

[6] ***Biometría y la identificación automática de personas***. Herbert Saal Gutierrez. 08/01/07. **http://capacitacionencostos.blogia.com/2007/010814-biometria-y-la-identificacion-automatica-de-personas.php**

[7] ***Biometría e identificación de personas***. Biometría. Control de Acceso y Presencia. RFID. **http://www.kimaldi.com/kimaldi/area_de_conocimiento**

[8] ***La Tecnología de las Tarjetas Inteligentes.*** Raúl Sánchez Reíllo. Servicio de Publicaciones de la E.T.S.I. Telecomunicación. Madrid. 1999.

[9] ***Breve introducción al protocolo TCP/IP.*** **http://www.ace.ual.es/~leo/redes/Rel3-01-02.pdf**

[10] ***Apuntes sobre IP de la asignatura Telemática II de I.T.T: Telemática.*** Universidad Carlos III de Madrid. 2007/08.

[11] ***Especificación de Bus Serie Universal Revisión 1.1.*** Compaq, Intel, Microsoft y NEC. 23 de Septiembre de 1998. **http://www.usb.org/developers/docs/**

[12] ***Especificación de Bus Serie Universal Revisión 2.0.*** Compaq, Hewlett-Packard, Intel, NEC, Microsoft, Lucent y Philips. 27 de Abril de 2000. **http://www.usb.org/developers/docs/**

[13] ***User Manual LPC2478***. 26/08/2009. **http://www.standardics.nxp.com/support/documents/microcontrollers/pdf/user.manual.lpc24xx.pdf**

[14] ***Datasheet LPC2478***. 11/11/2008. **http://www.nxp.com/documents/data_sheet/LPC2478.pdf**

[15] ***MCB2470 Evaluation Board***. Keil, An ARM company. 2010. **http://www.keil.com/mcb2470/**

[16] ***Datos del lector de tarjetas SDIO10.*** SCM Microsystems. **http://www.scmmicro.com/security/view_product_en.php?PID=19**

[17] ***Datos del lector de tarjetas SCL010.*** SCM Microsystems. **http://www.scmmicro.com/products-services/smart-card-readers-terminals/contactless-dual-interface-readers/scl010.html**