SARIC

Sistema de Accesos Restringidos a Instalaciones Civiles

Proyecto Fin de Carrera

Autores: **Iván García García**
Rubén Héctor García Ortega

Directores: **D. Andrés María Roldán Aranda**
D. Jose Enrique Cano Ocaña

Dpto. de Ciencias de la Computación e Inteligencia Artificial
Dpto. de Electrónica y Tecnología de Computadores
E.T.S. de Ingenierías Informática y de Telecomunicación
UNIVERSIDAD DE GRANADA

Índice general

Índice de figuras

Agradecimientos

Nos gustaría comenzar dando las gracias a nuestros directores de proyecto, *D. Andrés María Roldán Aranda*, profesor del Dpto. de Electrónica y Tecnología de Computadores, y *D. José Enrique Cano Ocaña*, profesor titular del Dpto. de Ciencias de la Computación e Inteligencia Artificial, por habernos dado la oportunidad de realizar el presente trabajo, siendo su aportación a éste esencial en algunas partes críticas. Igualmente, debemos agradecer la ayuda recibida por los alumnos de Ingeniería Electrónica y en especial la de *Simón Perán Sánchez* por su inestimable colaboración en la fabricación de los dispositivos electrónicos, sin la cual este proyecto no habría sido posible, y a *Anna Peña Martínez* que ha revisado tan minuciosamente este documento. Gracias también a *Olga Muñoz Ramos* por prestar su voz para la realización del video demostrativo del proyecto.

Así mismo, este proyecto es el resultado de dos largos años de trabajo que sirve de culminación de nuestros estudios de Ingeniería Informática en la Escuela Técnica Superior de Ingeniería Informática de Granada. Por ello sería injusto no agradecer la labor desempeñada por los profesores que han participado en nuestra formación como ingenieros, así como los compañeros con los que hemos tenido la oportunidad de coincidir durante estos años.

Fuera del ámbito informático, gracias a nuestros padres, pues sin ellos no seriamos las personas que somos hoy. Todo este trabajo esta dedicado ellos agradeciéndoles todo lo que nos han dado hasta ahora y esperando que se sientan orgullosos no solo de este momento sino de todos los que formaran parte de nuestras vidas de ahora en adelante.

Abstract

To control and to restrict people access to concrete areas of an enclosure, campus or building is a widely requested service that must satisfy very strict security and reliability requirements. It is not a trivial problem, because the available solutions lack the necessary flexibility for their application in complex surroudings.

The project: *Restricted Access System for Civil Facilities* (SARIC) is a robust, safe, centralized and fully scalable integral solution of access control, which is able to be adapted to any area or opening device. Each user has a smartcard which, combined with encryption algorithms, provides the system with the necessary confidence. The kernel is an administration program that allows to manage all the access points, the system users and the grants associated to each one of them, obtaining therefore a totally centralized system. The system scalability remains guaranteed thanks to two different control devices: the first one has a low cost and the second is equipped with remote administration capabilities.

The present document approaches the complete project description, as well as the analysis and design of the software and the hardware that it includes, the tools and methodologies that have been used and the planning along the developement time.

The heterogeneous nature of the project and the fellowship of two departments of the University of Granada involved in differents areas of knowledge, has provided a very wide perspective and they have allowed us to put in practice many of the knowledges acquired throughout the degree and extend them in areas where training is less important.

As a final finding a fully operative ready-for-use system has been obtained. Nevertheless it is possible to extend it with new identification characteristics such as biometric readers or face recognition devices and possible adaptations for people with any handicap.

Resumen

Controlar y restringir el acceso de personas a zonas concretas de un recinto, campus o edificio es un servicio ampliamente solicitado, que debe satisfacer requisitos muy estrictos de seguridad y fiabilidad. No se trata de un problema trivial, pues las soluciones disponibles carecen de la flexibilidad necesaria para su aplicación a entornos complejos.

El proyecto: *"Sistema de Acceso Restringido a Instalaciones Civiles"*(SARIC), es una solución integral de control de acceso robusto, seguro, centralizado y totalmente escalable capaz de adaptarse a cualquier recinto o mecanismo de apertura. Cada usuario dispone de una tarjeta inteligente que, combinado con algoritmos de encriptación, dota al sistema de la seguridad necesaria. El núcleo es un programa de gestión que permite administrar los puntos de acceso, los usuarios del sistema y los permisos asociados a cada uno de ellos obteniendo así un sistema completamente centralizado. La escalabilidad del sistema queda garantizada gracias a dos dispositivos diferentes de control: uno de bajo coste y otro con capacidad de administración remota.

En el presente documento se aborda la descripción completa del proyecto, así como el análisis y diseño del software y del hardware que lo componen, las herramientas y metodologías que se han empleado y la planificación temporal del mismo.

La naturaleza multidisciplinar del proyecto hace necesaria una participación interdepartamental, habiendo intervenido en el mismo dos departamentos de la Universidad de Granada de áreas diferentes que han proporcionado una perspectiva muy amplia y han permitido poner en práctica muchos de los conocimientos adquiridos a lo largo de la carrera así como ampliar estos en materias donde la formación es menor.

Como resultado final se ha obtenido un sistema completamente operativo y listo para su instalación y uso. Sin embargo es posible ampliarlo con nuevas características de identificación como lectores biométricos o de reconocimiento facial y adaptaciones para personas con alguna discapacidad.

1 Introducción

SARIC (Sistema de Acceso Restringido a Instalaciones Civiles) es un sistema capaz de centralizar y controlar diferentes puntos de acceso que pueden estar distribuidos por distintos recintos. Este control es posible gracias al uso de modernas tecnologías que permiten realizar sistemas interconectados seguros sin necesidad de complejas infraestructuras e inversiones.

Es un proyecto multidisciplinar ya que cubre aspectos que abarcan desde el software de gestión, *alto nivel*, hasta el diseño y fabricación de dispositivos hardware, *medio y bajo nivel*, (ver Figura 1.1).

La pieza fundamental de este sistema son las tarjetas inteligentes, pequeños dispositivos que permiten al usuario llevar consigo cierta información de forma segura, que son usados como medio de autenticación cuando se requiere la interacción del usuario.

Para permitir la interconexión de todo el sistema se usa una red *TCP/IP*[1] manteniendo así la compatibilidad con Internet, que lo dota de grandes capacidades de administración remota. Para controlar todos estos dispositivos se usa la plataforma JAVA™, gracias a esta y a un microcontrolador avanzado que incorpora soporte para esta, podemos usar un único lenguaje en prácticamente la totalidad del proyecto.

En este capítulo se abordan los objetivos del proyecto y el contexto

[1] *TCP/IP*: Ver definición en página 7

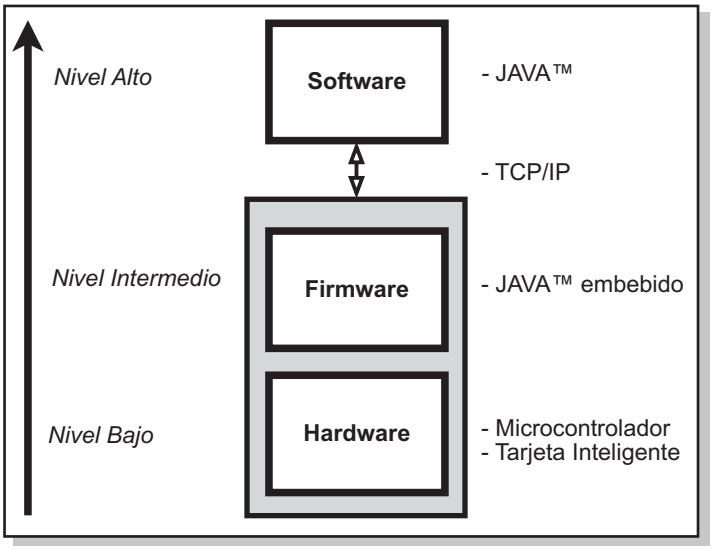


Figura 1.1: Tecnologías usadas en el proyecto y su contexto

por el cual se decide desarrollarlo así como una breve estructura de este documento.

1.1 Motivaciones

Este proyecto surge como una solución para centralizar los accesos a las distintas salas que componen la Facultad de Ciencias, incluyendo despachos, parking, aulas y puertas principales. Esto implica la posibilidad de controlar quién accede a cualquiera de estos recintos y cuándo lo hace.

Por extensión, se tiene este problema en la Universidad de Granada: varios campus repartidos por toda la ciudad, miles de personas entrando y saliendo cada día de sus instalaciones (profesores, alumnos, PAS...).

Esta idea se puede aplicar de forma similar a entidades o particulares que necesiten controlar el acceso a diferentes recintos, lo que conlleva una gran oportunidad de mercado. Algunos ejemplos de aplicación son: hoteles, aparcamientos, hospitales, empresas, edificios de

oficinas, etc. Sin un sistema centralizado por cada habitación se necesitaría una llave diferente lo que resulta muy engorroso. Otro problema es que no existe un control sobre quién y cuándo accede a cada lugar.

¿Por qué no se utilizan las tecnologías que hay en mercado y se incorporan a las cerraduras ya existentes otros dispositivos más seguros y eficaces? El alto coste que supondría instalar esta infraestructura es el principal problema: los productos que hay en el mercado además de caros son difíciles de implantar y requieren un mantenimiento que sólo el fabricante le puede ofrecer.

> ✎ Con SARIC se pretende resolver estos problemas ofreciendo una solución integral de gestión y control de accesos que puede funcionar de forma independiente o ser controlado mediante un ordenador a través de Internet manteniendo facilidad de uso y fiabilidad.

1.2 Historia de los sistemas de control de acceso

Desde los orígenes del tiempo *(informático)* se ha tratado de controlar el acceso a los recursos de información. La tecnología utilizada para este control de accesos ha evolucionado con los propios sistemas de información protegidos. En esta sección se pretende contemplar en una perspectiva histórica el problema de la autorización, contemplando su pasado, y su estado actual que indudablemente está ligado con las nuevas tecnologías de clave pública, tan en auge en este microcosmos de la seguridad.

1.2.1 El problema

La autenticación pretende establecer quién eres. La autorización (o control de accesos) establece qué puedes hacer con el sistema. Estos dos conceptos, que a menudo se mezclan de manera difusa, son completamente independientes. Sin embargo, especialmente cuando se

accede a un recurso de información vía red, sin protección física, existen tres actividades que irán siempre ligadas: La autenticación *(quién soy)*, la autorización *(qué puedo hacer)* y el registro de auditoría *(qué he hecho)*.

En un mundo cada vez más dependiente de las redes es vital no sólo proteger el acceso a los recursos por personas no autorizadas, sino también evitar una manipulación accidental con fatales consecuencias.

1.2.2 La solución conceptual

Existen tradicionalmente dos tipos básicos de controles de acceso con filosofías diametralmente opuestas:

- En el modelo de control de acceso discrecional (DAC - Discretionary Access Control) un usuario bien identificado (típicamente, el creador o 'propietario' del recurso) decide cómo protegerlo estableciendo cómo compartirlo mediante controles de acceso impuestos por el sistema. Este es el modelo habitual en buena parte de los sistemas operativos más habituales. Lo esencial es que el propietario del recurso puede cederlo a un tercero.

  En sus inicios estos sistemas eran excesivamente simples, al permitir un conjunto limitado de operaciones posibles sobre un recurso (rwx por propietario, grupo o resto de usuarios, como en Unix), si bien pronto se añadieron las famosas listas de control de accesos (ACLs - Acess Control Lists), listas de usuarios y grupos con sus permisos específicos. Las ACLs permiten un nivel de granularidad que en ocasiones resulta ser inconveniente ya que complica la administración de la seguridad.

- En el modelo de control de acceso mandatario (MAC - Mandatory Access Control) es el sistema quién protege los recursos. Este se basa en el modelo de clasificación de la información militar, en donde la confidencialidad de la información es lo más relevante, formando lo que se conoce como política de seguridad multinivel.

Los modelos DAC y MAC son inadecuados para cubrir las necesidades de la mayor parte de las organizaciones. El modelo DAC es demasiado débil para controlar el acceso a los recursos de información de forma efectiva, en tanto que el MAC es demasiado rígido.

Desde los 80 se ha propuesto el modelo de control de accesos basado en roles (RBAC - Role Based Access Control), como intento de unificar los modelos clásicos DAC y MAC, consiguiendo un sistema donde el sistema impone el control de accesos, pero sin las restricciones rígidas impuestas por las etiquetas de seguridad.

El modelo RBAC es hoy día muy usado: desde sistemas de base de datos relacionales, pasando por sistemas operativos de red, cortafuegos, productos de seguridad mainframe y entornos abiertos, sistemas de seguridad web...

La evolución de los sistemas de control de accesos ha ido pareja con la de los sistemas de información, han aparecido nuevos mecanismos y dispositivos que ayudan a garantizar la confiabilidad de la información y, poco a poco, se han ido introduciendo en los dispositivos de control de acceso. SARIC incorpora las últimas técnicas de control de acceso unidas a la seguridad que ofrecen las tarjetas inteligentes, partiendo de un sistema RBAC que permite especificar los permisos de forma jerárquica.

1.3 Objetivos del proyecto

Este proyecto Fin de Carrera pretende desarrollar un sistema integral de acceso restringido basado en tarjetas inteligentes de modo que permita unificar el control de tránsito entrada/salida a las diferentes estancias existentes en un edificio civil entendiendo tanto el acceso interior a despachos, oficinas, almacenes, salas de reuniones o laboratorios como exteriores aparcamientos, garajes, cubiertas transitables,

etc.

La figura 1.3 muestra de forma visual los objetivos del proyecto.

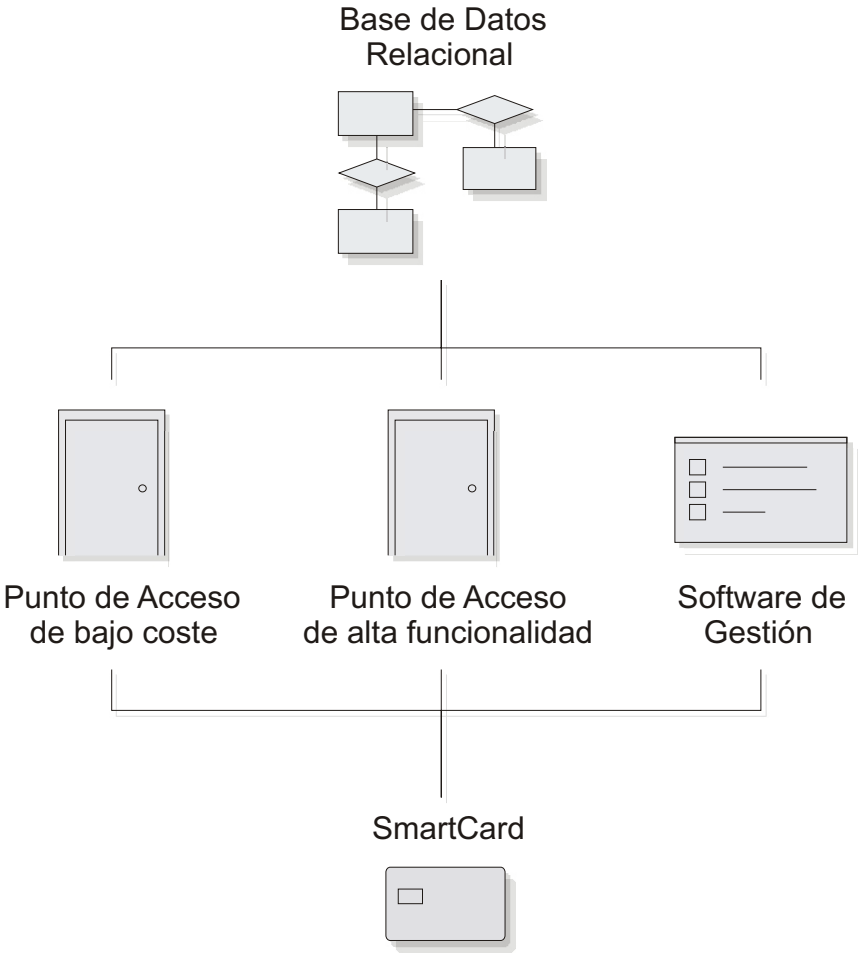


Figura 1.2: Objetivos del proyecto

de la red corporativa y por un gestor técnico que de manera remota podrá realizar tareas de actualización y mantenimiento del firmware del sistema.

El resto de accesos podrán controlarse con dispositivos sin conexión a la red que funcionen de forma autónoma y de coste muy reducido

Intranet	Red de Área Local (Los accesos más importantes en el conjunto inmobiliario dispondrán de conexión a la red, de este modo permitirán ser completamente telecontrolables y configurables por un gestor administrativo a través de la Intranet LAN) privada empresarial o educativa tiene como base el protocolo TCP/IP de Internet

TCP/IP	Familia de protocolos de Internet que sirven para enlazar computadoras que utilizan diferentes sistemas operativos, incluyendo PC, minicomputadoras y computadoras centrales sobre redes de área local (LAN) y área extensa.

capaces de gestionar bitácoras con datos históricos por seguridad. Ya que no disponen de conexión a la red, para consultar estas bitácoras será necesario hacerlo *in situ* mediante un cable serie estandar RS-232.

Del mismo modo, los terminales de control de acceso serán capaces de ejecutar rutinas de autotest para detectar fallos tanto hardware como de seguridad enviando una alerta al administrador de seguridad.

Para el desarrollo del hardware se utilizará una novedosa plataforma: TINI, que hace posible el desarrollo de código tanto de alto como de bajo nivel. El código de alto nivel permitirá realizar comunicaciones vía Ethernet, y el código de bajo nivel se usará en la programación de

RS-232	(también conocido como EIA RS-232C) es un interfaz que designa una norma para el intercambio serie de datos binarios entre un equipo terminal de datos y un equipo de terminación del circuito de datos. El RS-232 consiste en un conector tipo DB-25 de 25 pines, aunque es normal encontrar la versión de 9 pines DB-9, mas barato e incluso mas extendido para cierto tipo de periféricos (como el ratón serie del PC).

Ethernet	Norma o estándar (IEEE 802.3) que determina la forma en que los puestos de la red envían y reciben datos sobre un medio físico compartido que se comporta como un bus lógico, independientemente de su configuración física. Ethernet es popular porque permite un buen equilibrio entre velocidad, costo y facilidad de instalación. Estos puntos fuertes, combinados con la amplia aceptación en el mercado y la habilidad de soportar virtualmente todos los protocolos de red populares, hacen a Ethernet la tecnología ideal para la red de la mayoría de usuarios de la informática actual.

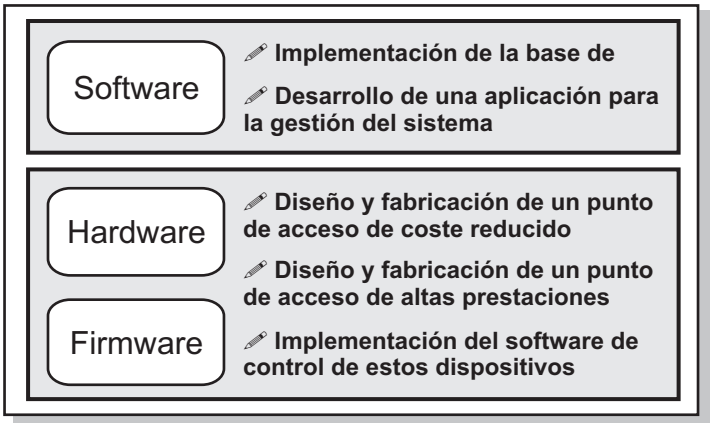


Figura 1.3: Objetivos del proyecto

los dispositivos.

Para el desarrollo del hardware se utilizará una novedosa plataforma: TINI, que incluye capacidades de red Ethernet o vía modem y un microcontrolador cuya arquitectura avanzada ejecuta una versión reducida de la máquina virtual de JAVA™ (JVM), permitiendo desarrollar código tanto a alto nivel para comunicaciones vía Ethernet, como a bajo nivel para la programación de los diferentes dispositivos físicos, *sensores/actuadores*, que controlará: pestillos eléctricos, pantalla, lector de tarjetas inteligentes, teclado, emisores/receptores de infrarrojos, interfaz ehternet y RS-232 entre otros.

En la figura 1.3 se resumen los objetivos del proyecto en cada una de las áreas, a continuación se hace una exposición más detallada:

1. Diseño e Implementación de una base de datos Relacional para gestionar eficientemente los datos de:
   - Accesos
     - Con conexión a la red
     - Sin conexión a la red
   - Grupos de accesos
   - Usuarios

   En lo respectivo a los usuarios, esta deberá ser capaz de almacenar además de los datos personales de cada usuario, una foto y las matrículas de sus coches.

2. Aplicación en JAVA™ para la gestión del sistema. Esta dispondrá de un mecanismo de autentificación para asegurar la correcta gestión de los permisos y realizara las siguientes tareas:
   - Dar de alta nuevas tarjetas y usuarios
   - Modificar/Consultar usuarios existentes
   - Cambiar fecha de caducidad de las tarjetas
   - Imprimir foto y datos personales en la tarjeta
   - Borrar usuarios
   - Añadir, modificar y borrar puntos de acceso
   - Modificar los permisos de los usuarios

3. Diseño y fabricación de un punto de acceso de coste reducido que proporcione los siguientes servicios:
   - Mecanismo de autentificación mediante un lector de tarjetas y un teclado númerico
   - Información del estado de operación del punto de acceso mediante una pantalla digital
   - Sistema de apertura de la puerta

- Bitácora de accesos e incidencias
- Capacidad para ser ampliado mediante distintos tipos de sensores/actuadores

4. Diseño y fabricación de otro punto de acceso sin las limitaciones de bajo coste que permita además de las funciones del anterior, alterar su comportamiento y consultar su estado de forma remota a través de la red.

   El objetivo de los autores durante el desarrollo del proyecto ha sido aplicar los conocimientos adquiridos durante sus estudios en programación, ingeniería del software, bases de datos, criptografía, estructura y arquitectura de los computadores y aprovechar la participación del Departamento de Electrónica y Tecnología de Computadores para ampliar éstos en las áreas donde la formación es menor, como diseño y programación de dispositivos hardware.

1.4 Estructura de la memoria

En este documento se aborda la descripción completa del proyecto, así como el análisis y diseño del software y del hardware que lo componen.

Los manuales de usuario y de desarrollador, las especificaciones de las interfaces, algoritmos y datos técnicos de los dispositivos se adjuntan como apéndices ya que consideramos que su lectura es prescindible para la comprensión del proyecto.

En el siguiente capítulo *(Capitulo 2)* se tratan todas las herramientas y metodologías que se han empleado durante el desarrollo del proyecto, como programas informáticos o el ciclo de vida empleado. Aunque este capítulo no aborda directamente características concernientes al desarrollo del proyecto final, dado que este documento pretende no

solo reflejar que se ha hecho, si no también el cómo se ha hecho, es interesante conocer las técnicas empleadas y las razones del uso de éstas.

El diseño del todo el sistema, su arquitectura y características relativas a la seguridad se abordan en el *capítulo tercero*. Este capítulo ayuda a hacerse una idea de la magnitud del proyecto mostrando los diferentes criterios de diseño y las soluciones que se han adoptado para satisfacer todos los requisitos que definen el proyecto.

Una vez descrito el sistema, en el *capítulo cuarto* se aborda la planificación y desarrollo del proyecto según el ciclo de vida descrito en el capítulo 2. Además se muestra su evolución temporal y la justificación del tiempo empleado en cada una de las fases de desarrollo. Finalmente este capítulo aborda la forma en que se han efectuado las pruebas para garantizar un sistema libre de fallos.

Una parte fundamental en el proyecto es el diseño y fabricación de la infraestructura hardware/firmware, esta se trata en los *capítulos 5 y 6*. En el primero se describen los dispositivos de acceso y los detalles de su fabricación, mientras que en el segundo se aborda el software que les permite realizar sus funciones.

En el último capítulo *(Capítulo 7)* se exponen conclusiones que se han obtenido del desarrollo de este proyecto así como futuras lineas de trabajo que se han considerado interesantes para mejorarlo.

1.5 Documentos que acompañan al proyecto

Con esta memoria se incluye un CD que contiene:

- Código Fuente del Panel de Gestión
- Código Fuente (Firmware) de cada uno de los puntos de acceso
- Datasheets de cada uno de los componentes hardware
- Video explicativo de SARIC realizado en la versión 2

- Drivers y manuales del lector de tarjetas
- Versión electrónica de esta memoria

2 Metodologías, tecnologías y herramientas utilizadas

Antes de comenzar con el desarrollo del proyecto, se ha investigado sobre las diferentes tecnologías que pudiesen emplearse en SARIC, tanto físicas (lectores de tarjetas, microcontroladores, etc.) como diferentes métodos de ingeniería del software.

Entre los objetivos del proyecto está el usar TINI (Tiny InterNet Interfaces). Esta tecnología es una plataforma de desarrollo basada en un microcontrolador que dispone de múltiples puertos de entrada/salida, una pila completa *TCP/IP*[1] y un entorno de ejecución de JAVA™ (JRE) que simplifica el desarrollo.

Ha sido especialmente problemático el encontrar un método eficaz para garantizar el acceso seguro y de coste reducido, que pudiese integrarse fácilmente tanto en dispositivos hardware como en ordenadores personales. Tras estudiar diferentes alternativas, se decidió usar tarjetas inteligentes combinadas con un código de acceso. Los motivos para su elección son:

Seguridad Estas tarjetas permiten almacenar gran cantidad de información de forma totalmente segura, ya que esta puede protegerse mediante técnicas de encriptación.

Facilidad de uso Estas son de uso común en la sociedad, en bancos, la Seguridad Social, y diversas instituciones como la Universidad de Granada.

[1] *TCP/IP*: Ver definición en página 7

JRE	JRE o Java Runtime Environment (Entorno de Ejecución Java) proporciona únicamente un subconjunto del lenguaje de programación Java sólo para ejecución. El usuario final normalmente utiliza JRE en paquetes y añadidos. El JRE es básicamente la máquina virtual de Java y las librerías básicas del J2SE sin las herramientas de desarrollo. Un usuario sólo necesita el JRE para ejecutar las aplicaciones desarrolladas en lenguaje Java, mientras que para desarrollar nuevas aplicaciones en dicho lenguaje es necesario un entorno de desarrollo, denominado JSDK, que además del JRE (mínimo imprescindible) incluye, entre otros, un compilador para Java.

Escalabilidad Al estar basadas en memoria, simplemente con alterar el contenido de ésta puede modificarse su comportamiento, por ejemplo otorgar acceso a diferentes recintos.

Integración Es posible utilizarlas tanto en un dispositivo hardware como en un ordenador, permitiendo así usarlas tanto para identificar al usuario al acceder a un punto de acceso como a la aplicación de gestión.

Bajo coste Otros dispositivos que pueden realizar funciones similares tienen una realación prestaciones/coste más baja.

Otra decisión importante ha sido el lenguaje a utilizar para la implementación tanto del Software de Gestión como del firmware de los microcontroladores. Dado que la plataforma TINI se programa en JAVA™, se ha optado por usar dicho lenguaje siempre que sea posible. Además podemos hacer uso de algunas bibliotecas ya existentes para la programación de contenidos multimedia (Java Media Framework) y acceso a los dispositivos lectores de tarjetas (Open Card Framework).

En cuanto a la metodología de desarrollo, son varios los motivos que justifican el uso de una metodología ágil, entre ellos, la necesidad de ejercitar prácticas iterativas orientadas hacia prestaciones y hacia la entrega, y de comunicación intensiva.

Metodología ágil	Los procesos ágiles de desarrollo de software, conocidos anteriormente como metodologías livianas, intentan evitar los tortuosos y burocráticos caminos de las metodologías tradicionales enfocándose en la gente y los resultados.

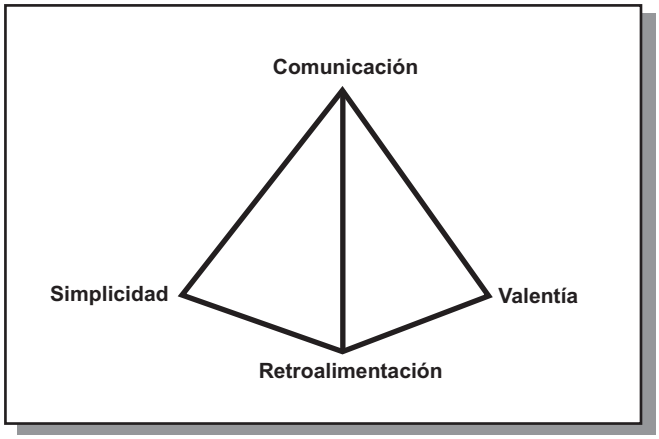


Figura 2.1: Pilares básicos de la programación extrema

En este casó se ha optado por la más famosa de todas ellas: la Programación Extrema (a la que dedicamos una sección completa en la pagina 18). Como se muestra en la Figura 2.1 se basa en estas cuatro premisas:

Comunicación: Es necesaria una comunicación permanente tanto entre los miembros del equipo como con el cliente.

Simplicidad: El problema se trata como una serie de subproblemas más sencillos. El sistema se define mediante metáforas que abstraen sus aspectos menos relevantes. Se debe diseñar la solución más simple que puede funcionar y ser implementada en un momento determinado del proyecto.

Retroalimentación: Continuamente se obtienen nuevas revisiones del software. Gran parte del éxito del proyecto se debe a que es el cliente quien conduce constantemente el trabajo hacia lo que aportará mayor valor de negocio y los programadores pueden resolver de manera inmediata cualquier duda asociada.

Valentía: No se debe dudar a la hora de realizar cambios importantes que impliquen rediseñar la aplicación.

2.1 Metodología de desarrollo

En SARIC convergen diversas circunstancias que favorecen la elección de una metodología de diseño ágil:

- El sistema puede dividirse en subsistemas pero estos son complejos y difícilmente planificables a priori, ya que gran parte de las prestaciones solicitadas requieren investigación y estudio por parte de los autores (acceso a dispositivos de bajo nivel, utilización de bibliotecas novedosas, fabricación del hardware, etc.).

- Para clarificar los objetivos es necesaria una realimentación permanente, una continua comunicación con el tutor que ha de evaluar cada funcionalidad solicitada, y esto solo puede conseguirse utilizando una *metodología ágil*[2] que permita obtener rápidamente versiones simplificadas del programa.

- Para obtener un mejor rendimiento en muchas ocasiones los autores tendrán que colaborar el uno con el otro para abordar ciertas tareas. En un proyecto de este tipo se requiere el dialogo y la comunicación entre los desarrolladores que en ocasiones tendrán que trabajar en el mismo terminal codo con codo.

Estas características se contradicen con las de metodologías pesadas como RUP o métrica3, pensadas para equipos de desarrollo grandes y jerarquizados como las que se estudiaron en las asignaturas de Ingeniería del Software. Los modelos clásicos ortodoxos exaltan casi siempre las virtudes del planeamiento y poseen un espíritu normativo: Comienzan con el análisis completo de los requerimientos del usuario. Después de un largo período de intensa interacción con usuarios y clientes, los ingenieros establecen un conjunto definitivo y exhaustivo

[2] *metodología ágil*: Ver definición en página 16

de rasgos, requerimientos funcionales y no funcionales. Esta información se documenta en forma de especificaciones para la segunda etapa, el diseño, en el que los arquitectos, trabajando junto a otros expertos en temas puntuales (como son estructuras y bases de datos), generan la arquitectura del sistema. Luego los programadores implementan ese diseño bien documentado.

Las metodologías ágiles son, por el contrario, estrategias de desarrollo de software que promueven prácticas que son adaptativas en vez de predictivas, centradas en la gente o en los equipos, iterativas, orientadas hacia prestaciones y hacia la entrega, de comunicación intensiva, y que requieren que el negocio se involucre en forma directa.

Están muy alineadas tanto en los principios como en las prácticas para el desarrollo de software en ambientes que requieren un alto grado de adaptabilidad y es por ello que los autores han elegido la mejor documentada (y más conocida) de todas ellas: La Programación Extrema, (Extreme Programming).

2.1.1 ¿Qué es la Programación Extrema?

La Programación Extrema o eXtreme Programming (XP) es una aproximación a la ingeniería de software formulada por Kent Beck, autor del primer libro sobre la materia, *Extreme Programming Explained: Embrace Change*. Se trata de un proceso ágil de desarrollo de software.

Las fundamentos del método son (Ver figura 2.2):

- Desarrollo iterativo e incremental: pequeñas mejoras, unas tras otras.
- Pruebas unitarias continuas, frecuentemente repetidas y automáticas, incluyendo pruebas de regresión. Se aconseja escribir el código de la prueba antes de la codificación.
- Programación por parejas: se recomienda que las tareas de desarrollo se lleven a cabo por dos personas en un mismo puesto.

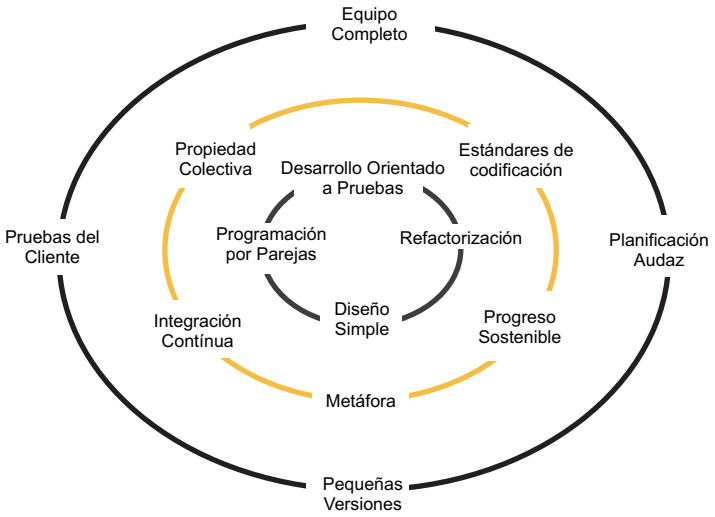


Figura 2.2: Fundamentos de la programación extrema

Se supone que la mayor calidad del código escrito de esta manera (el código es revisado y discutido mientras se escribe) es más importante que la posible pérdida de productividad inmediata.

- Frecuente interacción del equipo de programación con el cliente o usuario. Se recomienda que un representante del cliente trabaje junto al equipo de desarrollo.

- Corrección de todos los errores antes de añadir nueva funcionalidad. Hacer entregas frecuentes.

- Refactorización del código, es decir, reescribir ciertas partes del código para aumentar su legibilidad y mantenibilidad pero sin modificar su comportamiento.

- Las pruebas han de garantizar que en la refactorización no se ha introducido ningún fallo.

- Propiedad del código compartida: en vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, este método promueve el que todo el personal pueda corregir y extender cualquier parte del proyecto. Las frecuentes pruebas de regresión garantizan que los posibles errores serán detectados.

- Simplicidad en el código: es la mejor manera de que las cosas funcionen. Cuando todo funcione se podrá mejorar si es necesario.

2.2 Herramientas de Ayuda al diseño

Para la realización de este proyecto se han utilizado herramientas CASE y herramientas adicionales de apoyo al análisis y al diseño. A continuación se clasifica y comenta cada una de ellas:

Editores de Diagramas: Para crear diagramas de flujo de datos, jerarquías de objetos, diagramas entidad-relación, etcétera. Estos editores no son solo herramientas de dibujo puesto que conocen los tipos de entidades.

> **Dia**: Dia es una aplicación de creación de diagramas basada en gtk+ y desarrollada bajo licencia GPL. Está diseñada para ser muy similar al programa comercial de Ms Windows "Visio". Actualmente posee diversos objetos para dibujar diagramas Entidad-Relación, UML, de flujo y de red. Puede cargar y almacenar los trabajos a diferentes formatos XML, y exportarlos como imágenes vectoriales. http://www.gnome.org/projects/dia/

Recursos de importación y exportación: Permiten el intercambio de información del repositorio central con otras herramientas de desarrollo.

> **Subversion**: Subversion es un sistema de control de versiones libre (código fuente abierto). El repositorio es como un servidor de ficheros ordinario que además recuerda todos los cambios hechos a sus archivos y directorios. Esto le permite recuperar versiones antiguas de sus datos, o examinar la historia de cómo han cambiado. http://svnbook.red-bean.com/

Herramientas CASE: CASE es el acrónimo inglés de Computer Aided Software Engineering, que significa Ingeniería de Software Asistida por Ordenador.

Son instrumentos o sistemas automatizados que brindan soporte a las actividades de producción de software. Se pueden clasificar en:

CASE de alto nivel: herramientas que soportan las actividades (de alto nivel) de análisis de requerimientos y diseño. Por ejemplo, las herramientas de modelado diagrama de entidad relación y diagramas UML.

CASE cruzado: herramientas que apoyan actividades que tienen lugar a lo largo de todo el ciclo de vida. Se incluyen actividades como la gestión de proyectos y la estimación.

CASE de bajo nivel: herramientas que soportan las actividades (de bajo nivel) de programación, la depuración de programas y las pruebas.

Además de incluir "Editores de Diagramas" y "Recursos de importación/exportación", una herramienta CASE contiene "Herramientas de Diseño, Análisis y Verificación", "Depósito de lenguajes de consulta", "Diccionario de Datos", "Herramientas de definición y generación de informes", "Herramientas de definición de formularios" y "Generadores de Código". Para diseñar SARIC se ha utilizado una herramienta CASE de alto nivel: Umbrello.

Umbrello: Umbrello es aplicación de creación de diagramas UML de código abierto disponible nativamente para Unix. Es parte de KDE pero funciona bien en otros escritorios y entornos de programación. Maneja todos los tipos estándar de diagramas UML, puede importar clases C++ y exportar a diez lenguajes de programación diferentes (Java entre ellos). http://uml.sourceforge.net/index.php

Herramientas de Temporización: Sistemas de apoyo a la planificación que ofrecen herramientas para distribuir trabajo y recursos de forma adecuada a lo largo del ciclo de vida del proyecto.

> **Planner**: Planner es una herramienta de GNOME para planificar, programar y realizar el seguimiento de proyectos. Entre sus características destacan la definición de tareas, recursos y dependencias entre tareas, la posibilidad de definir calendarios laborales, la exportación de los proyectos en formato HTML, etc. http://www.imendio.com/projects/planner/

Herramientas de apoyo a la Metodología XP: En esta categoría se incluyen aquellas herramientas comunican de forma eficiente a los componentes del equipo entre ellos y con el tutor, y aquellas que permiten la colaboración directa entre los desarrolladores editando el mismo código fuente simultáneamente.

> **Gobby**: Gobby es un editor colaborativo que soporta múltiples documentos en la misma sesión y chat multiusuario. Puede ser ejecutado bajo Ms Windows, Mac OS X, GNU/Linux y otras plataforma de tipo Unix. Usa GTK+ como interfaz de ventana y se integra con el escritorio Gnome. http://gobby.0x539.de

Herramientas de Diseño Asistido por Ordenador (CAD) : Estas son herramientas que permiten crear todo tipo de diseños de ingeniería haciendo uso de un ordenador. En nuestro caso nos permiten diseñar dispositivos hardware tanto a nivel lógico como físico. Su finalidad es optimizar su desarrollo y consecuentes costos de fabricación y reducir al máximo las pruebas para la obtención del producto deseado.

P-CAD 2004: P-CAD 2004 es una herramienta integral de diseño de PCB que cubre el diseño lógico, el diseño de las placas de circuito impreso así como la realización de pruebas para asegurar que el diseño es correcto. www.pcad.com/

PADS: PADS es una herramienta de Mentor Graphics similar a la anterior usada en entornos profesionales. Incluye soporte para todo tipo de tests de análisis y verificación mas avanzados que P-CAD. Caben destacar sus características de enrutado interactivo y automático, simulación Analógica, HDL y Mixta y pruebas de integridad de la señal. http://www.mentor.com/products/pcb/pads/

2.3 Herramientas Hardware

Durante el proyecto se ha contado con varias herramientas hardware para facilitar el desarrollo de este:

- Un servidor dedicado, usado para mantener el repositorio de todos los documentos y ficheros del proyecto y para ejecutar las aplicaciones servidor necesarias por el mismo. Este dispone de un procesador AMD a 2600Mhz, 256MB de RAM y 40GB de disco duro y conexión a
- InternetInternet.
- Cada uno de los autores del proyecto cuenta con un portátil Acer con procesador Intel Centrino, 512MB de RAM y 60GB de disco duro equipados ambos con un lector/grabador de tarjetas LTC31 USB.
- Además se dispone de acceso a un laboratorio ubicado en la Facultad de Ciencias que cuenta con todos los materiales necesarios para la fabricación de placas de circuito impreso (PCB de ahora

PCB	Una placa de circuito impreso (PCB - del ingles 'Printed Circuit Board') es una placa de fibra de vidrio o compuesto similar al que están adheridas una o más capas de cobre, cada una separada por otra capa de fibra a modo de sándwich. De esta forma y mediante un proceso químico es posible imprimir una película con las distintas pistas y conexiones eléctricas entre los componentes que se montarán en el circuito impreso, dejando sólo el cobre útil y eliminando por ácido el cobre indeseado.

en adelante). A continuación se describen los mas relevantes (Ver figura 2.3):

Insoladora: Una insoladora es un dispositivo que contiene uno o varios tubos fluorescentes que sirve para marcar el dibujo del circuito en la PCB fotosensible para que este no sea disuelto por el ácido en un proceso posterior. La usada en este proyecto cuenta con un temporizador para controlar el tiempo de exposición, una bomba de vacío para asegurar la correcta alineación del fotolito y capacidad para placas de doble cara.

Osciloscopio: Un osciloscopio es un instrumento de medida electrónico para la representación gráfica de señales eléctricas que pueden variar en el tiempo.

Multímetro digital Un multímetro, a veces también denominado polímetro o tester, es un instrumento electrónico de medida que combina varias funciones en una sola unidad. Las más comunes son las de voltímetro, amperímetro y óhmetro que miden tensiones, intensidades y resistencias respectivamente.

Soldador: El soldador es un aparato con un mango aislante térmico, alineado con una resistencia eléctrica y una punta formada por varias capas metálicas.

Taladro: Un taladro con una broca muy pequeña para hacer in-

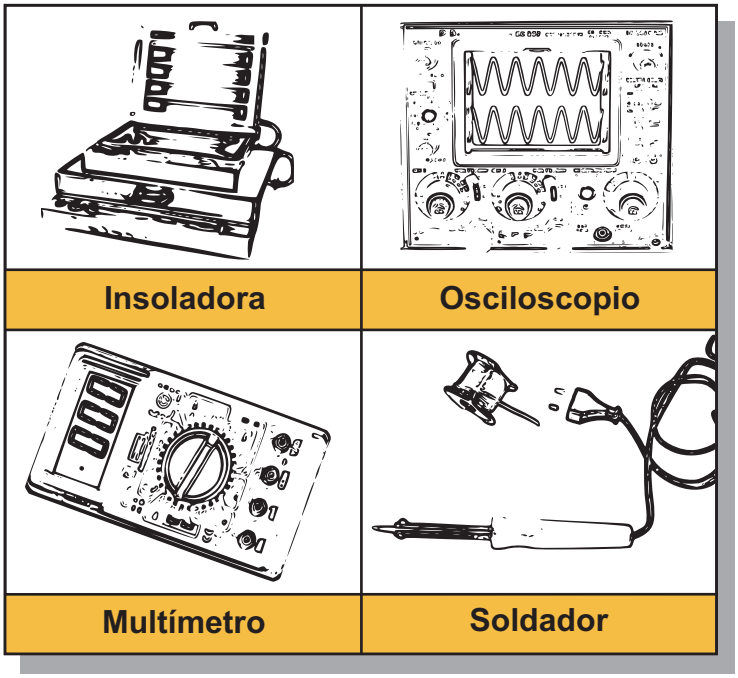


Figura 2.3: Imágenes de una insoladora, un osciloscopio, un multímetro y un soldador.

cisiones en las PCBs y permitir insertar posteriormente los componentes.

Barreños y productos de atacado químico Para el revelado y el posterior atacado de la placa son necesarios varios productos que mezclados en la proporción adecuada en un barreño eliminan la capa de revelado y el cobre no fotosensibilizado.

2.4 Tarjetas inteligentes

Fotolito	Cliché fotográfico obtenido mediante un programa informático de diseño electrónico como OrCAD que, una vez impreso sobre un soporte transparente y superpuesto en una de las capas de la PCB, permite marcar las zonas que no se desea que sean disueltas por el ácido.

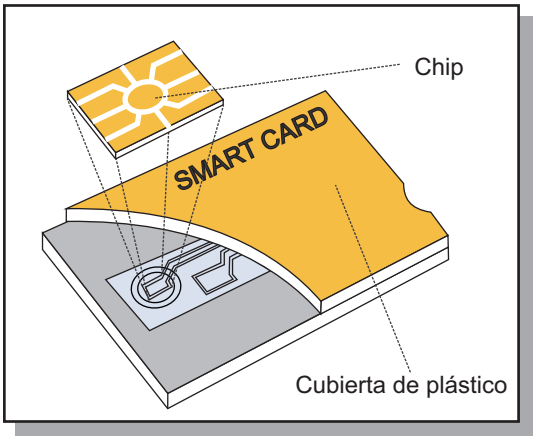


Figura 2.4: Esquema de una tarjeta inteligente

Las tarjetas inteligentes o SmartCards son pequeños dispositivos electrónicos de las dimensiones de una tarjeta de crédito que contienen una memoria electrónica y posiblemente in circuito integrado (IC). Las tarjetas inteligentes que contienen un circuito integrado muchas veces son llamadas Integrated Circuit Cards (ICCs). En la figura 2.4 se muestra el esquema de una tarjeta inteligente.

A pesar de su reducido tamaño estas tarjetas permiten almacenar información, encriptarla y así evitar su lectura por parte de personas no autorizadas. Otra característica destacable es que para poder utilizarse es necesaria una clave de acceso o PIN, incluso es posible incorporarle tecnología mas avanzada como identificación por técnica biométrica, huella digital o lectura de retina. Esto las hace ideales para almacenamiento y procesamiento de datos confidenciales.

Estos pequeños dispositivos están en auge en la actualidad, por el alto rendimiento que ofrecen tanto en fiabilidad como seguridad y su uso esta muy extendido en varias aplicaciones:

- Estado de las cuentas de crédito
- Historiales médicos
- Números de identificación personal
- Claves privadas (controles de acceso)
- Dinero electrónico

2.4.1 Funcionamiento

Las tarjetas se activan al introducirlas en un lector de tarjetas. Un contacto metálico (en las usadas en este proyecto), permite la transferencia de información entre el lector y la tarjeta. El estándar ISO 7816/3 define los comandos que admiten estas tarjetas; los más usados son: lectura y escritura para cada una de las zonas de memoria y cambio de PIN. Algunas tarjetas más avanzadas cuentan con comandos propios para encriptado de información y otras operaciones.

2.4.2 Evolución Histórica

Tarjetas de banda magnética Estas cuenta con una banda magnética en la parte posterior con capacidad para almacenar información, y a pesar de los muchos inconvenientes que presentan como su poca capacidad, fácil y que pueden almacenar información su uso está muy extendido y tiene una buena aceptación en el mercado.

Tarjetas de memoria Tarjetas con un chip integrado capaces de almacenar la información de forma segura. Son las primeras tarjetas "inteligentes" y están definidas en el estándar ISO 7816. Sus inconvenientes es que no son del todo seguras, solo son capaces de almacenar información, no son versátiles y presentan dificultad para la multiaplicación.

Tarjetas inteligentes con microprocesador Tarjetas con un microprocesador capaces de almacenar y procesar información de forma segura. Son las consideradas actualmente como tarjetas inteligentes y están definidas en las normas ISO 7816 / ISO 14443. Sus ventajas son su mayor seguridad frente a las anteriores, que son capaces de procesar información y su mayor versatilidad y capacidad de memoria. Sin embargo todo esto tiene un coste tanto económico como de esfuerzo a la hora de integrarlas en una aplicación.

2.4.3 ¿Por qué son seguras las tarjetas inteligentes?

Seguridad en los componentes El chip es a prueba de falsificación y no puede ser duplicado, además pueden contar con medios de protección frente a ataques por rayos X o ultravioleta, voltajes inusuales o cambios en la frecuencia de reloj.

Seguridad en el sistema operativo El SO cuenta con control de acceso a memoria y protección para los datos y ficheros.

Seguridad en las transacciones Los datos solo están disponibles tras introducir el código de acceso o PIN y puede trabajar con información encriptada con diferentes sistemas, clave pública (RSA), clave privada (DES) o firma digital (MD5)

2.4.4 Arquitectura básica de una tarjeta con microprocesador

La arquitectura de estas tarjetas es similar a la de un pequeño ordenador, cuenta con una CPU de 8 bits a 5Mhz y adicionalmente con un procesador dedicado para criptografía (criptoprocesor). Una memoria ROM cuyo equivalente en un PC seria la BIOS, donde se almacena el sistema operativo, algoritmos de encriptación y otras funciones útiles. También dispone de una pequeña memoria RAM que puede ser de 256 o 512 bytes y una memoria EEPROM con los datos del usuario que varia entre los 2K y los 64K de las más modernas. En la figura 2.5 se muestra un esquema de la arquitectura de estas tarjetas.

2.4.5 Ventajas

- Gran capacidad de memoria (frente a bandas magnéticas u otros medios para transportar información).
- Altos niveles de seguridad
- Reducción del fraude
- Confiabilidad

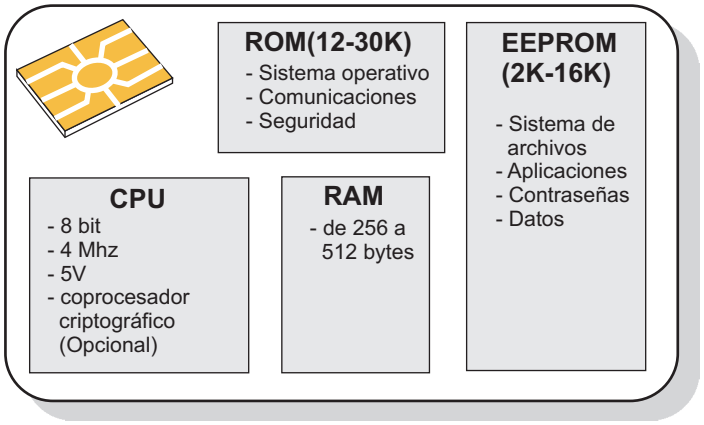


Figura 2.5: Arquitectura de una tarjeta inteligente basada en microprocesador

- Seguridad en la información
- Facilidad de uso sin necesidad de conexiones en línea o vía telefónica
- Comodidad para el usuario
- Caída de los costos para empresarios y usuarios
- Estándares específicos ISO
- Privacidad.

2.4.6 Tipos de tarjetas

En el mercado hay disponibles varios tipos de tarjetas:

Tarjeta Inteligente de Contacto Estas tarjetas son las que necesitan ser insertadas en una terminal con lector inteligente para que por medio de contactos pueda ser leída. Definidas en el estándar ISO 7816. Existen dos tipos de tarjeta inteligente de contacto: las síncronas y las asíncronas.

- *Tarjetas Inteligentes Síncronas:* Son tarjetas con sólo memoria y la presentación de esta tarjeta inteligente y su utilización se concentra principalmente en tarjetas prepagadas para hacer

llamadas telefónicas. Estas tarjetas contienen un chip de memoria que se utiliza generalmente para el almacenamiento de datos, dentro de esta categoría existen dos tipos de tarjeta:

- Memoria Libre: Carece de mecanismos de protección para acceder a la información.
- Memoria Protegida: que necesita de códigos y pasos previos para tener acceso a la información.

- *Tarjetas Asíncronas:* Son tarjetas inteligentes con microprocesador, ésta es la verdadera tarjeta inteligente, tiene el mismo tamaño y grosor de una tarjeta de crédito. Dentro del plástico se encuentra un elemento electrónico junto con la memoria RAM, ROM y EEPROM en el mismo chip

Tarjetas Inteligentes sin Contacto Son similares a las de contacto con respecto a lo que pueden hacer y a sus funciones pero utilizan diferentes protocolos de transmisión en capa lógica y física, no usan contacto galvánico sino de interfaz inductiva por medio de una antena incluida en la tarjeta que puede ser de media distancia sin necesidad de ser introducida en una terminal de lector inteligente. Una de las ventajas que esta tarjeta tiene es que como no existen contactos externos con la tarjeta, ésta es más resistente a los elementos externos y las transacciones se realizan más rápidamente.

Su definición se encuentra en el estándar ISO 14443

Tarjetas Mixtas Son una combinación de las anteriores.

2.4.7 La norma ISO 7816

La norma ISO 7816-3 Es el estándar de definición de las tarjetas inteligentes de microprocesador con contactos. Consta de cuatro partes/documentos básicos:

ISO 7816-1 Características físicas

- Tarjeta conforme con ISO 7810, 7813

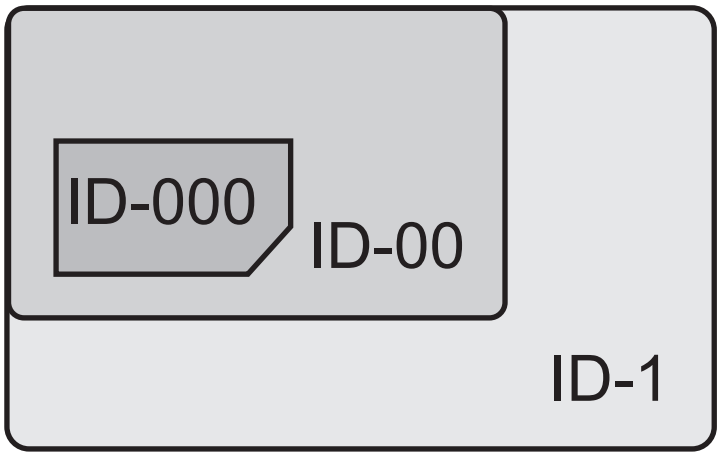


Figura 2.6: Formatos definidos en los estándares ISO 7810, 7813

- La tarjeta debe:
  - Resistir ataques con rayos X y luz Ultravioleta
  - Tener superficie plana
  - Permitir cierto grado de torsión
  - Resistir altos voltajes, campos electromagnéticos, electricidad estática
  - No disipar más de 2,5 W
- Tamaño de la tarjeta. Hay disponibles diferentes tamaños (Ver figura 2.6:
  - ID-1 (es el más habitual): El de las tarjetas de crédito.
  - ID-00: Un tamaño intermedio poco usado.
  - ID-000: Es el usado por la telefonía GSM.

ISO 7816-2 Dimensión y localización de los contactos: Existen 8 contactos pero solo se usan 6 y son los siguientes:

- VCC - Alimentación
- VPP - Alimentación
- GND - Masa
- RST - Reset

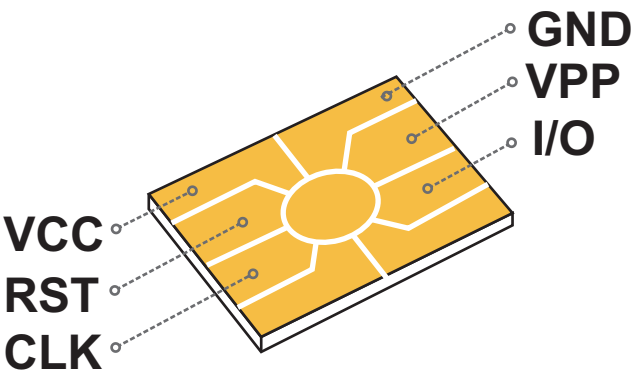


Figura 2.7: Distribución de los contactos en una tarjeta inteligente

- CLK - Señal de reloj
- I/O - Entrada/Salida

ISO 7816-3 Señales electrónicas y protocolos de transmisión: Este es el más importante en el ámbito del proyecto, puesto que se describen todas las señales electrónicas y protocolos de transmisión que luego tendrán que ser implementadas, tanto en el software como en el firmware. Son los siguientes:

- Procedimiento de operación (activación y reset)
- ATR (Answer To Reset) Devuelve un número de identificación para la tarjeta.
- Protocolos de comunicación: Se definen 2 tipos
  - Protocolo T=0 (protocolo orientado a carácter). Es el más utilizado por su uso en telefonia móvil (GSM), para recibir datos se requieren dos transacciones.
  - Protocolo T=1 (protocolo orientado a bloque). Se requiere una única transacción para recibir datos.

2.4.8 Las tarjetas inteligentes de SARIC

Para el desarrollo del proyecto se cuenta con tarjetas inteligentes síncronas con memoria protegida, concretamente son el modelo C3P2K. Cuenta con 256 bytes de memoria tipo EEPROM de los cuales los 32

primeros pueden ser protegidos escribiendo en la memoria de protección funcional de 32 bits. El tiempo de lectura/escritura es de 2.5ms. Además cuenta con 4 bytes adicionales de memoria de seguridad usados para almacenar el pin y el número de intentos restantes antes del bloqueo total de la tarjeta.

Características técnicas

- 256 x 8 bits de organización de EEPROM
- Direccionamiento de byte inteligente
- Protección irreversible contra escritura de los primeros 32 bits
- 32 x 1 bits de organización de la memoria de protección.
- ATR según la norma ISO 7816-3
- Tiempo de propagación de 2.5 ms por byte
- Mínimo de 10.000 ciclos de escritura/borrado
- Retención de datos por un mínimo de 10 años
- Los datos solo pueden ser accedidos tras introducir el código de seguridad de 3 dígitos programable

2.4.9 Lectores de Tarjetas

Los lectores de tarjetas son dispositivos que permiten acceder y modificar la información contenida en las tarjetas inteligentes. Hay muchísimos tipos pero podemos clasificarlos en dos:

Dispositivos para integración Simplemente son dispositivos adaptadores preparados para su integración en una *PCBs*[3] que comunican los contactos de la tarjeta con las pistas de la PCB. Estos pueden ser mas sofisticados e incluir medios mecánicos para retener la tarjeta y expulsarla automáticamente cuando finalice la operación.

[3] *PCB*: Ver definición en página 24

Dispositivos para PCs Estos suelen usar un puerto de serie (*RS-232*[4] o USB) para comunicarse con el ordenador y también los encontramos en diferentes formatos, externos, en bahías para integrarlos como un dispositivo de almacenamiento más en la propia torre o incluso en el teclado del ordenador.

La interacción con los dispositivos para integración ha de hacerse siguiendo la norma ISO 7813-3 descrita en el apartado anterior trabajando a nivel de señales eléctricas, sin embargo trabajar con dispositivos para PCs es algo más complicado porque hay que interaccionar con el driver del dispositivo. Para solucionar estos problemas de interoperatividad varias empresas fundan el grupo de trabajo PC/SC (Personal Computer/Smart Card).

2.4.10 El estandart PC/SC, M.U.S.C.L.E. y OCF

En mayo del 96, las empresas Groupe Bull, Hewlett-Packard, Microsoft, Schlumberger y Siemens Nixdorf fundal el grupo de trabajo PC/SC.

Podemos resumir los objetivos de PC/SC en los cuatro siguientes:

- Facilitar la integración y uso de las tarjetas inteligentes en entornos PC.
- Interoperatividad a diferentes niveles entre tarjetas y lectores.
- Neutral respecto a aplicaciones y fabricantes
- Especificaciones independientes de la plataforma

Las especificaciones PC/SC se basan en los estándares ISO 7816 y son compatibles con las especificaciones EMV y GSM. Existe amplia compatibilidad con el sector para estas especificaciones, así como un fuerte deseo de convertirlas en estándares independientes en el futuro. Desde la fundación y publicación inicial de las especificaciones, se han unido nuevos miembros al grupo de trabajo PC/SC. Entre ellos se incluyen Gemplus, IBM, Sun Microsystems, Toshiba y Verifone.

[4] *RS-232*: Ver definición en página 8

En la revisión 1.0 PC/SC soporta:

- Tarjetas asíncronas que cumplan el estándar ISO 7816
- Lectores de tarjetas “sencillos”
- Implementado en Win9X, Win NT 4.0 y Windows 2000
- Implementado en Linux (M.U.S.C.L.E)

En agosto de 2004 se publica la versión 2.0 de este estándar.

A pesar de la creación de este estándar hacer aplicaciones compatibles entre varios sistemas operativos, varias tarjetas y varios lectores sigue siendo una tarea ardua y compleja, para paliar estos problemas surge una biblioteca, o más bien un conjunto de estas (Framework) para trabajar con tarjetas inteligentes.

OCF: Open Card Framework

El OCF ha sido diseñado para hacer aplicaciones independientes de:

- Lectores de tarjetas inteligentes
- Las propias tarjetas
- La plataforma puesto que esta basado en JAVA

Es totalmente escalable y flexible y fácil de usar puesto que proporciona una capa de abstracción de las operaciones a bajo nivel. Sus primeros resultados se publican en Mayo del 98 cuando aparece la versión 1.0, y el último estándar OCF 1.2 en Enero del 2000.

Como se verá mas adelante, el uso de OCF justifica por si solo la elección de JAVA como plataforma de desarrollo en la mayor parte del proyecto.

3 Diseño del sistema

Diseñar un sistema es definir la arquitectura de hardware y software, componentes, módulos y datos de un Sistema para satisfacer ciertos requerimientos. En este capitulo se pretende describir de forma detallada el funcionamiento de todas las partes de SARIC y como se relacionan e interactúan entre sí.

El diseño y el análisis orientado a objetos es el método más ampliamente utilizado para el diseño de sistemas software y el proceso de modelado unificado (UML) UML es ahora un estándar llegando a usarse incluso en áreas diferentes al software.

3.1 Identificación de los objetivos del diseño

Los objetivos de diseño son las cualidades deseables del sistema, que van a determinar el diseño del mismo. Se obtienen a partir de los requisitos no funcionales o del dominio de la aplicación. En este proyecto, partiendo de los requisitos no funcionales se han identificado los siguientes:

- La interfaz del sistema, se diseñará con NetBeans 5.0 con una estructura lo más amigable e intuitiva posible, permitiendo que el usuario pueda manejar las aplicaciones sin necesidad de conocimientos informáticos mas allá de los que posee un usuario de escritorio.

UML	Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, Unified Modeling Language) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; aún cuando todavía no es un estándar oficial, está apoyado en gran manera por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables. El punto importante para notar aquí es que UML es un "lenguaje"para especificar y no un método o un proceso. UML se usa para definir un sistema de software; para detallar los artefactos en el sistema; para documentar y construir -es el lenguaje en el que está descrito el modelo. UML se puede usar en una gran variedad de formas para soportar una metodología de desarrollo de software (tal como el Proceso Unificado de Rational) -pero no especifica en sí mismo qué metodología o proceso usar.

- El usuario podrá acceder a un manual de la aplicación que le permitirá conocer y comprender completamente la funcionalidad del sistema, sirviéndole de orientación y ayuda.
- El sistema se implementará en Java y se captarán las excepciones y errores que se produzcan durante la utilización y procesamiento de la información, y se mostrarán mensajes de error lo más claros posible indicando el problema o la operación incorrecta realizada sin mostrar información que pueda confundir al usuario.
- El sistema gestionará la autentificación con SmartCards (o tarjetas inteligentes) y contraseñas con el objetivo de mantener la seguridad en el sistema. Dependiendo del tipo de usuario del sistema, éste tendrá acceso a unas determinadas vistas y funcionalidades.

SGBD	Los Sistemas Gestores de Bases de Datos son un tipo de software muy específico, dedicado a servir de interfaz entre la Base de datos y el usuario, las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. En los textos que tratan este tema, o temas relacionados, se mencionan los términos SGBD y DBMS, siendo ambos equivalentes, y acrónimos, respectivamente, de Sistema Gestor de Bases de Datos y DataBase Management System, su expresión inglesa.

Las contraseñas se almacenaran encriptadas para preservar los datos privados de cada usuario.

- El diseño del sistema permitirá la posibilidad de acceso concurrente al sistema por varios usuarios mediante
- InternetInternet o una LAN para el transporte de datos entre la aplicación de cada usuario y el SGBD.
- El dispositivo de almacenamiento de datos (servidor de bases de datos MySQL) se diseñará sin tener en cuenta el volumen de datos, aunque se presupone que el proyecto no es de gran envergadura. Es por esta razón por la que se ha optado por MySQL en detrimento de un servidor más potente como Oracle o SQLServer, ya que además permite disminuir considerablemente el coste del software al no ser necesario adquirir licencias para el SGBD ni para el Sistema operativo.
- El sistema dispondrá en todo momento de una configuración inicial por defecto que le permitirá recuperarse de los errores software que se puedan producir. Dicha configuración será la inicial con la que partirá el sistema tras la implantación.
- El sistema será portable tanto a sistemas operativos Windows como Linux, ya que la herramienta que se utilizará para desa-

MySQL	MySQL es un sistema de gestión de base de datos, multihilo y multiusuario con más de seis millones de instalaciones. MySQL AB desarrolla MySQL como software libre en un esquema de licenciamiento dual. Por un lado lo ofrece bajo la GNU GPL, pero, empresas que quieran incorporarlo en productos propietarios puede comprar a la empresa una licencia más permisiva que les permita ese uso.

rrollarlo, Java, se encuentra disponible en ambos, con el único requisito de tener instalada la Máquina Virtual Java (de libre distribución tanto para sistemas Linux como Windows) y configurar algunas bibliotecas.

- En lo que se refiere a la estructura del sistema, contará en principio con un solo tipo de cliente (entiéndase como cliente el concepto de la arquitectura cliente/servidor). Este cliente será capaz de adaptarse a cada tipo de usuario ocultando las opciones no disponibles para según que casos.
- Los equipos necesarios para que el software funcione correctamente se reducen a un ordenador que hará las funciones de servidor, el sistema operativo con la máquina virtual de java instalada, un dispositivo para leer las tarjetas y el servidor de bases de datos MySQL. Luego cada usuario deberá disponer de un PC con conexión a
- InternetInternet o a la red local.
- Sera posible agrupar diferentes puntos de acceso en “grupos de acceso”.
- El sistema será totalmente escalable sin necesidad de hacer cambios en los dispositivos que ya forman parte de él cuando se decida ampliarlo.
- Los dispositivos de autentificación hardware proporcionaran un lector de tarjetas y un método para introducir la contraseña, además

incluirán una pantalla digital para visualizar su estado de operación y proveer al usuario con mensajes que faciliten su uso.

- Se podrá conectar a los dispositivos cualquier mecanismo electrónico de apertura que requiera de una señal electrónica de activación.
- Los dispositivos almacenarán un pequeño registro de accesos e incidencias.
- Será posible incluir ampliaciones en los dispositivos sin necesidad de volver a rediseñar los mismos.
- En uno de los dispositivos se intentará mantener el coste de fabricación lo más bajo posible pero sin que este pierda funcionalidad. El otro dispositivo sin limitación de costes será dotado de capacidades de administración remota mediante una Red de Comunicaciones.
- Se incluirán métodos que permitan detectar la corrupción de los datos en las tarjetas por interferencias electromagnéticos y por un mal uso intencionado.
- Los datos incluidos en las tarjetas se protegerán con algún método de encriptación de forma que sea posible preservar la privacidad de los mismos.

3.2 Arquitectura Software

3.2.1 Características de la arquitectura seleccionada

Se ha decidido usar una arquitectura cliente-servidor; esta arquitectura goza de gran popularidad en el mercado del software actual y permite dividir el sistema en diferentes capas que pueden tratarse por separado. A continuación se describe brevemente en qué consisten este tipo de arquitecturas.

La arquitectura cliente/servidor es un modelo para el desarrollo de sistemas de información, en el que las transacciones se dividen en procesos independientes que cooperan entre sí para intercambiar información, servicios o recursos. Se denomina cliente al proceso que inicia el diálogo o solicita los recursos y servidor, al proceso que responde a las solicitudes.

En este modelo, las aplicaciones se dividen de forma que el servidor contiene la parte que debe ser compartida por varios usuarios, y en el cliente permanece sólo lo particular de cada usuario.

Los clientes interactuarán con el usuario, usualmente en forma gráfica. Frecuentemente se comunican con procesos auxiliares que se encargan de establecer conexión con el servidor, enviar la petición, recibir la respuesta, controlar los errores y realizar actividades de sincronización y de seguridad.

Los servidores proporcionan un servicio al cliente y devuelven los resultados. En algunos casos existen procesos auxiliares que se encargan de recibir las solicitudes del cliente, verificar la protección, activar un proceso servidor para satisfacer el pedido, recibir su respuesta y enviarla al cliente. Además, deben manejar los interbloqueos, la recuperación ante fallos, y otros aspectos afines. Por las razones anteriores, la plataforma computacional asociada con los servidores es más poderosa que la de los clientes. Además deben manejar servicios como administración de la red, mensajes, control y administración de la entrada al sistema ("login"), auditoría y recuperación y contabilidad. Usualmente en los servidores existe algún tipo de servicio de bases de datos.

Por su parte los servidores realizan, entre otras, las siguientes funciones:

- Gestión de periféricos compartidos.
- Control de accesos concurrentes a bases de datos compartidas.
- Enlaces de comunicaciones con otras redes de área local o exten-

sa.

- Siempre que un cliente requiere un servicio lo solicita al servidor correspondiente y éste, le responde proporcionándolo. Normalmente, pero no necesariamente, el cliente y el servidor están ubicados en distintos procesadores. Los clientes se suelen situar en ordenadores personales y/o estaciones de trabajo y los servidores en procesadores departamentales o de grupo.

Para que los clientes y los servidores puedan comunicarse se requiere una infraestructura de comunicaciones, la cual proporciona los mecanismos básicos de direccionamiento y transporte.

La mayoría de los sistemas Cliente/Servidor actuales, se basan en redes locales y por lo tanto utilizan protocolos no orientados a conexión, lo cual implica que las aplicaciones deben hacer las verificaciones. La red debe tener características adecuadas de desempeño, confiabilidad, transparencia y administración. No obstante dada la velocidad actual en las LAN, se tiende a usar solo protocolos orientados a conexión. Este proyecto usa arquitectura *TCP/IP*[1] .

Entre las principales características de la arquitectura cliente / servidor, se pueden destacar las siguientes:

- El servidor presenta a todos sus clientes una interfaz única y bien definida. El cliente no necesita conocer la lógica del servidor, sólo su interfaz externa.

- El cliente no depende de la ubicación física del servidor, ni del tipo de equipo físico en el que se encuentra, ni de su sistema operativo.

- Los cambios en el servidor implican pocos o ningún cambio en el cliente.

3.2.2 Características funcionales

[1] *TCP/IP*: Ver definición en página 7

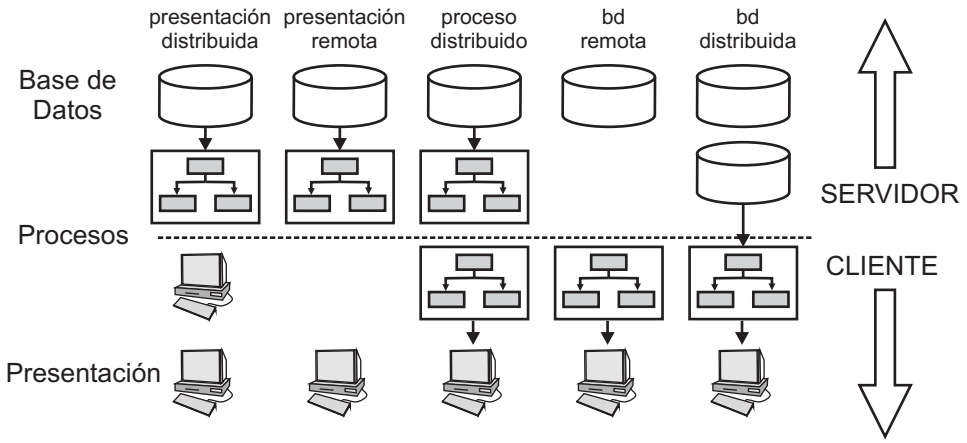


Figura 3.1: Clasificación de diferentes tipos de arquitectura

Mainframe	Un ordenador central o mainframe es un ordenador grande, potente y costoso usado principalmente por una gran compañía para el procesamiento de una gran cantidad de datos; por ejemplo, para el procesamiento de transacciones bancarias.

Esta arquitectura se puede clasificar en cinco niveles, según las funciones que asumen el cliente y el servidor, tal y como se puede ver en la figura 3.1.

- En el primer nivel el cliente asume parte de las funciones de presentación de la aplicación, ya que siguen existiendo programas en el servidor, dedicados a esta tarea. Dicha distribución se realiza mediante el uso de productos para el "maquillaje" de las pantallas del mainframe. Esta técnica no exige el cambio en las aplicaciones orientadas a terminales, pero dificulta su mantenimiento.

- En el segundo nivel, la aplicación está soportada directamente por el servidor, excepto la presentación, que es totalmente remota y reside en el cliente. Los terminales del cliente soportan la captura de datos, incluyendo una validación parcial de los mismos y una presentación de las consultas. En este caso se dice que hay una presentación remota, y de ahora en adelante lo llamaremos arquitectura de cliente delgado.

- En el tercer nivel, la lógica de los procesos se divide entre los distintos componentes del cliente y del servidor. El diseñador de la aplicación debe definir los servicios y las interfaces del sistema de información, de forma que los papeles de cliente y servidor sean intercambiables, excepto en el control de los datos, que es responsabilidad exclusiva del servidor. En este tipo de situaciones se dice que hay un proceso distribuido o cooperativo.
- En el cuarto nivel el cliente realiza tanto las funciones de presentación como los procesos. Por su parte, el servidor almacena y gestiona los datos que permanecen en una base de datos centralizada. En esta situación se dice que hay una gestión de datos remota, y de ahora en adelante lo llamaremos arquitectura de cliente grueso.
- En el quinto y último nivel, el reparto de tareas es como en el anterior y además el gestor de base de datos divide sus componentes entre el cliente y el servidor. Las interfaces entre ambos, están dentro de las funciones del gestor de datos y, por lo tanto, no tienen impacto en el desarrollo de las aplicaciones. En este nivel se da lo que se conoce como bases de datos distribuidas.

: Este proyecto utiliza un modelo de cliente grueso, ya que parte del procesamiento de la información se realiza en el cliente, en la figura 3.1 identificado como “BD remota”.

3.2.3 Características físicas

De la figura 3.1 se obtiene una idea de la estructura física de conexión entre las distintas partes que componen una arquitectura cliente/servidor. La idea principal consiste en aprovechar la potencia de los ordenadores personales para realizar, sobre todo, los servicios de presentación y, según el nivel, algunos procesos o incluso algún acceso a datos locales. De esta forma se descarga al servidor de ciertas tareas para que pueda realizar otras más rápidamente.

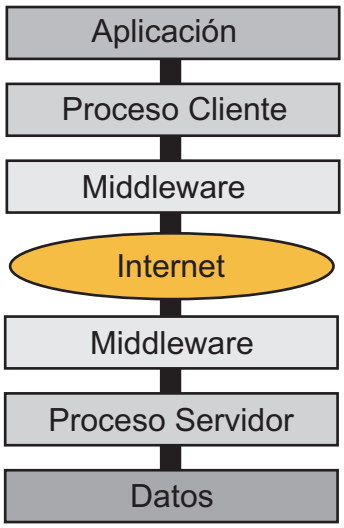


Figura 3.2: Capas en una aplicación cliente/servidor

También existe una plataforma de servidores que sustituye al ordenador central tradicional y que da servicio a los clientes autorizados. Incluso a veces el antiguo ordenador central se integra en dicha plataforma como un servidor más. Estos servidores suelen estar especializados por funciones (seguridad, cálculo, bases de datos, comunicaciones, etc.), aunque, dependiendo de las dimensiones de la instalación se pueden reunir en un servidor una o varias de estas funciones. En nuestro caso se nos ha comunicado que el servidor será un ordenador personal de última generación, con lo que no deberemos sobrecargarlo con mucho procesamiento de datos.

Las unidades de almacenamiento masivo en esta arquitectura, se caracterizan por incorporar elementos de protección que evitan la pérdida de datos y permiten multitud de accesos simultáneos (alta velocidad, niveles RAID, etc.). Acerca de este aspecto no se ha hecho ningún requerimiento en especial.

Para la comunicación de todos estos elementos se emplea un sistema de red que se encarga de transmitir la información entre clientes y servidores. Físicamente consiste en un cableado (coaxial, par trenzado, fibra óptica, etc.) o en conexiones mediante señales de radio o infra-

JDBC	DBC es el acrónimo de Java Database Connectivity, un API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java independientemente del sistema de operación donde se ejecute o de la base de datos a la cual se accede utilizando el dialecto SQL del modelo de base de datos que se utilice.

rrojas, dependiendo de que la red sea local (LAN o RAL), metropolitana (MAN) o de área extensa (WAN).

Para la comunicación de los procesos con la red se emplea un tipo de equipo lógico denominado *middleware* (ver figura 3.2) que controla las conversaciones. Su función es independizar ambos procesos (cliente y servidor). La interfaz que presenta es la estándar de los servicios de red, hace que los procesos "piensen" en todo momento que se están comunicando con una red. En JAVA esto se puede hacer o bien con JDBC directamente o mediante una abstracción superior a nivel de objetos con RMI.

: En lo referente a este proyecto es posible interconectar cliente y servidor mediante una LAN o

InternetInternet. Como middleware se utilizará JDBC directamente.

3.2.4 Elementos de diseño utilizados

En la figura 3.3 se muestra la arquitectura hardware sobre la que se asienta el sistema software:

Se dispone de un ordenador principal que proporciona los servicios de base de datos y de servidor Web. El resto de ordenadores se conectarán a él para poder ejecutar la aplicación correctamente.

La arquitectura de la aplicación que se va a realizar va a seguir el patrón del modelo cliente/servidor en n-capas. De este modo, en

RMI	RMI (Java Remote Method Invocation) es un mecanismo ofrecido en Java para invocar un método remotamente. Al ser RMI parte estándar del entorno de ejecución Java usarlo provee un mecanismo simple en una aplicación distribuida que solamente necesita comunicar servidores codificados para Java. Si se requiere comunicarse con otras tecnologías debe usarse CORBA o SOAP en lugar de RMI. Por medio de RMI, un programa Java puede exportar un objeto. A partir de esa operación este objeto está disponible en la red, esperando conexiones en un puerto TCP. Un cliente puede entonces conectarse e invocar métodos. La invocación consiste en el "marshaling" de los parámetros (utilizando la funcionalidad de "serialización"que provee Java), luego se sigue con la invocación del método (cosa que sucede en el servidor). Mientras esto sucede el llamador se queda esperando por una respuesta. Una vez que termina la ejecución el valor de retorno (si lo hay) es serializado y enviado al cliente. El código cliente recibe este valor como si la invocación hubiera sido local.

el servidor se ejecutan los servicios mencionados anteriormente y los equipos clientes se conectan al servidor.

En la figura 3.4 se puede ver un modelo simplificado de esta arquitectura.

Capa de Usuario

En esta capa se presenta la información y se lleva a cabo la interacción usuario-aplicación. Por ello en esta capa se implementan todos los aspectos de la aplicación relativos a la interfaz de usuario. Algunas funciones en esta capa son las siguientes:

- Formularios para la conexión a la aplicación
- Pantallas para el acceso y modificación de la base de datos
- Visualización de datos

Middleware	El Middleware es un software de conectividad que permite ofrecer un conjunto de servicios que hacen posible el funcionamiento de aplicaciones distribuidas sobre plataformas heterogéneas. Funciona como una capa de abstracción de software distribuida que se sitúa entre las capa de aplicaciones y las capas inferiores (sistema operativo y red). El Middleware nos abstrae de la complejidad y heterogeneidad de las redes de comunicaciones subyancentes, así como de los sistemas operativos y lenguajes de programación, proporcionando una API para la fácil programación y manejo de aplicaciones distribuidas. Dependiendo del problema a resolver y de las funciones necesarias serán útiles diferentes tipo de servicios de middleware. Por lo general el middleware del lado cliente esta implementado por el Sistema Operativo subyacente, el cual posee las librerías que implementan todas las funcionalidades para la comunicación a través de la red.

Esta capa se comunica con la inmediatamente inferior para solicitar estos servicios:

1. Peticiones de información para obtener los datos que se le presentarán al usuario.
2. Actualización de información para reflejar en la base de datos las modificaciones que haya efectuado el usuario.

Capa de Procesamiento de Información

En esta capa se implementarán clases para gestionar la información que recibe de la base de datos y poder acceder a estos datos de manera ordenada y jerarquizada. Esta capa se comunica con la capa superior de Usuario y con la inferior de administración de datos. De este modo:

1. Proporciona a la capa superior los servicios que le solicita.
2. Se comunica con la capa inferior para obtener o modificar los datos de la capa superior.

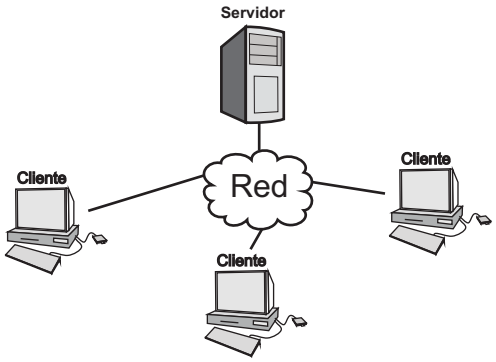


Figura 3.3: Arquitectura Hardware de la aplicación

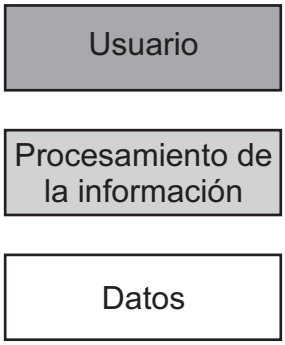


Figura 3.4: Arquitectura Lógica de la aplicación

3. Sirve como nexo de unión entre las capas inferior y superior. Se pueden destacar dos funciones básicas en esta capa:

   - Sistema de presentación de datos, formado por un conjunto de clases que permiten manipular los datos de forma cómoda.
   - Sistema de transformación de datos, que se encarga de transformar los datos directamente obtenidos de la base de datos a instancias de una clases y viceversa, transformar una instancia de una clase a un lenguaje entendible por el sistema de gestión de la base de datos.

Capa de administración de datos

Se encarga de la interacción a más bajo nivel con la base de datos. Además es la capa que utiliza la tecnología de JDBC.

Se ha optado por este modelo por lo siguiente:

- Nos permite realizar las tareas de diseño, implementación y pruebas de cada parte de forma independiente facilitando la corrección de errores y la separación en tareas.
- Permite la fácil ampliación de la funcionalidad de la aplicación tanto en la parte del servidor como en la de los clientes.
- La aplicación es más portable.

Comunicación entre elementos: JDBC

El API JDBC se presenta como una colección de interfaces Java y métodos de gestión de manejadores de conexión hacia cada modelo específico de base de datos. Un manejador de conexiones hacia un modelo de base de datos en particular es un conjunto de clases que implementan las interfaces Java y que utilizan los métodos de registro para declarar los tipos de localizadores a base de datos (URL) que pueden manejar. Para utilizar una base de datos particular, el usuario ejecuta su programa junto con la librería de conexión apropiada al modelo de su base de datos, y accede a ella estableciendo una conexión. Para ello provee la localización de la base de datos y los parámetros de conexión específicos y, a partir de allí puede realizar con cualquier tipo de tareas con la base de datos a las que tenga permiso: consultas, actualizaciones, creado modificado y borrado de tablas, ejecución de procedimientos almacenados en la base de datos, etc.

3.3 Arquitectura de SARIC

En el hardware no existen procesos de diseño tan estandarizados como en el software, no obstante es posible aplicar los mismos conceptos. En el principio de este capitulo, se han expuesto los requisitos mínimos que debe cumplir el sistema al completo, en esta sección se presentan detalles del diseño que aseguran que se cumplan esas características.

Es neceario diseñar una arquitectura que permita interconectar todos los componentes del sistema y que mantenga las condiciones

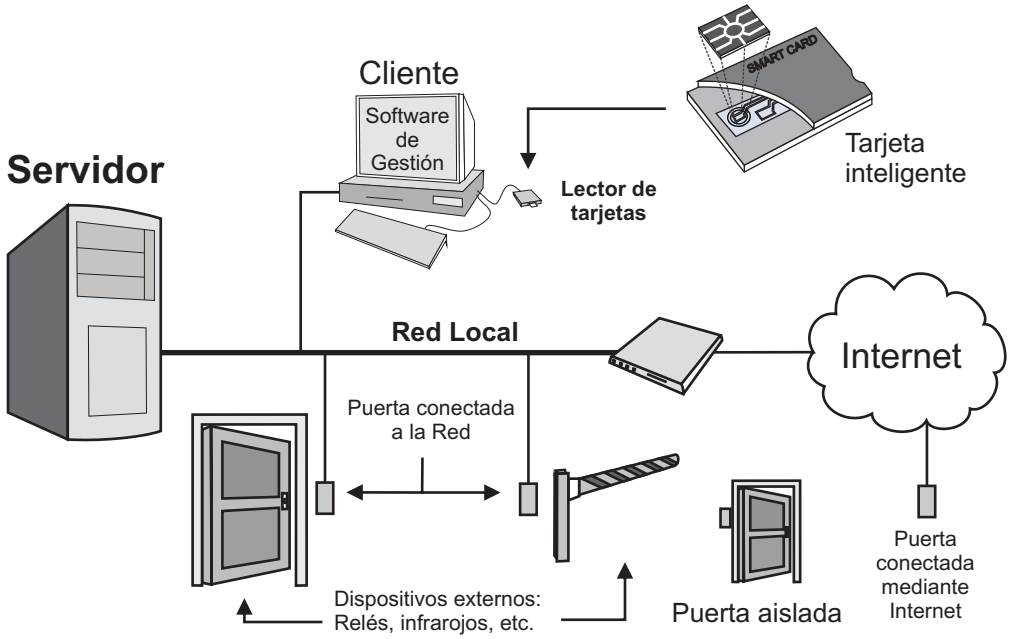


Figura 3.5: Estructura completa de SARIC

de fiabilidad y seguridad. La solución final adoptada se muestra en la figura 3.5. El sistema esta formado por tres componentes principales:

Servidor: Aquí se encuentra almacenada la base de datos, a la que accederán los dispositivos que se encuentran conectados; opcionalmente también es posible instalar un servidor Web que permita el acceso cómodo a los puntos que disponen de conexión a la red.

Estación PC: Es posible conectar al sistema cualquier número de estaciones clientes que se desee, incluso es posible conectar el sistema a

InternetInternet para poder acceder a este desde una estación remota. Estas estaciones podrán ejecutar una aplicación que gestione todo el Sistema y realice todas las funciones descritas en los objetivos. Además contarán con un lector de tarjetas y, gracias a esto, se podrá unificar el método de autentificación, tanto en la aplicación de gestión como en los puntos de acceso, que será la tarjeta más

un código de identificación o PIN. (Ver sección anterior)

Accesos: En cada lugar que se desee controlar es necesario instalar un punto de acceso. Cumpliendo con los objetivos, se dispone de dos tipos de dispositivos:

Con conexión a la Red: Estos dispositivos están conectados fisicamente a la red local o bien a

InternetInternet, permitiendo así la interconexión de distintos edificios. Además contendrán un pequeño servidor Web que mostrara información de su estado permitiendo interactuar con ellos desde cualquier punto de la red.

Sin conexión a la Red: Estos dispositivos se encuentran aislados del resto. Disponen de puertos para el acceso a datos de bitácora almacenados en su interior. Su actualización se realizará de forma manual *in-situ*.

Es posible conectar periféricos externos a estos dispositivos como: relés, detectores de infrarrojos, barreras de parking, etc.

3.4 Seguridad en SARIC

En el mercado existen muchos tipos de tarjetas inteligentes, con diferentes procesadores, con más o menos memoria, etc. Las usadas en este proyecto son muy básicas y no disponen de un sistema de codificación de datos integrado. Por lo tanto es necesario implementarlo vía software, lo que nos abre un abanico enorme de posibilidades en cuanto a algoritmos de encriptación.

Existen dos grandes tipos de cifrado: los algoritmos que utilizan una única clave tanto en el proceso de cifrado como en el de descifrado y los que utilizan una clave para cifrar mensajes y una clave distinta para descifrarlos. Los primeros se denominan cifras simétricas o de clave simétrica y son la base de los algoritmos de cifrado clásico. Los segundos se denominan cifras asimétricas, de clave asimétrica o de clave pública y clave privada y forman el núcleo de las técnicas de cifrado

modernas.

Los sistemas de cifrado de clave pública o sistemas de cifrado asimétricos se inventaron con el fin de evitar por completo el problema del intercambio de claves de los sistemas de cifrado simétricos. Con las claves públicas no es necesario que el remitente y el destinatario se pongan de acuerdo en la clave a emplear. Todo lo que se requiere es que, antes de iniciar la comunicación secreta, el remitente consiga una copia de la clave pública del destinatario. Es más, esa misma clave pública puede ser usada por cualquiera que desee comunicarse con su propietario. Por tanto, se necesitarán sólo n pares de claves por cada n personas que deseen comunicarse entre sí.

La mayor ventaja de la criptografía asimétrica es que se puede cifrar con una clave y descifrar con la otra, pero este sistema tiene bastantes desventajas:

- Para una misma longitud de clave y mensaje se necesita mayor tiempo de proceso.
- Las claves deben ser de mayor tamaño que las simétricas.
- El mensaje cifrado ocupa mas espacio que el original.

Dado que SARIC es un circuito cerrado, es decir el intercambio de claves se lleva a cabo directamente sin ningún tipo de intermediaros, la principal ventaja de los sistemas de encriptación asimétricos es inútil. Además dado que no se dispone de demasiada capacidad de procesamiento, ni de memoria abundante, las desventajas de éste adquieren una especial importancia.

Por todo esto, se ha decidido usar un algoritmo de clave privada o criptografía simétrica. Aún eliminando todos los algoritmos asimétricos, las posibilidades siguen siendo enormes ya que existen numerosos algoritmos de este tipo. No todos ofrecen los mismos niveles de seguridad ni requieren el mismo tiempo de procesamiento, por ello se realizó un pequeño estudio de los más importantes analizando: tiempo de

desencriptado, longitud de clave y criptoanálisis conocidos. Los algoritmos estudiados fueron: DES, AES, Blowfish, IDEA y TEA.

Hoy por hoy, los ordenadores pueden calcular claves con extrema rapidez, y ésta es la razón por la cual el tamaño de la clave es importante en los criptosistemas modernos. El algoritmo de cifrado DES usa una clave de 56 bits, lo que significa que hay 2 elevado a 56 claves posibles. 2 elevado a 56 son 72.057.594.037.927.936 claves. Esto representa un número muy alto de claves, pero un PC de uso general puede comprobar todo el dominio de claves en cuestión de días. Un máquina especializada lo puede hacer en horas. Por otra parte, algoritmos de cifrado de diseño más reciente como 3DES, Blowfish e IDEA usan claves de 128 bits, lo que significa que existen 2 elevado a 128 claves posibles. Esto supone un incremento enorme de claves, y aun en el caso de que todas las máquinas del planeta estuvieran cooperando, todavía tardarían más tiempo que la misma edad del universo en encontrar la clave.

> **TEA**: TEA es un algoritmo extremadamente rápido, además es muy fácil de implementar. La longitud de clave es de 128 bits, lo cual hace prácticamente imposible su adivinación por métodos de fuerza bruta. Además los ataques de criptoanálisis conocidos, requieren de grandes cantidades de texto codificado con la misma clave para resultar efectivos. Por ello se ha confiado a este algoritmo la seguridad de SARIC.

A pesar de esto, un algoritmo de encriptación no puede evitar que se modifiquen los datos del interior de la tarjeta, ya sea por un acto malintencionado o por una interferencia electromagnética. Por este motivo en SARIC se usa una segunda protección, un compendio CRC de 16 bits de todo el contenido de la tarjeta. Así es posible saber si los datos han sufrido algún tipo de modificación desde que fueron grabados por el Gestor de SARIC.

CRC	Los códigos cíclicos también se llaman CRC (Códigos de Redundancia Cíclica) o códigos polinómicos. Su uso está muy extendido porque pueden implementarse en hardware con mucha facilidad y son muy potentes. Estos códigos se basan en el uso de un polinomio generador $G(X)$ de grado r, y en el principio de que n bits de datos binarios se pueden considerar como los coeficientes de un polinomio de orden $n-1$.

3.5 Distribución de los datos en la SmartCard

Las tarjetas inteligentes disponen de 256-bytes de memoria, de los cuales los 32 primeros están ocupados por datos del fabricante, lo que nos deja con 224 bytes útiles. Es necesario almacenar: el nombre, las puertas a las que se tiene acceso y el compendio CRC comentado en la sección anterior. Además será necesario almacenar la clave de seguridad, si bien para este propósito, la SmartCard incluye una memoria especial protegida para evitar su lectura. En la figura 3.6 se muestra la distribución de los datos. Se han reservado 32 bytes para el nombre, 16 para el CRC y los 176 restantes para los códigos de las puertas a las que el propietario de la tarjeta tiene acceso, lo que nos deja con espacio para 88 puertas o grupos de puertas ya que cada código consume 2-bytes.

Si no se usan los 176 bytes, tras el último código se grabará un byte como 0x00 y los siguientes serán datos aleatorios. Esto evita posibles ataques contra el algoritmo de encriptación.

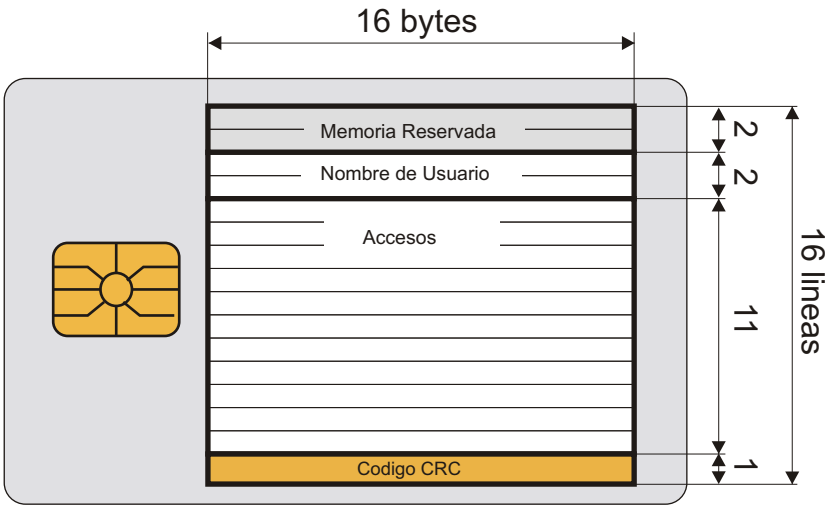


Figura 3.6: Distribución de los datos en la SmartCard

4 Temporización y Desarrollo

En principio, observando el sistema es fácil detectar que hay tres grandes apartados en los que podemos subdividirlo:

- Software
- Hardware
- Firmware

Cuando se aceptó el proyecto, el primer problema con el que hubo que enfrentarse fue resolver de qué forma iba a ser posible conjugar estos tres apartados y cómo se podría temporizar algo que jamás habían hecho antes los integrantes del grupo. La respuesta no es sencilla ya que el proyecto implica además tres formas diferentes de trabajar, una por cada apartado.

El desarrollo de software permite una interactividad muy superior al de hardware, donde las pruebas de validación se realizan únicamente en las últimas fases de fabricación y funciona de forma iterativa. El firmware, puede ser desarrollado con simuladores, pero requiere el conocimiento a bajo nivel del hardware.
Además, las herramientas que se utilizan para cada apartado, de nuevo, son muy distintas entre sí, con la salvedad de poder usar JAVA en el software de Gestión y en el firmware, aunque de una forma completamente diferente. Las restricciones de la aplicación y del Servidor y cliente de Bases de Datos las impone el tipo de ordenador, la red de

interconexión, el sistema operativo y el lenguaje de programación, pero en el firmware las restricciones cambian al tener que manejar una memoria limitada y puertos donde la temporización de los pulsos es crucial. El hardware, por su parte, implica una forma de trabajar totalmente desconocida por los desarrolladores del proyecto.
Por todas estas dificultades que entrañan el estudio y aplicación de las técnicas de implementación de SARIC, el equipo decide utilizar una metodología de desarrollo flexible y ágil, que permita continuas revisiones y diálogo con los tutores en todo momento. La metodología elegida, como se puede leer en los capítulos introductorios es la metodología de Programación Extrema o XP, también aplicada al desarrollo de hardware y firmware.

En este capítulo se detallan las subpartes en que se ha divido el proyecto y los objetivos a corto y largo plazo que se han alcanzado desde sus inicios.
No es objetivo de este capítulo el explicar de nuevo las bases de la metodología XP[1], sino indicar cómo han sido aplicadas específicamente a este proyecto.

La primera sección introduce algunos conceptos que luego serán manejados a lo largo del capítulo. Las 3 secciones restantes corresponden a las versiones o etapas que han ido sucediéndose a lo largo del proyecto. Sus nombres son sólo orientativos ya que dentro de cada versión han tenido que mezclarse tareas de software, hardware y firmware, no obstante no todos estos aspectos tienen el mismo peso dentro de SARIC.

4.1 Metodología de trabajo con XP

La programación extrema o eXtreme Programming (XP) es una aproximación a la ingeniería de software, un proceso ágil de desarrollo de software. Sus características principales son: desarrollo iterativo

[1]Para leer las bases de la Metodología XP, ir al apartado correspondiente en el capítulo 2: “Metodologías, tecnologías y herramientas utilizadas”

e incremental, pruebas unitarias continuas, programación por parejas, frecuente interacción del equipo de programación con el cliente o usuario, corrección de todos los errores antes de añadir nueva funcionalidad, refactorización del código, propiedad del código compartida y Simplicidad en el código.

4.1.1 Conceptos Específicos de XP

La metodología XP, cómo ya se explicó en el capítulo 2, ha sido elegida para la realización de este proyecto por ser flexible, ligera y potente. En este apartado se detalla cada una de las fases y subfases comunes a todos los proyectos XP.

Todas estas ideas, sin las cuales no es posible planificar y dividir el proyecto, han sido plasmadas de forma visual en el esquema de la figura 4.1.

- División del trabajo y terminología:

  Un desarrollo mediante programación extrema está compuesto por una serie de iteraciones cortas. Cada iteración concluye ejecutando un conjunto de pruebas de aceptación que permitan al cliente comprobar si está satisfecho con el resultado. En XP no existe una fase de requisitos propiamente dicha, en su lugar, al comienzo de cada iteración, se lleva a cabo el juego de la planificación (Ver figura 4.2). En él, el cliente y el equipo de desarrollo negocian el alcance del proyecto para una iteración, identifican un conjunto de historias de usuario y seleccionan las más importantes para el cliente para ser implementadas en la siguiente iteración. Las pruebas de aceptación se elaboran a lo largo de la iteración, en paralelo con el desarrollo del sistema, y adaptándose a los cambios que el sistema sufra.

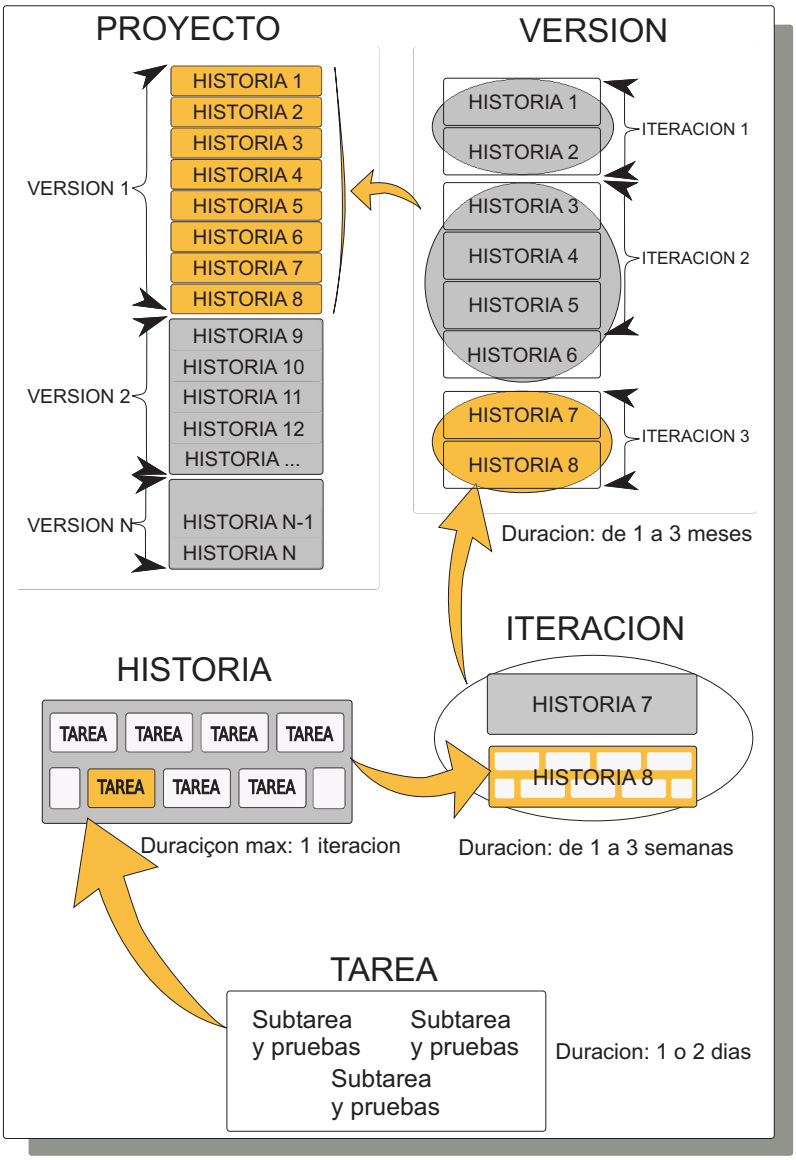


Figura 4.1: Diagrama que muestra las divisiones en tiempo y trabajo en XP

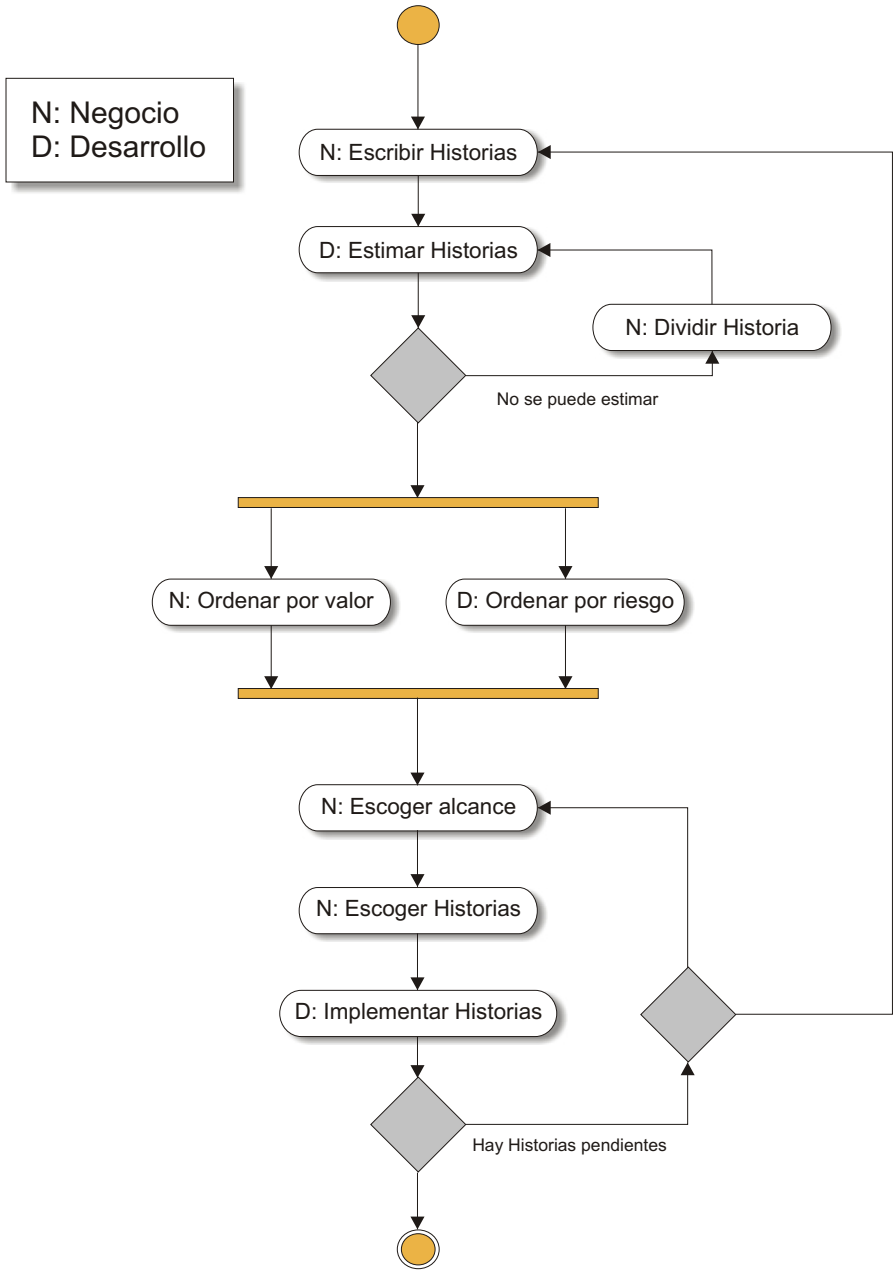


Figura 4.2: El juego de la planificación

Historia:

Historia (XP)	Descripción de un objetivo o necesidad del producto que se va a desarrollar, realizada por el cliente y supervisada por los desarrolladores de forma que se elimine toda ambigüedad y pueda ser estimada y probada.

Cada historia descrita por el cliente y su estimación realizada por los desarrolladores se escriben en una tarjeta. Una historia debe de ser lo suficientemente pequeña como para poder desarrollarse en una iteración, de una a tres semanas. En programación extrema, las historias se utilizan para suplir la especificación de requerimientos, falta de casos de uso y otras herramientas de la ingeniería del software clásica. En XP no existe una fase de requisitos propiamente dicha, en su lugar, al comienzo de cada iteración, se lleva a cabo el juego de la planificación. En él, el cliente y el equipo de desarrollo negocian el alcance del proyecto para una iteración, identifican un conjunto de historias de usuario y seleccionan las más importantes para el cliente para ser implementadas en la siguiente iteración. Ver figura 4.3.

El cliente debe de ser capaz de especificar las pruebas de aceptación que verifiquen que la historia es correcta y completa. Además, debe de poder priorizar todas las historias, y dividirlas para que no haya grandes diferencias entre las prioridades. La pruebas de aceptación no están basadas en las historias de uso sino en la funcionalidad que conoce el cliente asociada a dicha historia de uso. Por tanto cualquier propuesta de generación de pruebas basada sólo en historias de uso será incompleta y, por eso, necesitamos que sea el usuario quien las escriba.

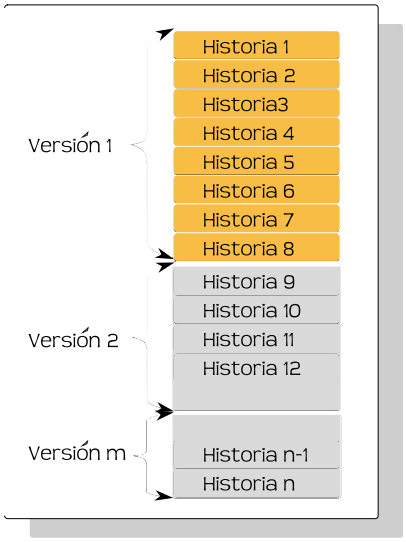


Figura 4.3: Diagrama simple de un proyecto XP: Proyecto-Versión

Punto de fijación:

Punto de fijación	Prueba rápida que lleva a profundizar sobre un aspecto en cuestión y que permite estimar el trabajo y tiempo que serán invertido en el desarrollo de una historia.

Los puntos de fijación se preparan en el periodo de exploración, y no formarán parte de las historias. Su duración depende de la importancia de la prueba y varía entre 1 día y varias semanas.

Prueba de Aceptación:

Prueba de Aceptación	Criterio de finalización de una historia definido por el cliente

Como se ha dicho un desarrollo mediante programación extrema está compuesto por una serie de iteraciones cortas; cada iteración concluye ejecutando un conjunto de pruebas de aceptación que permitan al cliente comprobar si está satisfecho con el resultado. Las pruebas de aceptación se elaboran a lo largo de la iteración, en paralelo con el desarrollo del sistema, y adaptándose a los cambios que el sistema sufra.

Tarea:

Tarea	Cada una las partes que componen una historia. Su duración no ha de ser superior a uno o dos días.

A la hora de obtener las tareas que componen una historia, el cliente también participa pero en menor medida, ya que la experiencia del quipo de desarrollo garantizará una distribución temporal mas homogénea. Las tareas no son asignadas, los programadores se comprometen con aquellas en las que quieren trabajar. Ninguna tarea se puede dejar sin elegir. Además se intentará dar prioridad a las tareas con las que se obtienen resultados directos.

No se permite que los programadores se responsabilicen de más tareas de las que se completaron en la iteración anterior.

Todo el software de producción se escribe por parejas (aunque cada programador sea el responsable de las tareas individualmente). Una pareja de desarrolladores debe compartir ordenador, teclado y ratón. El principal objetivo es realizar de forma continua y sin parar el desarrollo una revisión de diseño y de código...

Caso de prueba:

Casos de prueba	Criterios de verificación de las tareas y subtareas, que son susceptibles de ser programados y que se alternan con la creación del código del programa de forma continua cada pocos minutos

Las pruebas son tan importantes como la producción de código. Se reúnen en las pruebas de unidad y son ejecutadas cada vez que cambia el código. Se deben pasar todas las pruebas tanto antes como después de que el código sea integrado en el sistema principal.
Uno de los lemas de XP dice: “Escribe primero la prueba, después el código”. Eso demuestra el papel tan importante que tienen los casos de prueba en la programación extrema.
En la práctica, los programadores ejecutan frecuentemente las pruebas para asegurarse de que no han estropeado nada de forma inadvertida.

- División del tiempo y terminología:

Version:

Versión	Fases en que se divide el desarrollo de un proyecto. Cada una de ellas proporciona un valor de negocio al cliente. Su duración puede ser de uno a tres meses.

Cuando se planifica una versión hay que seguir una regla sencilla: El equipo no puede comprometerse a hacer más trabajo del que había hecho en la versión anterior. Este trabajo se mide en términos de estimaciones que se escriben en las tarjetas. La idea para obtener las versiones es: "En cuanto el software aporte algún valor añadido, se debe obtener una nueva versión".

Exploración:

Exploración	Fase inicial de todo proyecto XP en la que los desarrolladores han de identificar, priorizar y estimar los requisitos.

En esta fase, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración puede llevar de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.

Cuando se identifican requisitos suficientes como para proporcionar el sistema más pequeño posible que aporte valor al cliente, se planificará la primera versión. Durante el desa-

rrollo del proyecto pueden realizarse más exploraciones que darán lugar a diferentes versiones.

Velocidad de la versión:

Velocidad de la versión	Suma de las estimaciones en tiempo de las historias que serán cubiertas en cada versión.

Las estimaciones de esfuerzo asociado a la implementación de las historias la establecen los programadores utilizando como medida el número de semanas ideales de programación. Las historias generalmente valen de 1 a 3 semanas. Por otra parte, el equipo de desarrollo mantiene un registro de la "velocidad" de desarrollo, establecida en semanas por iteración, basándose principalmente en la suma de semanas correspondientes a las historias de usuario que fueron terminadas en la última .

La planificación se puede realizar basándose en el tiempo o el alcance. La velocidad del proyecto es utilizada para establecer cuántas historias se pueden implementar antes de una fecha determinada o cuánto tiempo llevará implementar un conjunto de historias. Al planificar por tiempo, se multiplica el número de iteraciones por la velocidad del proyecto, determinándose cuántas semanas ideales de programación se pueden completar. Al planificar según alcance del sistema, se divide la suma de semanas de las historias de usuario seleccionadas entre la velocidad del proyecto, obteniendo el número de iteraciones necesarias para su implementación.

Iteración:

Iteración	Cada una de las subfases en que se divide cada versión. Su duración varía entre 1 y 3 semanas.

Ver esquema Versión-Iteración de la figura 4.4

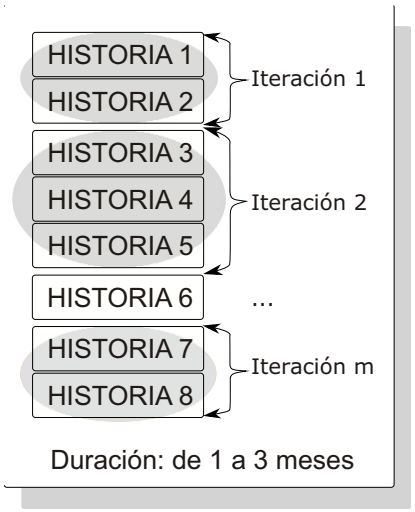


Figura 4.4: Diagrama simple de un proyecto XP: Versión-Iteración

El plan de entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que fuercen la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que es el cliente quien decide qué historias se implementarán en cada iteración (para maximizar el valor de negocio). Al final de la última iteración el sistema estará listo para entrar en producción.

Los elementos que deben tomarse en cuenta durante la elaboración del plan de la iteración son: historias de usuario no abordadas, velocidad del proyecto, pruebas de aceptación no superadas en la iteración anterior y tareas no terminadas en la iteración anterior. Todo el trabajo de la iteración es expresado en tareas de programación, cada una de ellas es asignada a un programador como responsable, pero llevadas a cabo por parejas de programadores.

Semana Ideal de Programación	Cantidad de código que puede ser realizado por una persona durante una semana suponiendo que se aprovecha el tiempo de trabajo de forma totalmente eficiente

4.1.2 Criterios de temporización

Antes de comenzar a producir el software y el hardware, hay que fijar ciertas variables que influyen sobre la velocidad del proyecto.

- Semana ideal de programación:
  Cada historia lleva asociada una estimación medida en "semana ideal de programación".

  La semana ideal de programación para cada uno de los integrantes del proyecto ha sido estimada en **10 horas** (2 horas por día), ya que es necesario compaginar el proyecto con otras asignaturas a diferentes horarios. Esta semana ideal de trabajo no se aplica a días festivos ni periodos vacacionales. Más aun, hay que tener en cuenta que los días inmediatamente anteriores a los exámenes son tomados a efectos prácticos como libres.

  La semana ideal de programación se utiliza como medida de estimación de las historias.

- Velocidad de la versión:
  La velocidad de versión para el proyecto ha sido estimada en **3 meses**.
  Por la naturaleza del proyecto puramente didáctica y dado que se precisa de revisiones continuas por los tutores del proyecto, es preferible que la velocidad de versión sea mínima. Por otro lado, con unas pocas horas al día es difícil tener una versión con suficiente valor de negocio. Como término medio se ha elegido una velocidad de 3 meses.

  Dado que cada mes comprende 4 semanas de programación, ca-

da versión comprende 12 semanas ideales de programación y el trabajo será realizado por dos programadores:

$$V_{version_{persona}} = 3\,meses * 4\,semanas = 12\,semanas$$
$$V_{version_{2\,personas}} = V_{version_{persona}} * 2\,personas = 24\,semanas$$

Para cada versión solamente pueden ser seleccionadas historias cuyas estimaciones, en suma, no superen las **24 semanas ideales de programación**.

- Velocidad de la iteración:
  La velocidad de la iteración para el proyecto ha sido estimada en **3 semanas**. Cada versión estará dividida en **4 iteraciones**

  Siguiendo la lógica usada para el cálculo de la velocidad de versión, se estima que cada iteración solo puede contener historias cuyas estimaciones no superen las **6 semanas ideales de programación**.

  Dado que cada mes comprende 4 semanas de programación, cada versión comprende 12 semanas ideales de programación y el trabajo será realizado por dos programadores:

$$V_{iteracion_{persona}} = 3\,semanas$$
$$V_{iteracion_{2\,personas}} = V_{iteracion_{persona}} * 2\,personas = 6\,semanas$$

  Las velocidades de versión e iteración quedan como se ve en la figura 4.5

- Calendario:
  Como el proyecto se paraleliza con diversas asignaturas de la carrera, es necesario dejar unos meses libres para la preparación de los exámenes. Si a esto unimos el periodo normal de vacaciones, obtendremos un calendario con los días laborables y no laborables del año. Se considerarán como periodos no laborables en lo que concierne al proyecto los siguientes:

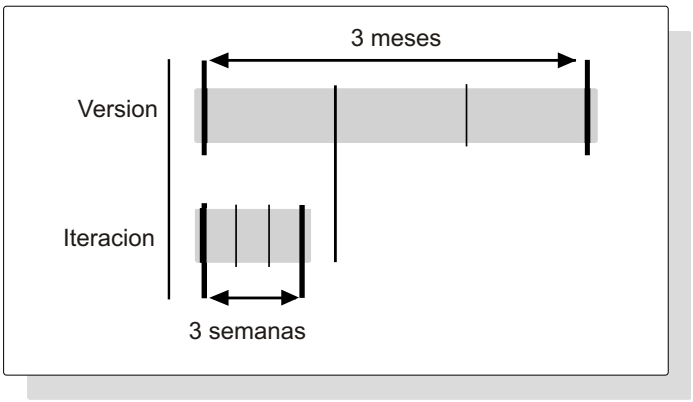
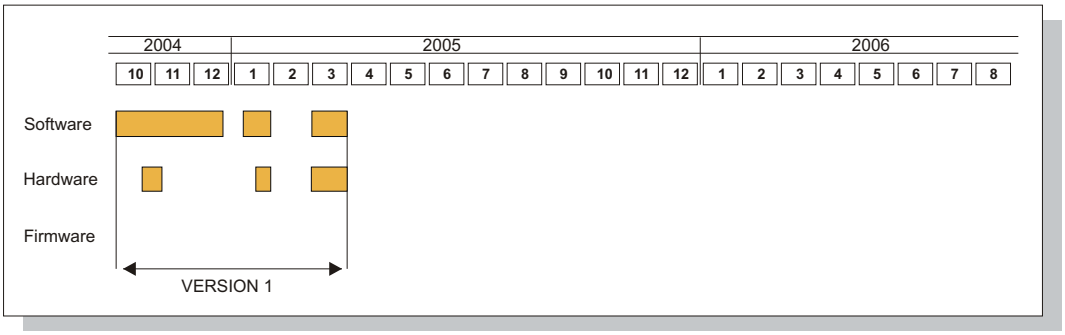


Figura 4.5: Velocidad de Versión e Iteración para el proyecto

- meses de Febrero, Junio y Septiembre: A causa de los exámenes que se realizan estos meses hemos preferido no dedicarle tiempo al proyecto.
- vacaciones de Navidad del día 25 de Diciembre al 8 de Enero.
- segunda quincena de Julio de 2005: Por vacaciones de verano.
- mes de Abril de 2005: Las vacaciones de Semana Santa unidas al viaje de fin de carrera suponen dejar este mes como periodo no laborable.
- resto de días señalados en el calendario como festivos.

4.2 Versión 1: Diseño e Implementación del Software



4.2.1 Exploración de la Versión 1

Duración: 6 semanas.

Tras una primera exploración se definen las siguientes historias, consideradas como suficientes para la primera versión:

> : A partir de este momento se utilizará el término '**unidad de estimación**' cada vez que se haga referencia a la estimación de una historia, para evitar la confusión entre semana ideal de trabajo por persona y semana real de trabajo (en este caso, al tratarse de dos programadores, corresponde a dos semanas ideales de trabajo)

Historia	Estimación
Login del usuario sin tarjeta	5
Visualización de los datos de usuario	1
Modificación de los datos de usuario	2
(Restricción) Sólo el administrador puede cambiar datos de otros usuarios	
Inserción y borrado de nuevos usuarios	3
Modificación de los datos de otros usuarios	2
(Restricción) No implementar búsquedas de usuarios	
Borrar tarjeta	1
Grabación y lectura de datos personales en la tarjeta	x
Logeo por tarjeta	x
Fabricación del dispositivo con conexión	x
Fabricación del dispositivo sin conexión	x

Posibilidad de hacer fotos con una cámara web	3
Gestionar las puertas y grupos de puertas	3
Gestionar permisos	2
Gestionar fechas de caducidad	2
(Restricción) Interfaz ergonómica	

Algunas de las historias obtenidas son muy ambiguas y extensas, por ejemplo la de fabricación del dispositivo con y sin conexión. Hay que redefinirlas y dividirlas deforma que encajen dentro de una versión.

Además, el equipo decide preparar puntos de fijación para poder estimar con mayor eficacia la dificultad de las historias:

- Login del usuario sin tarjeta.
  - Punto de fijación: Instalación y configuración del Servidor GNU/Linux.
    Duración: 1 semana.
  - Punto de fijación: Instalación y configuración de Entorno de Programación Java Netbeans.
    Duración: 1 semana
- Grabación y lectura de datos personales en la tarjeta
  - Punto de fijación: Estudio de las tarjetas inteligentes y de memoria.
    Duración: 1 semanas.
  - Punto de fijación: Estudio de la biblioteca de Java OpenCard.
    Duración: 2 semanas.

Cada historia debe de quedar completamente delimitada, y ser verificable. A continuación se detallan los cambios que se realizaron sobre las historias:

- La historia 'Login del usuario sin tarjeta' se intercambia por las historias:
  - Creación de una Base de Datos sencilla de usuarios.
  - Creación de un programa de identificación que haga de cliente de base de datos.
- La historia 'Gestionar puertas y grupos de puertas' se intercambia por las historias:
  - Diseño de una Base de Datos que permita la gestión de puertas y grupos
  - Creación de un módulo que permita crear y borrar puertas y grupos
- La historia 'Grabación y lectura de datos personales en la tarjeta' se intercambia por las historias:
  - Diseño y estructuración de los datos en memoria de tarjeta
  - Asignación de un código pin
  - Grabación y lectura de los datos de tarjeta
  - Verificación de los datos de tarjeta
- La historia 'Visualización de los datos de usuario' añade la historia:
  - Diseño de interfaz

Las historias sobre fabricación del dispositivo con conexión y sin conexión son tan amplias y ambiguas que su estudio se transfiere a la exploración correspondiente por ser excesivo para esta.

4.2.2 Planificación de la Versión 1

Las historias disponibles actualmente son ordenadas por importancia siguiendo los criterios de la programación extrema, elegiremos una configuración de estas que nos permita al final de la primera versión disponer de la mayor funcionalidad posible:

Historia 1

Creación de una Base de Datos sencilla de usuarios

estimación: 1

Historia 2

Interfaz ergonómica

restricción

Historia 3

Diseño de interfaz

estimación: 2

Historia 4

Creación de un programa de login que conecte a la Base de Datos

estimación: 2

Historia 5

Visualización de los datos de usuario

estimación: 2

Historia 6

Modificación de los datos de usuario

estimación: 2

Historia 7

Sólo el administrador puede cambiar datos de otros usuarios

restricción

Historia 8

Inserción y borrado de nuevos usuarios

estimación: 2

Historia 9

Modificación de los datos de otros usuarios

estimación: 2

Historia 10

No implementar búsquedas de usuarios

restricción

Historia 11

Borrar tarjeta

estimación: 1

Historia 12

Diseño y estructuración de los datos en memoria de tarjeta

estimación: 3

Historia 13

Asignación de un código pin a la tarjeta

estimación: 1

Historia 14

Acceso a la aplicación por tarjeta

estimación: 1

Historia 15

Grabación y lectura de los datos de la tarjeta

estimación: 3

Historia 16

Verificación de los datos de tarjeta

estimación: 1

Historia 17

Diseño de una Base de Datos que permita la gestión de puertas y grupos

estimación: 1

Historia 18

Creación de un módulo que permita crear y borrar puertas y grupos

estimación: 1

Historia 19

Gestionar permisos

estimación: 3

Historia 20

Gestionar fechas de caducidad

estimación: 2

Historia 21

Permitir hacer fotos con una cámara web

estimación: 3

Historia 22

Fabricación del dispositivo con conexión

estimación: ?

Historia 23

Fabricación del dispositivo sin conexión

estimación: ?

Al seleccionarlas hay que tener en cuanta que no pueden superar un máximo de 24 unidades de estimación. Cada iteración contiene historias cuyas estimaciones no superen las 6 unidades.

Nº	Historia	Est.	Iteración.
1	**Creación de una Base de Datos sencilla de usuarios**	1	1.1
3	**Diseño de interfaz**	2	1.1
4	**Creación de un programa de login que conecte a la Base de Datos**	2	1.1
5	**Visualización de los datos de usuario**	2	1.2
6	**Modificación de los datos de usuario**	2	1.2
8	**Inserción y borrado de nuevos usuarios**	2	1.2
9	**Modificación de los datos de otros usuarios**	2	1.3
11	**Borrar tarjeta**	1	1.3
12	**Diseño y estructuración de los datos en memoria de tarjeta**	3	1.3
13	**Asignación de un código pin a la tarjeta**	1	1.4
14	**Acceso a la aplicación por tarjeta**	1	1.4
15	**Grabación y lectura de los datos de la tarjeta**	3	1.4
16	**Verificación de los datos de tarjeta**	1	1.4
17	**Diseño de una Base de Datos que permita la gestión de puertas y grupos**	1	1.4
		24	

El Diagrama de Gantt del tiempo empleado para las historias de la primera versión es el correspondiente a la figura 4.6.

Nombre	Trabajo
Exploracion1	3d
Inst / Conf de Servidor GNU/Linux	10d
Inst / Conf de Entorno de Programación	10d
Estudio de las Tarjetas Inteligentes	5d
Estudio de la biblioteca OpenCard	10d
Exploracion2	3d
BD de usuarios	2d
Diseño de interfaz	10d
Programa de login que conecte con la BD	10d
Visualización de los datos de usuario	10d
Modificación de los datos de usuario	10d
Inserción y borrado de nuevos usuarios	5d
Modificación de datos de otros usuarios	10d
Borrar tarjeta	10d
Estructuración de los datos en SmartCard	8d
Asignación de un código pin a la tarjeta	2d
Acceso a la aplicación por tarjeta	3d
Grabación / Lectura de los datos en tarjeta	8d
Verificación de los datos de tarjeta	3d
BD que permita gestionar puertas y grupos	3d

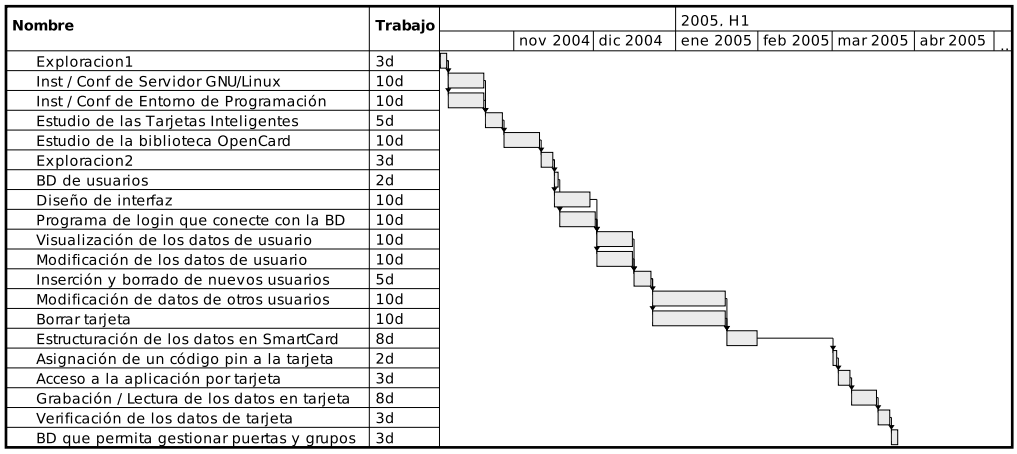
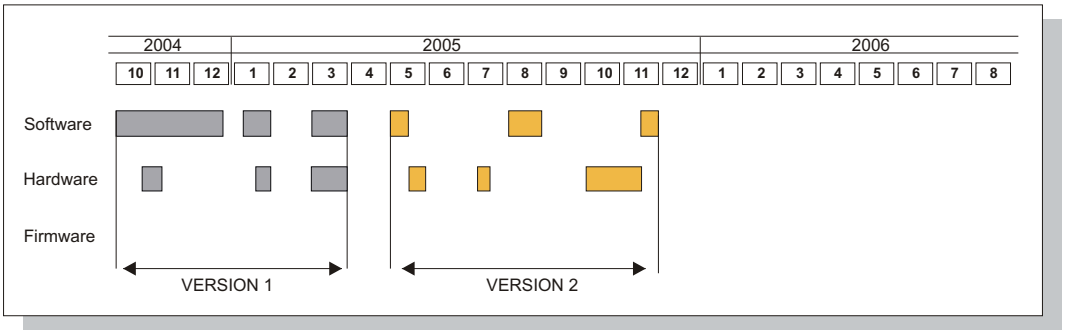


Figura 4.6: Diagrama de Gantt de la Version 1

4.3 Versión 2: Diseño e Implementación del Hardware



4.3.1 Exploración de la Versión 2

Duración: 5 semanas.

Tras la primera versión, se redefinen las historias sobrantes y se buscan nuevas para la segunda versión.
Para la versión 2, dado que la 1 ha servido para darle funcionalidades básicas al software, se cree oportuno que sirva para conseguir un hardware de mínima funcionalidad mientras se mejoran aspectos del

software.

Las historias 18, 19, 20, 21, 22 y 23 se mantienen:

Historia	Estimación
Creación de un módulo que permita crear y borrar puertas y grupos (historia 18)	3
Gestionar Permisos (historia 19)	3
Gestionar fechas de caducidad (historia 20)	2
Permitir hacer fotos con una cámara web (historia 21)	3
Fabricación del dispositivo con conexión (historia 22)	?
Fabricación del dispositivo sin conexión (historia 23)	?
Búsqueda de usuarios	2
Lectura de datos de usuario por tarjeta	2
Inserción de calendarios	1
Mejorar la apariencia de la aplicación	2

Los puntos de fijación en esta exploración son los siguientes:

- Grabación y lectura de datos personales en la tarjeta
  - Punto de fijación: Estudio de Java Multimedia Framework.
    Duración: 2 semanas.
- Fabricación de dispositivos.
  - Punto de fijación: Investigación sobre diseño de placas de circuito impreso (PCBs) .
    Duración: 2 semanas.

Figura 4.7: Fotografía obtenida en el curso de Diseño y montaje de placas de circuito impreso en Julio de 2005

Llegados a este punto hay que señalar que la investigación autodidacta sobre diseño de placas de circuito impreso no resulta ser tan satisfactoria como cabe esperar y, dada la cercanía de los exámenes de Junio, el equipo del proyecto decide continuar con la exploración tras la realización de un curso de una semana de duración sobre dicho tema. El curso recibido es el de '**Diseño y montaje de placas de circuito impreso**', impartido por el Departamento de Electrónica y Tecnología de Computadores de la Universidad de Granada del día 6 al 10 de Julio de 2005. Ver figura 4.7

Durante el curso los integrantes del grupo aprenden los conceptos prácticos y teóricos del diseño de PCBs, como el diseño de esquemáticos, el uso de diferentes herramientas software que facilitan el diseño de placas complejas y los procesos químicos y mecánicos que intervienen en la fabricación de las Placas de Circuito Impreso.

No obstante, aunque los conocimientos adquiridos fueron útiles para la realización del primer prototipo, dada la duración limitada del curso, sólo fue posible aprender la realización de placas sencillas a nivel "básico", y fue necesario un aprendizaje posterior, estudio de herramientas mas complejas y análisis de los métodos de fabricación profesionales para obtener el resultado final.

Gracias a los conocimientos adquiridos se pueden hacer nuevas estimaciones sobre el trabajo que implica la fabricación de los dispositivos.

Se añade un nuevo punto de fijación:

- Fabricación de dispositivos.
  - Punto de fijación: Estudio de diseños previos con el PIC16F876. **Duración:** 1 semana.

Además, se definen las siguientes historias nuevas:

- La historia 'Fabricación del dispositivo sin conexión' se intercambia por las historias:
  - Diseño del esquemático de la puerta aislada
  - Diseño de la PCB de la puerta aislada
  - Fabricación del primer prototipo de la puerta aislada
  - Pruebas de aceptación de la PCB de la puerta aislada
- La historia 'Fabricación del dispositivo con conexión' se intercambia por las historias:
  - Diseño del esquemático de la puerta inteligente
  - Diseño de la PCB de la puerta inteligente
  - Fabricación del primer prototipo de la puerta inteligente
  - Pruebas de aceptación de la PCB de la puerta inteligente

4.3.2 Planificación de la Versión 2

Las historias de que se dispone actualmente son ordenadas por importancia:

Historia 18

Creación de un módulo que permita crear y borrar puertas y grupos

estimación: 1

Historia 19

Gestionar permisos

estimación: 3

Historia 20

Gestionar fechas de caducidad

estimación: 2

Historia 24

Diseño del esquemático de la puerta aislada

estimación: 1

Historia 25

Diseño de la PCB de la puerta aislada

estimación: 2

Historia 26

Fabricación del primer prototipo de la puerta aislada

estimación: 1

Historia 27

Pruebas de aceptación de la PCB de la puerta aislada

estimación: 1

Historia 28

Diseño del esquemático de la puerta inteligente

estimación: 1

Historia 29

Diseño de la PCB de la puerta inteligente

estimación: 2

Historia 30

Fabricación del primer prototipo de la puerta inteligente

estimación: 1

Historia 31

Pruebas de aceptación de la PCB de la puerta inteligente

estimación: 1

Historia 21

Permitir hacer fotos con una cámara web

estimación: 3

Historia 32

Busqueda de usuarios

estimación: 2

Historia 33

Lectura de datos de usuario por tarjeta

estimación: 2

Historia 34

Inserción de calendarios

estimación: 1

Historia 35

Mejorar la apariencia de la aplicación

estimación: 2

Para la segunda versión se seleccionan por orden aquellas que en suma requieran un máximo de 24 unidades de estimación.

Nº	Historia	Est.	Iteración.
18	**Creación de un módulo que permita crear y borrar puertas y grupos**	1	2.1
19	**Gestionar permisos**	3	2.1
20	**Gestionar fechas de caducidad**	2	2.1
21	**Permitir hacer fotos con una cámara web**	3	2.2
33	**Lectura de datos de usuario por tarjeta**	2	2.2
34	**Inserción de calendarios**	1	2.2
24	**Diseño del esquemático de la puerta aislada**	1	2.3
28	**Diseño del esquemático de la puerta inteligente**	1	2.3
25	**Diseño de la PCB de la puerta aislada**	2	2.3
29	**Diseño de la PCB de la puerta inteligente**	2	2.3
26	**Fabricación del primer prototipo de la puerta aislada**	1	2.4
30	**Fabricación del primer prototipo de la puerta inteligente**	1	2.4
27	**Pruebas de aceptación de la PCB de la puerta aislada**	1	2.4
31	**Pruebas de aceptación de la PCB de la puerta inteligente**	1	2.4

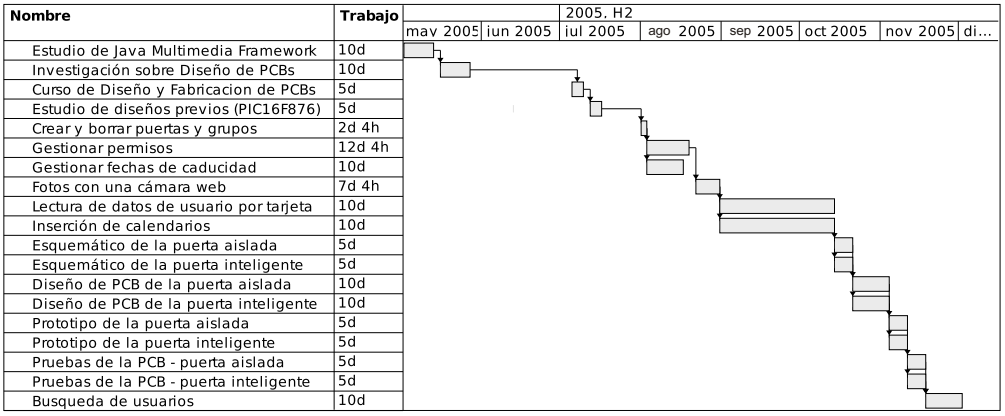


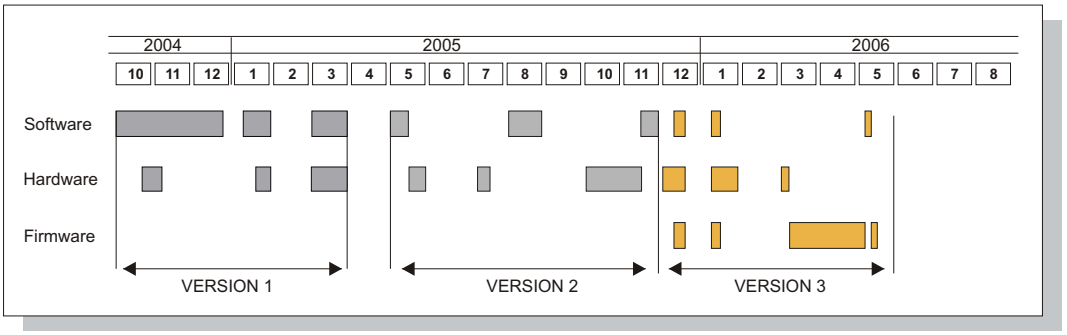
Figura 4.8: Diagrama de Gantt de la Version 2

32	**Busqueda de usuarios**	2	2.3
		24	

El Diagrama de Gantt del tiempo empleado para las historias de la segunda versión es el correspondiente a la figura 4.8.

En este gráfico se puede observar que la fabricación y diseño de las placas se realiza simultáneamente para las dos puertas. Las pruebas de aceptación de las placas se dejan para la última iteración con el fin de que sirva además como punto de partida de la siguiente versión: conocer si ha ocurrido algún error durante la fabricación de las placas permite que nuevas historias para nuevos diseños puedan ser incluidas en la siguiente versión.

4.4 Versión 3: Diseño e Implementación del Firmware



4.4.1 Exploración de la Versión 3

Duración: 5 semanas.

La primera versión del proyecto permitió obtener un software bastante funcional, la segunda permitió diseñar los primeros prototipos hardware, realizar numerosas pruebas sobre estos para detectar errores y dotar de mayor funcionalidad al software. Para ésta resta obtener prototipos finales de los dispositivos, y escribir el código fuente del firmware de cada uno de ellos.

Utilizando los test de placas de la versión anterior como puntos de fijación de la actual, tras descubrir diversos fallos de diseño y fabricación se hace necesario solventarlos. Para ello se decide recurrir a un nuevo software de diseño de PCB y un nuevo método de fabricación menos artesanal.

Para la versión 3, como se presenta lógico, se introducen historias para la revisión del hardware y para el diseño del firmware, de forma que el sistema comience a funcionar de manera íntegra.

Historia	Estimación
Rediseño de la PCB de la puerta aislada	2

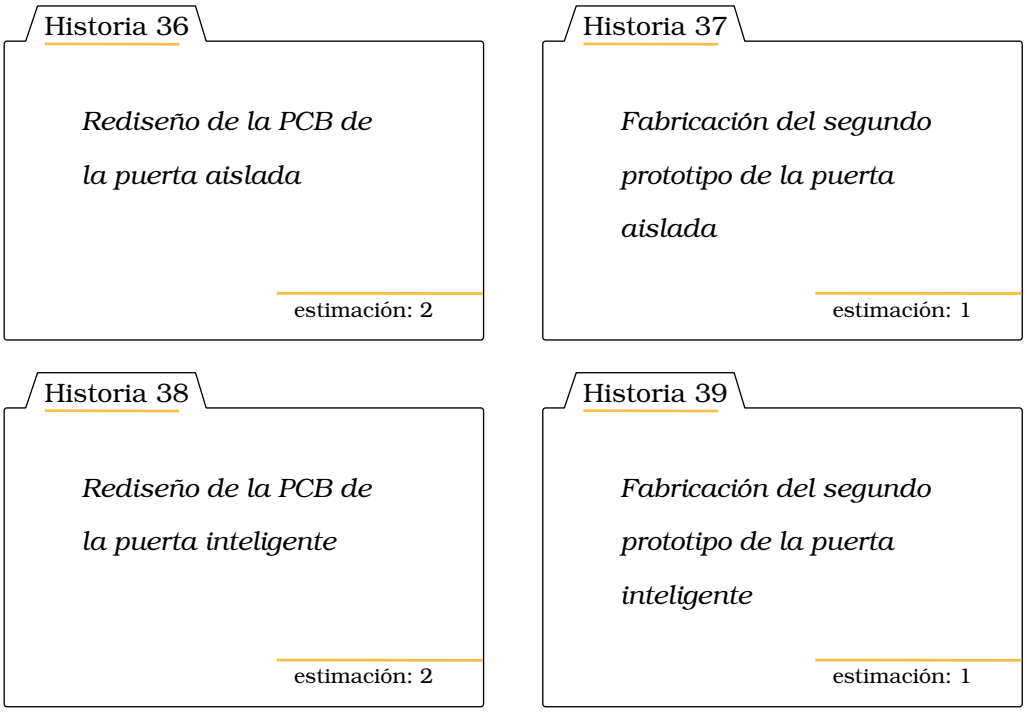
Fabricación del segundo prototipo de la puerta aislada	1
Pruebas de aceptación de la PCB de la puerta aislada	1
Rediseño de la PCB de la puerta inteligente	2
Fabricación del segundo prototipo de la puerta inteligente	1
Pruebas de aceptación de la PCB de la puerta inteligente	1
Creación de biblioteca para lector de tarjetas en la puerta inteligente	2
Creación de biblioteca para lector de tarjetas en la puerta aislada	2
Creación de biblioteca para display alfanumérico en la puerta inteligente	2
Creación de biblioteca para display alfanumérico en la puerta aislada	2
Creación de biblioteca para teclado de 16x16 en la puerta inteligente	2
Creación de biblioteca para teclado de 16x16 en la puerta aislada	2
Mejorar la apariencia de la aplicación	2
Implementación del algoritmo de encriptación para el Panel de Gestión	1
Implementación del algoritmo de encriptación para la puerta aislada	1
Implementación del algoritmo de encriptación para la puerta inteligente	1

Los puntos de fijación en esta exploración son los siguientes:

- Diseño de PCB
  - Punto de fijación: Estudio de software PADS.
    Duración: 2 semanas.
- Encriptación de datos.
  - Punto de fijación: Investigación sobre el algoritmo más adecuado.
    Duración: 1 semana.
  - Punto de fijación: Estudio del funcionamiento del algoritmo TEA.
    Duración: 1 semana.

4.4.2 Planificación de la Versión 3

Las historias de que se dispone actualmente son ordenadas por importancia:



Historia 40

Pruebas de aceptación de la PCB de la puerta aislada

estimación: 1

Historia 41

Pruebas de aceptación de la PCB de la puerta inteligente

estimación: 1

Historia 42

Creación de biblioteca para lector de tarjetas en la puerta aislada

estimación: 1

Historia 43

Creación de biblioteca para lector de tarjetas en la puerta inteligente

estimación: 1

Historia 44

Creación de biblioteca para display alfanumérico en la puerta aislada

estimación: 1

Historia 45

Creación de biblioteca para display alfanumérico en la puerta inteligente

estimación: 1

Historia 46

Creación de biblioteca para teclado de 16x16 en la puerta aislada

estimación: 1

Historia 47

Creación de biblioteca para teclado de 16x16 en la puerta inteligente

estimación: 1

Historia 48

Implementación del algoritmo de encriptación para el Panel de Gestión

estimación: 1

Historia 49

Implementación del algoritmo de encriptación para la puerta aislada

estimación: 1

Historia 50

Implementación del algoritmo de encriptación para la puerta inteligente

estimación: 1

Historia 35

Mejorar la apariencia de la aplicación

estimación: 2

Nº	Historia	Est.	Iteración.
36	**Rediseño de la PCB de la puerta aislada**	2	3.1
38	**Rediseño de la PCB de la puerta inteligente**	2	3.1
37	**Fabricación del segundo prototipo de la puerta aislada**	1	3.1
39	**Fabricación del segundo prototipo de la puerta inteligente**	1	3.1
40	**Pruebas de aceptación de la PCB de la puerta aislada**	1	3.2
41	**Pruebas de aceptación de la PCB de la puerta inteligente**	1	3.2

42	**Creación de biblioteca para Lector de Tarjetas en la puerta aislada**	1	3.2
43	**Creación de biblioteca para Lector de Tarjetas en la puerta inteligente**	1	3.2
44	**Creación de biblioteca para Display Alfanumérico en la puerta aislada**	1	3.2
45	**Creación de biblioteca para Display Alfanumérico en la puerta inteligente**	1	3.3
46	**Creación de biblioteca para Teclado de 16x16 en la puerta aislada**	1	3.3
47	**Creación de biblioteca para Teclado de 16x16 en la puerta inteligente**	1	3.3
48	**Implementación del algoritmo de encriptación para el Panel de Gestión**	1	3.3
49	**Implementación del algoritmo de encriptación para la puerta aislada**	1	3.3
50	**Implementación del algoritmo de encriptación para la puerta inteligente**	1	3.3
35	**Mejorar la apariencia de la aplicación**	2	3.4
		19	

El Diagrama de Gantt del tiempo empleado para las historias de la tercera versión es el correspondiente a la figura 4.9.

4.5 Pruebas

Uno de los pilares de la Programación Extrema (XP) es el proceso de pruebas. XP anima a probar constantemente tanto como sea posible.

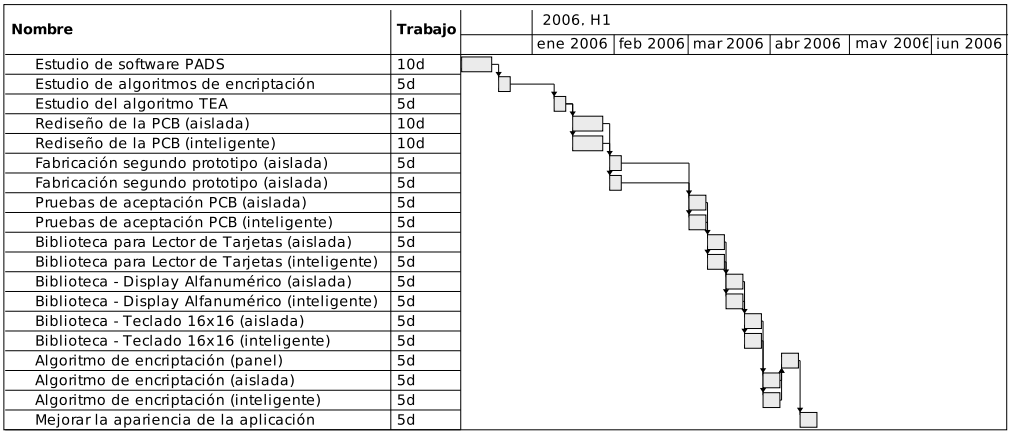


Figura 4.9: Diagrama de Gantt de la Version 3

Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. También permite aumentar la seguridad de evitar efectos colaterales no deseados a la hora de realizar modificaciones y refactorizaciones.

XP divide las pruebas del sistema en dos grupos:

Pruebas unitarias encargadas de verificar el código y diseñada por los programadores

Pruebas de aceptación o pruebas funcionales destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida diseñadas por el cliente final.

Las pruebas del sistema tienen como objetivo verificar la funcionalidad del sistema a través de sus interfaces externas comprobando que dicha funcionalidad sea la esperada en función de los requisitos del sistema. Generalmente las pruebas del sistema son desarrolladas por los programadores para verificar que su sistema se comporta de la manera esperada, por lo que podrían encajar dentro de la definición de pruebas unitarias que propone XP. Sin embargo, las pruebas del sistema tienen como objetivo verificar que el sistema cumple los requisitos establecidos por el usuario por lo que también pueden encajar dentro de la categoría de pruebas de aceptación.

En este proyecto, las pruebas unitarias han sido diseñadas y realizadas exclusivamente por los autores del proyecto, mientras que para las pruebas de aceptación, se ha contado con la colaboración de los tutores y personas externas al desarrollo del proyecto.

4.5.1 Espeficiación de requisitos y pruebas en Programación Extrema

El objetivo de las pruebas del sistema es verificar los requisitos. Por este motivo, los propios requisitos del sistema son la principal fuente de información a la hora de construir pruebas del sistema.

En procesos de desarrollo no extremos, como el Proceso Unificado, la funcionalidad del sistema se recoge en los requisitos funcionales, expresados principalmente como casos de uso. Estos casos de uso son completados con plantillas de texto que describen la secuencia principal o escenario de éxito, secuencias alternativas, tratamiento a los posibles errores, precondiciones, poscondiciones e invariantes. Además, el uso de plantillas permite recoger información adicional, si se estima conveniente, como la frecuencia de uso o la prioridad del caso de uso. Toda esta información facilita la generación de casos de prueba. Conociendo el camino principal, los caminos alternativos, errores posibles y su tratamiento y precondiciones, poscondiciones e invariantes es posible explorar exhaustivamente todas las combinaciones posibles asegurando que, en todo momento, el sistema responde de acuerdo a sus requisitos.

Sin embargo, toda esa información sobre los requisitos del sistema no está disponible en XP. En un desarrollo XP, los requisitos, es decir lo que el sistema debe hacer, se recogen en las historias durante el juego de la planificación. Una historia solo recoge una descripción en lenguaje natural, en general ambigua, incompleta e informal, de un fragmento de la funcionalidad del sistema, Aunque las tarjetas de historias pueden ser completadas con notas, el tamaño recomendado para la tarjeta no permite añadir información clara ni exhaustiva. Además,

por estos motivos, es imposible automatizar la generación de pruebas del sistema, siendo posible solo la automatización de su ejecución con las herramientas adecuadas.

La solución a este problema es la misma solución adoptada a la hora de implementar los requisitos. Debido a la escasez de información sobre el comportamiento esperado del sistema se hace imprescindible contar con la colaboración continua del cliente para el desarrollo de las pruebas del sistema. Dado que este, con toda seguridad no tiene conocimientos de ingeniería de software ni de pruebas del software, es difícil que puedan diseñar conjuntos de pruebas que tengan la cobertura adecuada. Algunas técnicas clásicas, como el análisis de posibles caminos, valores límite o categorías equivalentes son imprescindibles para garantizar que se verifica adecuadamente la implementación de los requisitos.

4.5.2 Pruebas de aceptación

Las pruebas de aceptación son una parte integral del desarrollo de un proyecto en XP. Todas las historias de usuario, se apoyan en las pruebas de aceptación, que son definidas por el cliente. Estas pruebas ayudan a eliminar malentendidos que se hayan podido producir durante la fase de especifación de requisitos. Las pruebas de aceptación son más importantes que las pruebas unitarias dado que implican la satisfacción del cliente con el producto desarrollado y el final de una iteración.

Las pruebas de aceptación obligan al cliente a profundizar en el dominio del conocimiento y como debería comportarse la aplicación en circunstancias específicas, por esto, el cliente es la persona adecuada para diseñar las pruebas de aceptación. Sin embargo esto supone el grave problema de que el cliente no tiene que tener, y en general no tiene, la formación adecuada para desarrollar buenas pruebas de aceptación. Por ejemplo, el cliente, en la mayoría de los casos sabe que es lo que quiere que la aplicación haga correctamente, pero puede

no ser capaz de desarrollar un conjunto de pruebas que garantice la total cobertura de la funcionalidad especificada en la historia de uso, limitándose a probar que el sistema hace lo que debe sin verificar todas las variantes que pueden aparecer.

Existen en la actualidad muchas herramientas, para desarrollar pruebas de aceptación, lo suficientemente sencillas para que un cliente pueda manejarlas. Sin embargo estas herramientas son inútiles si el cliente es incapaz de diseñar un conjunto completo de pruebas que verifiquen toda la funcionalidad del sistema y no solo una parte o con unos valores concretos.

Los tutores del proyecto juegan el papel de clientes y por lo tanto, a la hora de definir todas las historias son ellos los que han impuesto los requisitos de la aplicación y los que han aceptado o rechazado cada una de las iteraciones de cada versión.

5 Software de Gestión

Una vez especificado el diseño de todo el sistema, se comienza el desarrollo del software de gestión que debe cumplir los siguientes requisitos:

Multiusuario El sistema se comporta de manera diferente en función del tipo de usuario que lo esté utilizando.

Acceso concurrente Se permite el acceso simultáneo de varios usuarios. Esto se consigue gracias a la arquitectura Cliente-Servidor, y a la atomicidad de las transacciones que asegura el Sistema de Gestión de Base de Datos.

Multiplaforma El Software de gestión se ha programado en JAVA, utilizando bibliotecas compatibles.

Lecura y escritura de las SmartCards Se realiza a través de la biblioteca OpenCard Framework.

Gestionar Usuarios, Accesos y Permisos La aplicación podrá controlar la información de los usuarios del sistema, los accesos y los permisos, que se encuentra en la Base de datos. El acceso se realizará a través de JDBC.

Imprimir la SmartCard El software es capaz de capturar la fotografía de un usuario mediante una webcam e imprimir una tarjeta con una impresora especial.

En este apartado se describen los detalles acerca de la implementación.

Diagrama Entidad-Relación	El diagrama entidad-relación (a veces denominado por su siglas, E-R) es una herramienta para el modelado de datos de un sistema de información. Este diagrama expresa entidades relevantes para un sistema de información, sus interrelaciones y propiedades.

5.1 Diseño de la Base de Datos

Parte de la información que ha de manejar el sistema reside en una Base de Datos centralizada, que sirve información a los programas de gestión y a las puertas inteligentes concurrentemente. Esta Base de Datos almacena elementos como usuarios, puertas, grupos de puertas y permisos. Su diseño debe de ser correcto y su estructura de tablas debe de ser congruente con los requerimientos de almacenamiento de las tarjetas inteligentes. En los siguientes apartados se aborda su diseño conceptual.

5.1.1 Diseño conceptual

Para describir los conceptos que serán manejados por la base de Datos, serán realizados un Diagrama Entidad-Relación y un Diccionario de Datos. A continuación se muestran solamente los documentos finales, pertenecientes a la última versión del proyecto, cuando alcanza su máxima complejidad.

Diagrama Entidad Relación

El diagrama E-R de la base de datos de SARIC corresponde al mostrado en la figura 5.1.

Diccionario de Datos

A continuación se describen detalladamente todos los elementos visibles en el diagrama anterior mediante una tabla. Los campos de la

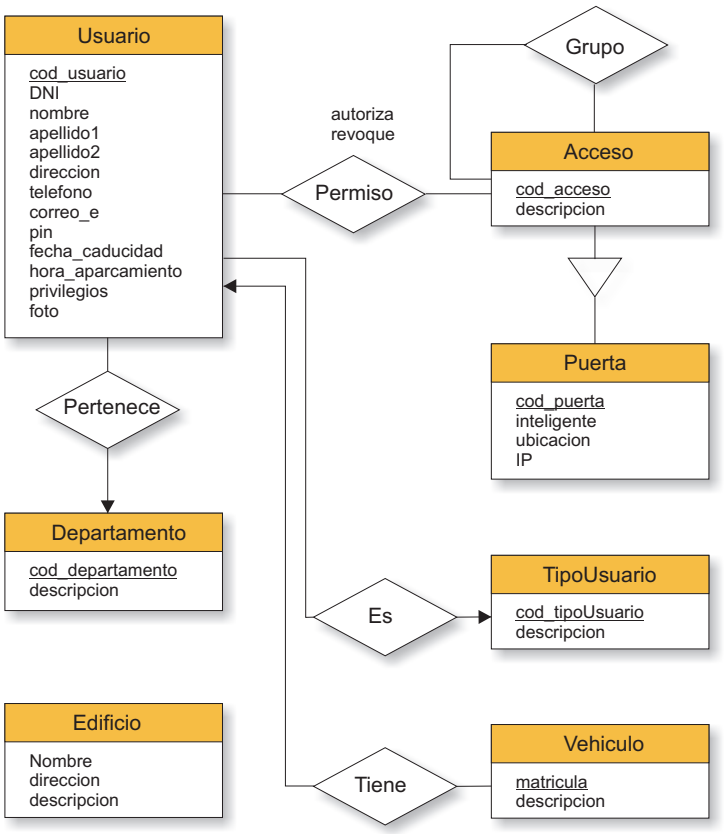


Figura 5.1: Diagrama Entidad-Relación de SARIC

tabla se interpretan de la siguiente forma:

- Concepto:Nombre del dato.
- Descripción:Breve explicación del dato.
- Tipo:Tipo de dato y representación interna en la Base de Datos.
- Dominio:Rango de valores que puede tomar el dato.
- CK:Indica si el dato es una llave candidata o no. CK son las siglas del termino en inglés “Candidate Key”.
- Nulo:Indica si el dato puede o no ser nulo.

Concepto	Definición	Tipo	Dom.	CK	Nulo
Usuario	Entidad que representa a un usuario del sistema				
cod-usuario	Código único que designa a cada usuario	entero	[0-99999]	✓	
DNI	DNI del usuario	entero	[9 digitos]		✓
nombre	Nombre del usuario	cadena	tam¡50		
apellido1	Primer apellido del usuario	cadena	tam≤50		
apellido2	Segundo apellido del usuario	cadena	tam≤50		
dirección	Dirección del usuario	cadena	tam≤200		✓
teléfono	Teléfono del usuario	cadena	tam≤10		✓
correo-e	Dirección del correo electrónico del usuario	cadena	tam≤50		✓
pin	Código pin de la tarjeta del usuario	entero	tam=4		
fecha-caducidad	Fecha de caducidad de la tarjeta del usuario	fecha			
hora-aparcamiento	Hora en que dejó el coche en el aparcamiento	fecha			✓

privilegios	Privilegios del usuario	entero	[1-3]		
foto	Foto del usuario	binario			✓
Departamento	Entidad que representa al Departamento al que pertenece el usuario				
cod-departamento	Código único que designa a cada departamento	entero	[0-999]	✓	
descripción	breve texto descriptivo del departamento	cadena	tam≤200		
Edificio	Entidad que representa al Departamento al que pertenece el usuario				
nombre	Nombre del edificio donde se ha implantado SARIC	cadena	tam≤100	✓	
dirección	Dirección del edificio	cadena	tam≤200		
descripción	Breve texto descriptivo del edificio	cadena	tam≤200		
Acceso	Entidad que representa un acceso genérico del sistema (puerta o grupo de puertas)				
cod-acceso	Código único que designa a cada acceso	entero	[0-65535]	✓	
descripcion	Designación o descripción del acceso	cadena	tam≤200		
Puerta	Entidad que representa una puerta gestionada por SARIC				
cod-pueta	Código único que designa a cada puerta	entero	[0-65535]	✓	
inteligente	campo que indica si la puerta es inteligente o no	booleano			

ubicación	lugar donde se encuentra la puerta	cadena	tam≤200		
IP	numero que identifica a cada puerta dentro de la red local	entero	tam≤12		✓
Tipo Usuario	Entidad que representa un tipo de usuario				
cod_tipoUsuario	Código único que identifica cada tipo de usuario	entero	[0-65535]	✓	
descripción	Breve texto descriptivo del tipo de usuario	cadena	tam≤100		
Vehículo	Entidad que representa un vehiculo				
matricula	Matrícula del vehículo que es única	cadena	tam≤10	✓	
descripción	Breve texto descriptivo del vehículo	cadena	tam≤100		✓

Las relaciones entre las entidades se describen en la tabla siguiente con el formato descrito a continuación:

- Relación:Nombre de la relación.
- Deficinión: Breve explicación de la relación.
- Entidad 1: Primera entidad involucrada en la relación.
- Entidad 2: Segunda entidad involucrada en la relación.
- Cardinalidad: Indica la cardinalidad de la relación: Uno a muchos, muchos a uno o muchos a muchos.

Relación	Definición	Entidad 1	Entidad 2	Cardinalidad
Pertenece	Relación que indica la pertenencia de un usuario a un departamento	Usuario	Departamento	* a 1
Grupo	Relación que indica que un acceso pertenece a un grupo de accesos	Acceso	Acceso	* a *

Permiso	Relación que indica que un usuario posee un determinado permiso para un acceso	Usuario	Acceso	* a *
Es	Relación que indica que un usuario es de cierto tipo	Usuario	Tipo_usuario	* a 1
Tiene	Relación que indica que un vehículo pertenece a un determinado usuario	Usuario	Vehículo	1 a *

5.2 Diagrama de clases

En la figura 5.2 se muestra la estructura general del proyecto en un diagrama de paquetes. Cada paquete corresponde a un submodelo (subsistema) del modelo (sistema). Estos son:

interfaz: Paneles de la interfaz y componentes de la aplicación no estándares de JAVA

mysql: Clases de acceso a datos y entidades de negocio

util: Clases con funciones de propósito general: conversión de tipos, encriptación de datos, etc.

webcam: Clases para el acceso a la webcam

smartcard: Clases para el uso de OCF y captura de interrupciones de la smartCard.

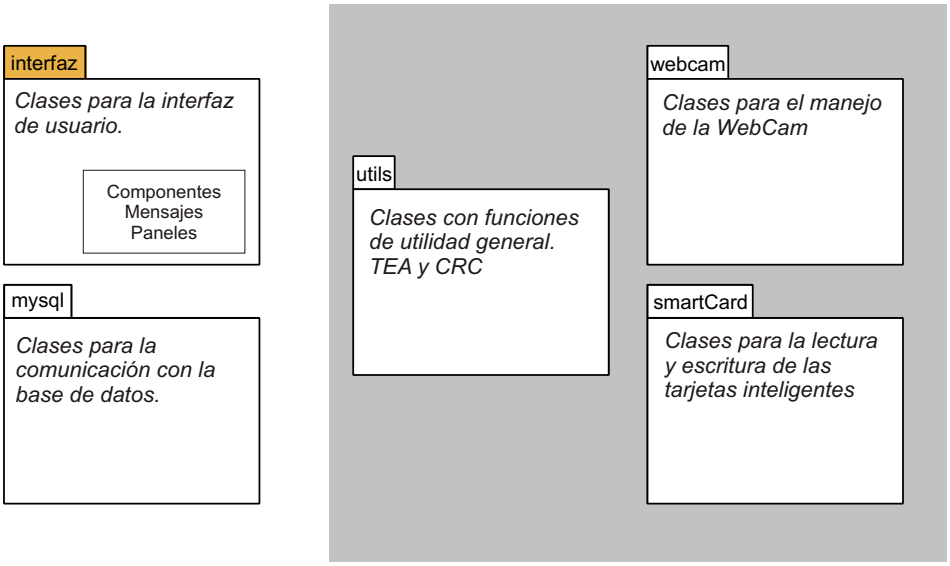


Figura 5.2: Diagrama de paquetes del proyecto

Cumpliendo con el desarrollo en capas de la arquitectura cliente servidor, las clases relativas a la interfaz de usuario se agrupan en el paquete "interfaz". Este a su vez se divide en dos subpaquetes: componentes y paneles. Mediante el uso de Open Card Framework, los paneles se comunican con el lector de tarjetas y de forma similar, usando

Java Media Framework estos pueden comunicarse con la webcam. El intercambio de datos se realiza mediante objetos de tipo Usuario, Acceso o Permiso. Estos se encuentran en la capa justo inferior a la de interfaz y se comunican con la capa de datos para solicitar las transferencias de información con el servidor. La capa más inferior, es la que se encarga de *hablar* con la base de datos mediante JDBC.

Todo esto se encuentra resumido de forma visual en la figura 5.3

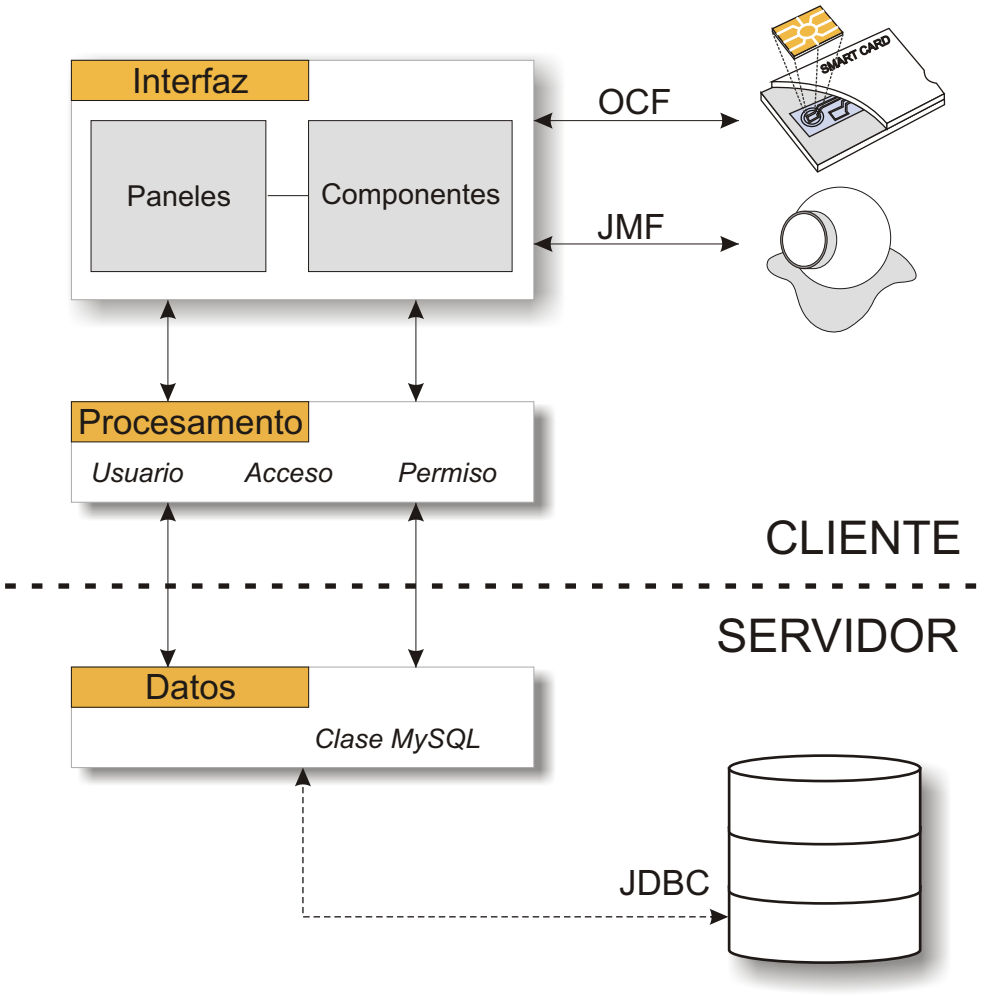


Figura 5.3: Arquitectura software detallada

A continuación se presentan diagramas de clases de los paquetes más interesantes del proyecto. En la figura 5.4 puede verse la jerarquía de los paneles de toda la aplicación. El panel denominado “PanelSa-

ric" se encarga de inicializar todos los demás. Este panel contiene el menú de selección, el banner decorativo de la parte superior de la aplicación y el resto de paneles organizados en forma de fichas de forma que solo uno de ellos es visible al mismo tiempo.

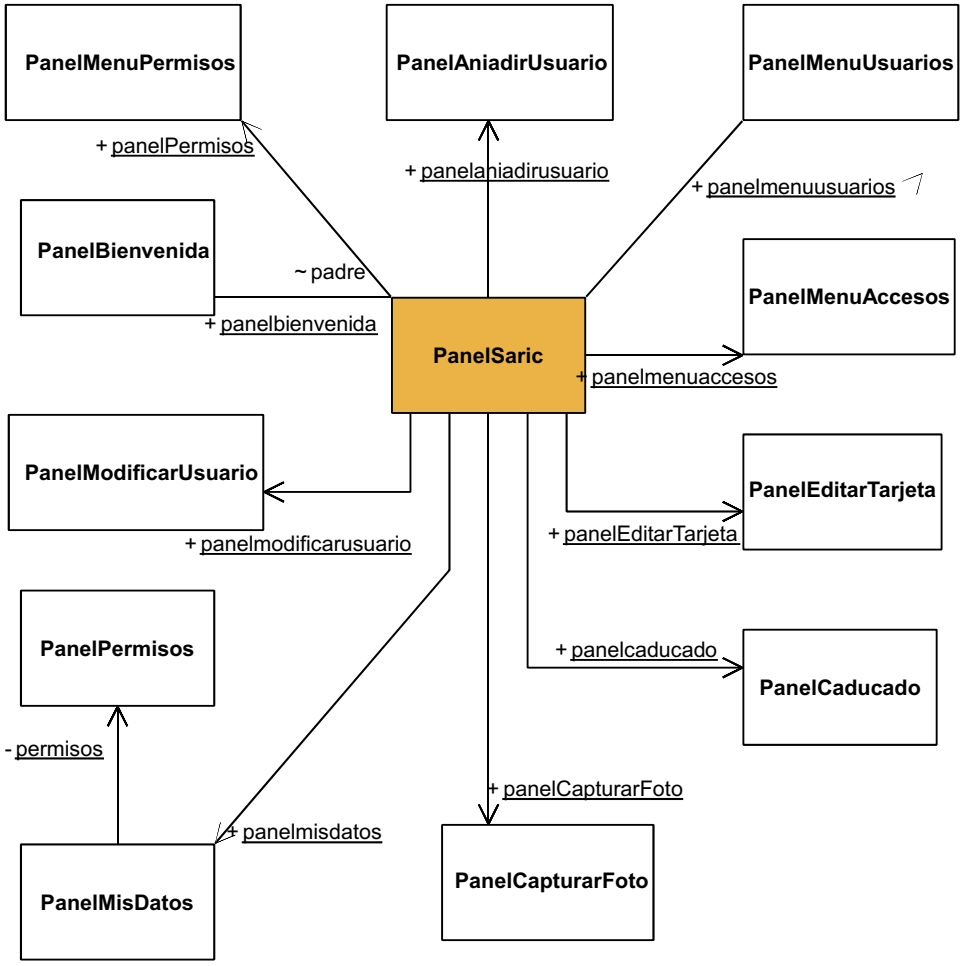


Figura 5.4: Diagrama de clases de los paneles del gestor de administración

Es también de interés el paquete "smartcard" (Ver figura 5.5). Para seguir con la filosofía de la programación orientada a eventos, se ha diseñado una clase "Evento Tarjeta" que se lanza cuando el usuario introduce o extrae la tarjeta. La clase "Hebra Tarjeta" se encarga de procesar estos eventos de forma asíncrona ayudada por el controlador: "ControladorTarjeta".

Conjuntamente con esta jerarquía de clases se diseñó un programa para realizar las pruebas. Este tras recibir una tarjeta lee su contenido y muestra su mapa de memoria.

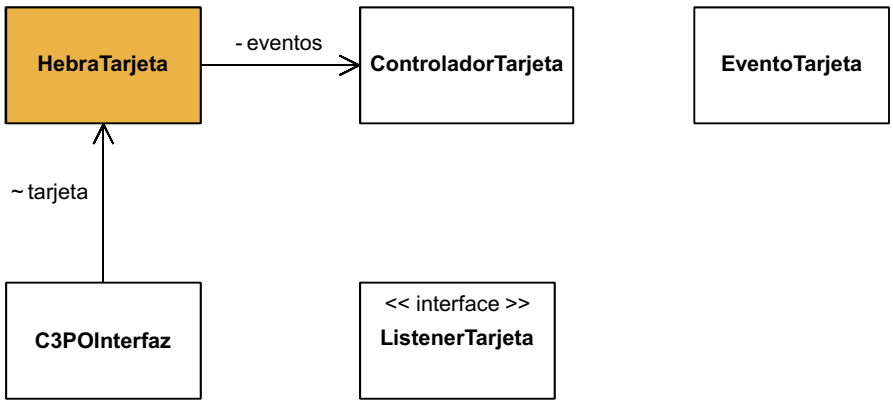


Figura 5.5: Diagrama de clases para las smartcards

6 Dispositivos Hardware

Una vez definidos los requisitos del sistema y la elaboración del diseño software, es necesario abordar con más detalle las características de los dispositivos hardware necesarios para el correcto funcionamiento del sistema. Hay que recordar que el objetivo es diseñar 2 dispositivos diferentes, intentado que el primero sea de muy bajo coste y el segundo de grandes capacidades de administración remota.

En la sección 1.3, puntos 3 y 4 ya se adelantaron las características mínimas que debían reunir los prototipos: ambos debían proporcionar un método de autentificación mediante un lector de tarjetas inteligentes, un teclado numérico, una interfaz sencilla usando una pantalla digital y por supuesto, el mecanismo de apertura de la puerta. Adicionalmente el segundo dispositivo debe permitir consultar su estado de forma remota mediante una interfaz *Ethernet*[1] y alterar su comportamiento.

Para facilitar la comprensión del texto, se denominará al prototipo de bajo coste "Puerta Aislada" dado que no posee la característica de conexión de Red. El dispositivo más avanzado se denominará "Puerta Inteligente" por su mayor capacidad de procesamiento y sus capacidades de Red.

Ya que ambos dispositivos son muy diferentes, se van a tratar por separado. El principal paso es la elección del procesador en cada uno

[1] *Ethernet*: Ver definición en página 9

DSP	DSP es el acrónimo de Digital Signal Processor, que significa Procesador Digital de Señal, un nombre bastante descriptivo, pues su función no es otra sino recibir una señal como entrada, hacer unas operaciones sobre ella y sacar a su salida una nueva. Un DSP es un sistema basado en un procesador o microprocesador que posee un juego de instrucciones, un hardware y un software optimizados para aplicaciones que requieran operaciones numéricas a muy alta velocidad. Debido a esto es especialmente útil para el procesado y representación de señales analógicas en tiempo real: en un sistema que trabaje de esta forma (tiempo real) se reciben samples (muestras), normalmente provenientes de un conversor analógico/digital (ADC), el sistema debe hacer todas las operaciones con el sample recibido antes de que llegue el siguiente.

Microprocesador	Un microprocesador es un conjunto de circuitos electrónicos altamente integrado para cálculo y control computacional. El microprocesador es utilizado como Unidad Central de Proceso en un sistema microordenador y en otros dispositivos electrónicos complejos como cámaras fotográficas e impresoras, y como añadido en pequeños aparatos extraíbles de otros aparatos más complejos como por ejemplo equipos musicales de automóviles.

de ellos. Los microcontroladores son los más utilizados en este tipo de dispositivos empotrados ya que no necesitan la potencia de cálculo de un microprocesador ni las características de procesado de señal de un DSP.

Actualmente es fácil encontrar microcontroladores en vehículos, teléfonos móviles o la mayoría de electrodomésticos presentes en cualquier hogar. La gama de microprocesadores disponible es enorme y dificulta bastante la elección del dispositivo correcto para cada tipo de

Microcontrolador	Circuito integrado o chip que incluye en su interior las tres unidades funcionales de un ordenador: CPU, Memoria y Unidades de E/S, es decir, se trata de un computador completo en un solo circuito integrado. Aunque sus prestaciones son limitadas, además de dicha integración, su característica principal es su alto nivel de especialización. Aunque los hay del tamaño de un sello de correos, lo normal es que sean incluso más pequeños, ya que, lógicamente, forman parte del dispositivo que controlan.

aplicación. Dado que uno de los requisitos del proyecto es usar la plataforma TINI para la Puerta Inteligente, solo es necesario elegir el microcontrolador adecuado para la Puerta Aislada. Para ello es necesario plantearse ciertas cuestiones y evaluar algunos factores que ayuden a tomar la decisión correcta.

A continuación se analizan dichos factores:

- Potencia requerida de cálculo (MIPS): No se necesita realizar ningún tipo de operaciones lo suficientemente complejas ni proceso de datos para tener que tomar este punto en consideración.
- Periféricos integrados: Sería interesante que el microcontrolador incorporase una interfaz *Ethernet*[2] en el caso de la Puerta Inteligente pero en la Puerta Aislada no se necesita ningún periférico adicional que pueda venir integrado en un microprocesador. Sí sería interesante que el microcontrolador disponga de facilidades para depurar la aplicación usando un emulador.
- Interconexión con periféricos/elementos externos: Este es un punto muy importante, es necesario saber cuantos dispositivos se van a conectar y asegurarse de que el microprocesador tiene suficientes puertos de Entrada/Salida. También influyen las características de los dispositivos, si son síncronos o asíncronos, digitales o

[2] *Ethernet*: Ver definición en página 9

analógicos... Los microprocesadores suelen tener diferentes versiones con distinto número de patillas y diferentes encapsulados, por lo tanto se tratará este punto una vez elegidos el microprocesador y la familia.

- Requerimientos de consumo: Energía, calor disipado... No se han hecho ningún tipo de requerimientos respecto a este punto por lo que no será tenido en cuenta.
- Posibilidad de actualización: Ya que se trata de un prototipo de pruebas, es necesario poder actualizar el firmware de forma sencilla y rápida.
- Integración y encapsulado: Para facilitar el montaje y ensamblado, dado que no se dispone de equipo profesional, en el proyecto usaremos encapsulado (Dual-In-Line) DIP (Ver figura 6.17.
- Time to Market: Hay microcontroladores que por disponer de mejor documentación o experiencia de los ingenieros en proyectos anteriores, pueden anteponerse a otro si el tiempo es un factor importante.

Teniendo en cuenta todos estos puntos, se ha optado por usar los microcontroladores de Microchip, concretamente los PICs de la familia 16xxx. Su bajo coste, la enorme documentación disponible en la red, la disponibilidad de diferentes encapsulados y modelos de diferente número de patillas y la experiencia en proyectos anteriores con este tipo de microcontroladores justifican sobradamente su elección.

6.1 Microcontrolador PIC16F876

El microcontrolador PIC16F876 de Microchip pertenece a una gran familia de microcontroladores de 8 bits (bus de datos) que tienen las siguientes características generales que los distinguen de otras familias:

- Arquitectura Harvard

MIPS

MIPS es el acrónimo de"millones de instrucciones por segundo". Es una forma de medir la potencia de los procesadores. Sin embargo, esta medida sólo es útil para comparar procesadores con el mismo juego de instrucciones y usando benchmarks que fueron compilados por el mismo compilador y con el mismo nivel de optimización. Esto es debido a que la misma tarea puede necesitar un número de instrucciones diferentes si los juegos de instrucciones también lo son; y por motivos similares en las otras dos situaciones descritas. En las comparativas, usualmente se representan los valores de pico, por lo que la medida no es del todo realista. La forma en que funciona la memoria que usa el procesador también es un factor clave para la potencia de un procesador, algo que no suele considerarse en los cálculos con MIPS. Debido a estos problemas, los investigadores han creado pruebas estandardizadas tales como SpecInt para medir el funcionamiento real, y las MIPS han caído en desuso.

- Repertorio de instrucciones RISC
- Tecnología CMOS

Estas características se conjugan para lograr un dispositivo altamente eficiente en el uso de la memoria de datos y programa y por lo tanto en la velocidad de ejecución.

Microchip ha dividido sus microcontroladores en tres grandes subfamilias de acuerdo al número de bits de su bus de instrucciones, en concreto en PIC16F876 se encuentra en una gama media con un bus de instrucciones de 14bits.

6.1.1 Principales características del PIC1F876

- CPU RISC
- Solo 35 instrucciones que aprender

Encapsulado	La comunicación de un microprocesador con el exterior, esto es, con la memoria principal y con las unidades de control de los periféricos, se realiza mediante señales de información y señales de control que son enviadas a través del patillaje del microprocesador. El número y tamaño de las patillas ha ido variando con el tiempo según las necesidades y las tecnologías utilizadas. Para comunicarse con el resto del sistema informático el procesador utiliza las líneas de comunicación a través de sus patillas (pines). Se define como encapsulado la forma en que se empaqueta la oblea de silicio para efectuar su conexión con el sistema.

RISC	En arquitectura de los computadores, RISC del inglés Reduced Instruction Set Computer (Computadora con Conjunto de Instrucciones Reducido) es un tipo de microprocesadores con las siguientes características fundamentales: ▪ Instrucciones de tamaño fijo y presentadas en un reducido número de formatos. ▪ Sólo las instrucciones de carga y almacenamiento acceden a la memoria por datos. Además estos procesadores suelen disponer de muchos registros de propósito general.

- Todas las instrucciones se ejecutan en un ciclo de reloj, excepto los saltos que requieren dos
- Frecuencia de operación de 0 a 20 MHz (DC a 200 nseg de ciclo de instrucción)
- Hasta 8k x 14 bits de memoria Flash de programa
- Hasta 368 bytes de memoria de datos (RAM)
- Hasta 256 bytes de memoria de datos EEPROM
- Hasta 4 fuentes de interrupción
- Pila de hardware de 8 niveles

CMOS	CMOS (inglés: Complementary Metal Oxide Semiconductor (Semiconductor complementario óxido - metal), "MOS Complementario") es una tecnología utilizada para crear circuitos integrados, como pueden ser puertas lógicas, contadores, etc. Consiste básicamente en dos transistores, uno PFET y otro NFET. De esta configuración resulta el nombre.

- Reset de encendido (POR)
- Temporizador de encendido (PWRT)
- Temporizador de arranque del oscilador (OST)
- Sistema de vigilancia Watchdog timer.
- Protección programable de código
- Modo SEP de bajo consumo de energía
- Opciones de selección del oscilador
- Programación y depuración serie In-Circuit (ICSP) a través de dos pines
- Lectura/escritura de la CPU a la memoria flash de programa
- Rango de voltaje de operación de 2.0 a 5.5 volts
- Alta disipación de corriente de la fuente: 25mA
- Rangos de temperatura: Comercial, Industrial y Extendido
- Bajo consumo de energía:

Periféricos

- Timer0: Contador/Temporizador de 8 bits con pre-escalador de 8bits
- Timer1: Contador/Temporizador de 16 bits con pre-escalador

EEPROM	EEPROM son las siglas de electrically-erasable programmable read-only memory (ROM programable y borrable eléctricamente), en español se suele referir al hablar como E-2-PROM y en inglés "E-Squared-PROM´´. Es un tipo de memoria ROM que puede ser programado, borrado y reprogramado eléctricamente, a diferencia de la EPROM que ha de borrarse mediante rayos ultravioletas. Aunque una EEPROM puede ser leída un número ilimitado de veces, sólo puede ser borrada y reprogramada entre 100.000 y 1.000.000 de veces.

- Timer0: Contador/Temporizador de 8 bits con pre-escalador y post-escalador de 8 bits y registro de periodo.
- Dos módulos de Captura, Comparación y PWM
- Convertidor Analógico/Digital: de 10 bits, hasta 8 canales
- Puerto Serie Síncrono (SSP)
- Puerto Serie Universal (USART/SCI).
- Puerto Paralelo Esclavo (PSP): de 8 bits con líneas de protocolo

6.1.2 Diagrama de bloques

En la figura 6.2 se muestra en un diagrama de bloques la organización interna del PIC16F876. Se muestra también junto a este diagrama su diagrama de patillas (Figura 6.1), para tener una visión conjunta del interior y exterior del chip.

6.1.3 Descripción de la CPU

La CPU es la responsable de la interpretación y ejecución de la información (instrucciones) guardada en la memoria de programa. Muchas de estas instrucciones pueden operan sobre la memoria de datos, y en el caso de realizar operaciones lógicas o aritméticas requieren el uso de la Unidad de Aritmético-Lógica (ALU); esta controla los bits de

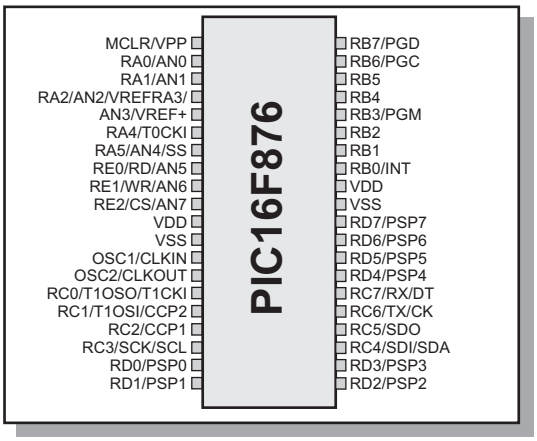


Figura 6.1: Diagrama de patillas del PIC16F876

ALU	Se denomina Unidad Aritmético-Lógica (UAL) o ALU (Arithmetic and logical unit) Físicamente, la ALU es parte de la altamente integrada lógica-electrónica del microprocesador principal de cualquier computadora.

estado (Registro STATUS) cuyos bits pueden alterarse en función del resultado de algunas operaciones.

Ciclo de Instrucción

El Contador de Programa (PC, del Inglés Program Counter) es un registro que controla el ciclo de instrucción como se muestra en la figura 6.3. En cada ciclo de instrucción la CPU lee (ciclo Fetch) la instrucción almacenada en la memoria de programa apuntada por PC y al mismo tiempo ejecuta la instrucción anterior, esto debido a una cola de instrucciones que le permite ejecutar una instrucción mientras lee la próxima. En Arquitectura de los Computadores esto se conoce como procesamiento segmentado.

Como puede verse, cada ciclo de instrucción (Tcy) se compone a su vez de cuatro ciclos del oscilador (Tosc). Cada ciclo Q proporciona la sincronización para los siguientes eventos:

Q1: Decodificación de la instrucción

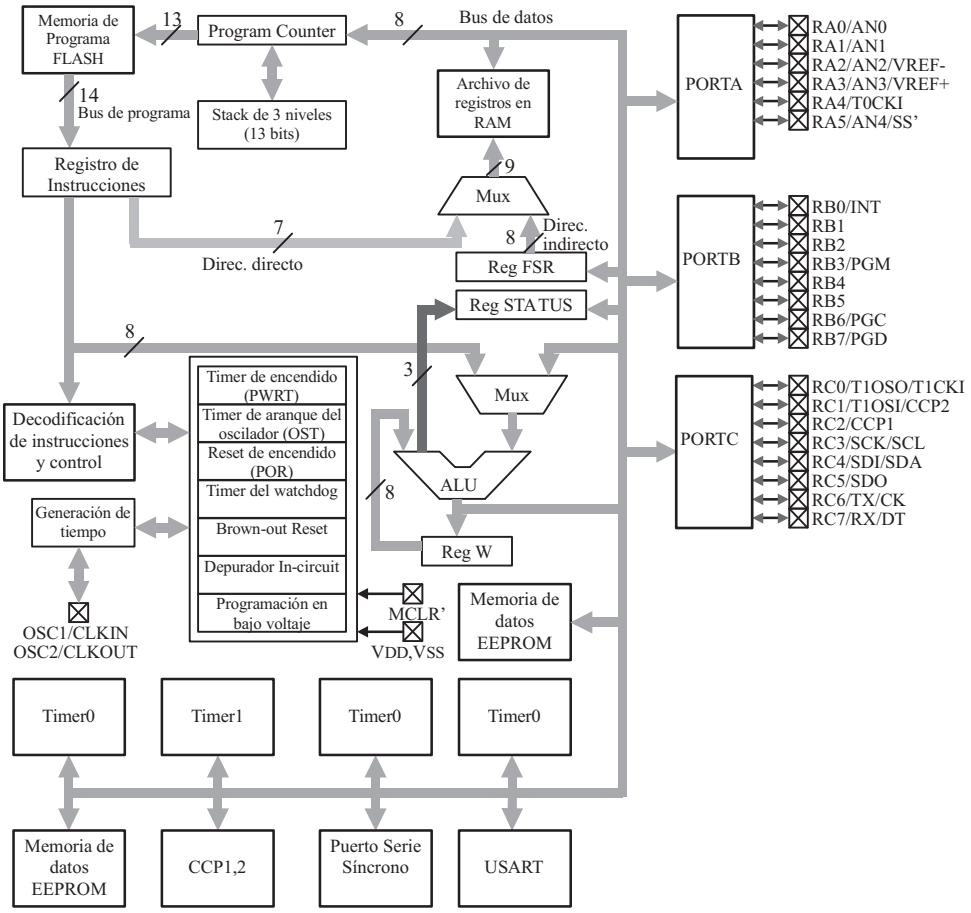


Figura 6.2: Diagrama de bloques del PIC16F876

Q2: Lectura del dato (si lo hay)

Q3: Procesa el dato

Q4: Escribe el dato

Debido a esto cada ciclo de instrucción consume 4 ciclos de reloj, de manera que si la frecuencia de oscilación es Fosc, Tcy será 4/Fosc.

Registros de la CPU

Registro PC Registro de 13 bits que siempre apunta a la siguiente instrucción a ejecutarse.

Registro de Instrucción Registro de 14 bits. Todas las instrucciones se colocan en él para ser decodificadas por la CPU antes de ejecutarlas.

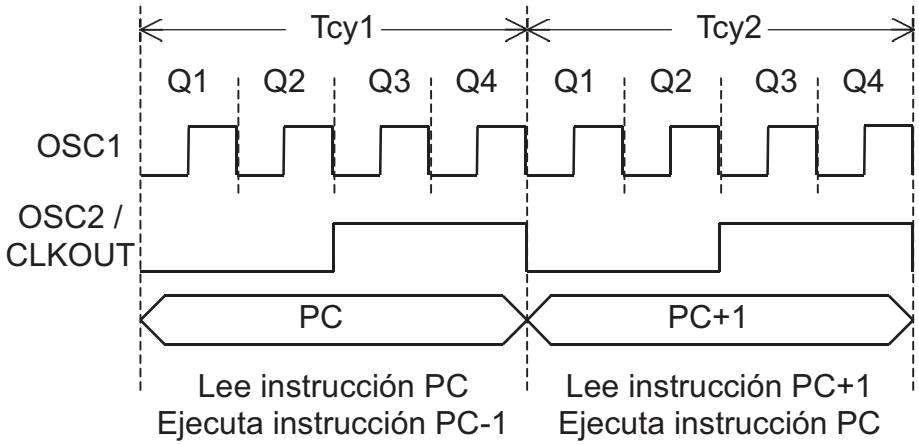


Figura 6.3: Ciclo de instrucción del PIC16F876

Registro W Registro de 8 bits que guarda resultados temporales de las operaciones realizadas por la ALU.

Registro STATUS Registro de 8 bits, cada uno de los cuales (denominados flags) es un indicador de estado de la CPU o del resultado de la última operación.

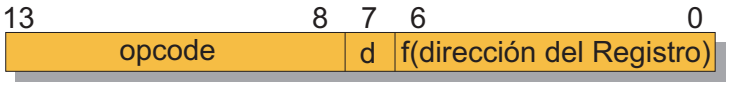
6.1.4 Repertorio de instrucciones

El repertorio de instrucciones del PIC16F876, consta de 35 instrucciones y todas ellas consumen 1 solo ciclo de reloj excepto las instrucciones de salto que necesitan 2 ciclos.

Formato general de las instrucciones

Cada instrucción en lenguaje de máquina (binario) del PIC contiene un código de operación (opcode) el cual puede ser de 3 a 4 o 6 bits, dependiendo del tipo de instrucción. A continuación se describe el formato para cada tipo de instrucción de los PIC de rango medio:

Operaciones con el archivo de registros orientadas a bytes:



El bit d especifica el destino del resultado de la operación:

d = 0: Destino W

d = 1: Destino f

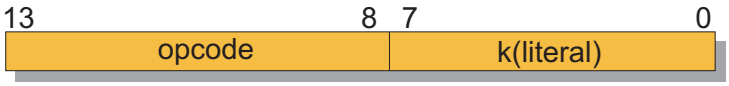
f: Dirección de 7 bits del archivo de registros]

Operaciones con el archivo de registros orientadas a bits:

b: Especificación en tres bits del bit sobre el que se va a operar 0..7

f: Dirección de 7 bits del archivo de registros]

Operaciones con literales y de control:
Formato general:



Formato para CALL y GOTO:

k (Literal): Valor de un operando de 8 bits

6.1.5 Organización de la memoria del PIC

Los PIC tienen dos tipos de memoria: Memoria de Datos y Memoria de programa (Arquitectura Hardvard), cada bloque con su propio bus:

Bus de datos y Bus de programa; por lo cual se puede acceder a cada bloque durante un mismo ciclo de oscilación.

La Memoria de datos a su vez se divide en:

- Memoria RAM de propósito general
- Archivo de Registros (Special Function Registers (SFR))

La Memoria de Programa

Los PIC de rango medio poseen un registro Contador del Programa (PC) de 13 bits, capaz de direccionar un espacio de 8K x 14, como todas la instrucciones son de 14 bits, esto significa un bloque de 8k instrucciones. El bloque total de 8K x 14 de memoria de programa está subdividido en 4 páginas de 2K x 14. En la siguiente figura 6.4 se muestra esta organización.

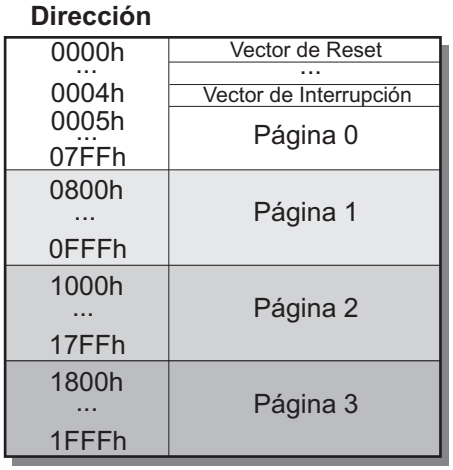


Figura 6.4: Organización de la memoria en el PIC16F876

Vector de Reset Cuando ocurre un reset el contenido del PC es forzado a cero, ésta es la dirección donde la ejecución del programa continuará después del reset, por ello se le llama“dirección del vector de reset”.

Vector de interrupción Cuando la CPU acepta una solicitud de interrupción ejecuta un salto a la dirección 0004h, por lo cual a esta

se le conoce como “dirección del vector de interrupción”.

La Memoria de Datos

La memoria de datos consta de dos áreas mezcladas y destinadas a funciones distintas:

- Registros de Propósito Especial (SFR)
- Registro de Propósito General (GPR)

Los SFR son localidades asociadas específicamente a los diferentes periféricos y funciones de configuración del PIC y tienen un nombre específico asociado con su función. Mientras que los GPR son memoria RAM de uso general.

Bancos de memoria

Toda la memoria de datos está organizada en 4 bancos numerados 0, 1, 2 y 3. Para seleccionar un banco se debe hacer uso de los bits del registro STATUS¡7:5¿denominados IRP, RP1 y RP0. Hay dos maneras de acceder a la memoria de datos: Direccionamiento directo e indirecto.

Cada banco consta de 128 bytes (de 00h a 7Fh). En las posiciones más bajas de cada banco se encuentran los SFR, y encima de éstos se encuentran los GPR. Toda la memoria de datos está implementada en RAM estática.

Memoria de Pila

La memoria de pila es una área de memoria completamente separada de la memoria de datos y la memoria de programa. La pila consta de 8 niveles de 13 bits cada uno. Esta memoria es usada por la CPU para almacenar las direcciones de retorno de subrutinas. El apuntador de pila no es ni legible ni escribible. Cuando se ejecuta una instrucción CALL o es reconocida una interrupción el PC es guardado en la pila y el apuntador de pila se incrementa en 1 para apuntar a la siguiente posición vacía. A la inversa, cuando se ejecuta una instrucción RETURN,

RETLW o RETFIE el contenido de la posición actual de la pila se coloca en el PC.

6.2 Plataforma Tiny InterNet Interfaces

Las primeras implementaciones de TINI datan de finales de 1998 cuando unos ingenieros de Dallas Semiconductor, trabajando con ingenieros de Sun Labs, mostraron un dispositivo muy pequeño y programable en JAVA que era capaz de controlar distintos aparatos electrónicos. Estos prototipos fueron empotrados en interruptores de luz, cafeteras, ventiladores y otros electrodomésticos, así estos eran capaces de comunicarse entre sí y con un servidor central mediante un sistema de Red primitivo.

La idea principal era no solo proporcionar control local de los dispositivos, si no conectividad en red permitiendo monitorización y control remotos. Esto incrementaba la flexibilidad y la facilidad de uso de los dispositivos. Aunque en la actual plataforma no quede nada de esta tecnología, si prevalecen los conceptos de esta: un entorno de ejecución programable en JAVA usado para crear aplicaciones empotradas en Red.

Las mejoras incluidas desde entonces pasan por el cambio a una interfaz de Red *Ethernet*[3] , una pila *TCP/IP*[4] y al aumento de las capacidades de E/S. Actualmente, TINI es una amplia plataforma que incluye hardware y software, y que es usada para diseñar dispositivos inteligentes de Red. Estos dispositivos suelen ser pequeños, de bajo consumo de energía y de coste reducido. Unos ejemplos podrían ser equipamiento industrial de automatización, maquinas dispensadoras, sensores remotos o control de accesos como es el caso de este proyecto.

[3] *Ethernet*: Ver definición en página 9
[4] *TCP/IP*: Ver definición en página 7

6.2.1 Descripción

Tiny InterNet Interface (TINI) es una plataforma desarrollada por Dallas Semiconductors para proveer a desarrolladores hardware y software de un medio simple, flexible y de bajo coste para diseñar una gran variedad de dispositivos hardware que puedan ser conectados directamente a redes corporativas o de usuario. La plataforma es una combinación de pequeños pero poderosos chipsets y un entorno de ejecución de JAVA. El chipset proporciona procesado, control, comunicación a nivel de dispositivos y un entorno de red.

Las capacidades del hardware subyacente, están disponibles para el desarrollador software mediante una serie de APIs en JAVA.

El objetivo principal del dispositivo es permitir conectar sensores, actuadores y toda clase de dispositivos a la red. La combinación de entradas y salidas, la pila *TCP/IP*[5] y el entorno de programación Java permite a los programadores crear rápida y sencillamente aplicaciones que permitan el control local y remoto del sistema.

6.2.2 El hardware de TINI

La implementación mínima de una plataforma TINI esta compuesta por:

- Microcontrolador
- Memoria Flash ROM
- Memoria RAM estática

El microcontrolador es la parte fundamental de TINI y se encarga de ejecutar el código nativo del entorno de ejecución JAVA. Actualmente se usa el microcontrolador DS80C390. Es pequeño e incluye soporte para distintos protocolos de E/S, entre ellos el serie o el cada vez más

[5] *TCP/IP*: Ver definición en página 7

usado CAN. Además posee varios puertos de propósito general que pueden ser usados para tareas simples de control como encendido de leds (Light Emiting Diode) o accionar relés.

La memoria flash almacena el entorno de ejecución y cumple los siguientes requisitos:

- El contenido de la memoria se mantiene incluso con la ausencia de energía.
- La memoria es reprogramable

La memoria RAM estática contiene los datos del sistema y la pila del recolector de basura donde se crean todos los objetos de JAVA. También almacena los datos del sistema de ficheros. La persistencia de los datos en esta memoria en ausencia de energía depende de si se incorpora una batería a ésta o no.

Es posible conectar otro tipo de periféricos que no sean memorias directamente al bus de datos y direcciones del microcontrolador. Dos muy comunes son la interfaz *Ethernet*[6] y un Reloj de Tiempo Real (RTC del inglés Real Time Clock).

En la figura 6.5 se presenta un diagrama de bloques de un dispositivo TINI a alto nivel.

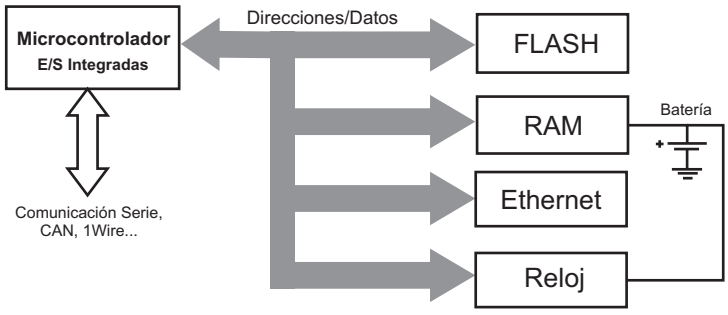


Figura 6.5: Diagrama de Bloques de un dispositivo TINI

[6] *Ethernet*: Ver definición en página 9

6.2.3 El mapa de memoria

Un mapa de memoria especifica donde se decodifican memoria y otros dispositivos periféricos en el espacio de direcciones del microcontrolador. El mapa de memoria usado por TINI se muestra en la figura 6.6 y consiste en los siguientes 3 segmentos diferentes:

- Código
- Datos
- Periféricos

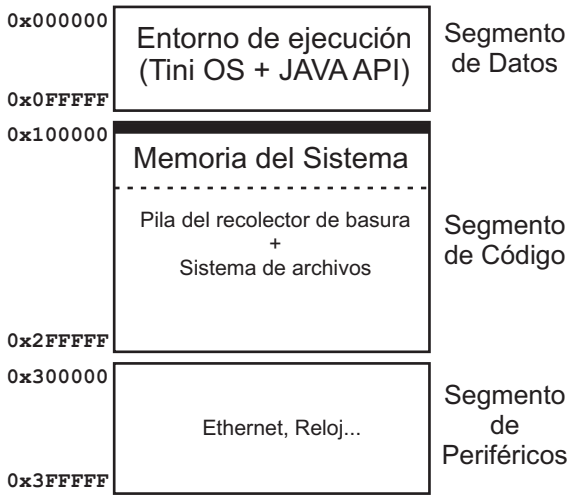


Figura 6.6: Mapa de memoria de TINI

Los tamaños de los segmentos mostrados en la figura son los máximos posibles (además, todos ellos múltiplos de 1 MegaByte). Si, por ejemplo, solo existen 512 Kb de Flash ROM en el segmento de código, la dirección de comienzo del segmento de datos se mantiene en 0x100000. En otras palabras, las direcciones de comienzo de los diferentes segmentos son siempre como se muestran en la figura 6.6
. Pero las direcciones finales pueden ser menores a las indicadas dependiendo de cuanto espacio este actualmente ocupado por los chips de memoria. Los requisitos mínimos de memoria para los segmentos de

código y datos son de 512 kilobytes cada uno.

Los segmentos de código y datos están ocupados por chip de memoria y el segmento de periféricos por otros tipos de componentes hardware como el controlador *Ethernet*[7] y el Reloj de tiempo real mostrados en la figura 6.6. Es posible mapear otros dispositivos periféricos que soporten una interfaz de bus paralelo compatible con el bus del microcontrolador en el segmento de periféricos.

El controlador y el reloj de tiempo real ocupan los siguientes rangos de direcciones:

- Ethernet controller - [0x300000 - 0x0x307FFF]
- Real-Time clock - 0x310000

Los diseñadores hardware deben evitar el uso de estos rangos para conectar cualquier dispositivo diferente de un controlador ethernet o un reloj de tiempo real.

También esta disponible un área de periféricos de 4-megabyte conocida como el espacio "Periferial Chip Enable (PCE)", que puede usarse para conectar chips de memoria externos (hasta de 4-megabytes) u otros dispositivos hardware directamente a los buses de memoria y datos del microcontrolador. Sin embargo, la mayoría del hardware se mapea en el segmento de periféricos mostrado en la figura 6.6 debido a que este permite el acceso de forma mucho más eficiente. El microcontrolador usa cuatro pines para controlar el espacio PCE. Si no se mapea ningún dispositivo en este espacio, estos pines pueden usarse como pines adicionales de propósito general.

6.2.4 E/S integradas

Los dispositivos periféricos descritos anteriormente están todos conectados a los buses de direcciones y datos del microcontrolador. Sin embargo, una amplia gama de dispositivos que son interesentes

[7] *Ethernet*: Ver definición en página 9

para trabajar con las capacidades de red de TINI no soportan la conexión a un bus de datos paralelo. Normalmente, estos dispositivos trabajan mediante interfaces serie. Esto por lo general implica una comunicación de menor ancho de banda. Pero una interfaz serie también reduce el número de pines requeridos, simplifica la comunicación, y normalmente tiene menores costes cuando se compara con dispositivos que poseen interfaces de tipo bus paralelo. Las interrupciones serie también tienen la ventaja de no añadir ninguna carga a los buses del microcontrolador. Se ha integrado soporte en el microcontrolador para los siguientes tipos de protocolos serie a bajo nivel:

Comunicación Serie: Soporta protocolos de serie síncronos, que usan una interfaz de 2 cables, y comunicación serie asíncrona basada en el estándar *RS232-C*[8] . El controlador de TINI proporciona 2 circuitos UART (Transmisor/Receptor Asíncrono Universal) para facilitar la comunicación en serie. Los puertos serie asíncronos son sumamente comunes en los dispositivos integrados.

Controller Area Network (CAN): Originalmente desarrollada por Bosch-Siemens, CAN se describe ahora en dos estándares ISO. Proporciona un bus de comunicación serie fiable que se usa de forma muy común en aplicaciones de control industrial y automoción. El microcontrolador de TINI incluye dos controladores integrados CAN.

Red 1-Wire: Desarrollada por Dallas Semiconductor, la red 1-Wire es una red de pequeños sensores, actuadores y elementos de memoria que comparten el mismo conductor para energía y comunicación.

E/S TTL: Estos pines de los puertos del microcontrolador, de propósito general y bidireccionales pueden usarse para diferentes tares de control y no están necesariamente atados a algún tipo de conexión serie.

Usando las capacidades integradas de E/S del microcontrolador

[8] *RS232*: Ver definición en página 8

en vez de E/S mapeadas en memoria, se reduce tanto el número total de dispositivos como el coste de comunicación con un dispositivo externo ya que carga menos la CPU. Por ejemplo, la CPU del núcleo del microcontrolador funciona a máxima velocidad, ejecutando el entorno de ejecución, mientras simultáneamente la UART está enviando y recibiendo caracteres serie. Las comunicaciones que usan el bus, requieren que la CPU se detenga, ejecute las instrucciones necesarias para la comunicación y luego reanude se curso normal de ejecución.

6.2.5 Un diseño de Referencia Hardware para TINI

No requerir de un único diseño hardware o un tamaño específico, provee a los diseñadores de la flexibilidad necesaria para diseñar el chipset TINI según los requisitos de sus productos. Pero sin una implementación de referencia concreta y comercialmente disponible de TINI, cada nuevo diseño tendría que empezar con el proceso bastante tedioso de diseñar y depurar nuevo hardware. Se ha desarrollado la placa TINI Modelo 390 (TBM390) (Ver figura 6.7 para resolver este problema. Permite tanto a los diseñadores hardware como a los programadores empezar el prototipado y desarrollo sin una gran inversión tanto de tiempo como de dinero.

El TBM390 cumple los siguientes propósitos:

Implementación de referencia: Todos los detalles de su diseño son públicos. Los desarrolladores hardware son libres de usar información obtenida del TBM390 a la hora de diseñar sus propios chipset de dispositivos basados en TINI.

Herramienta de desarrollo: Proporciona un método acceso sencillo a la mayoría de las E/S de la plataforma, permitiendo a los diseñadores conectar rápidamente hardware propio externo y desarrollar sus aplicaciones. El equipo de ingenieros de TINI también lo ha usado para desarrollar y probar el entorno de ejecución.

Componente del sistema: El TMB390 es un diseño que ha sido intensamente probado y testeado, y sus condiciones de operación

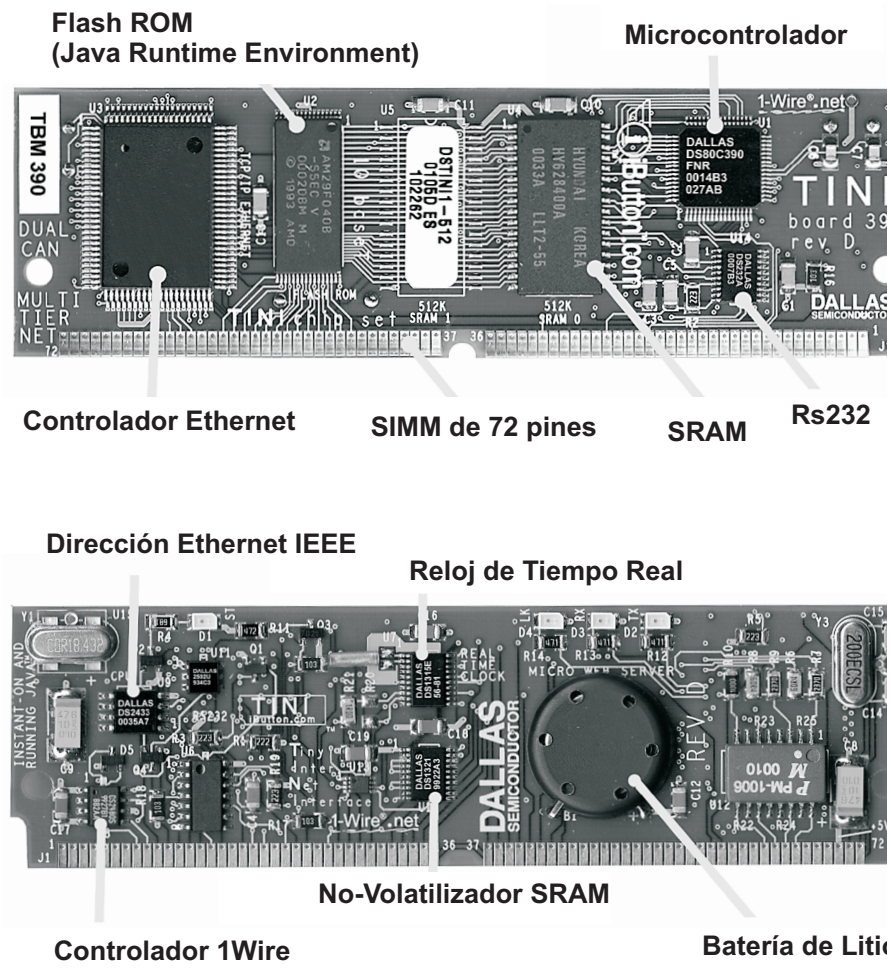


Figura 6.7: TBM390: vistas superior e inferior

de voltaje y temperatura han sido especificadas de forma clara. Estas características hacen que se ajuste perfectamente para su uso como componente principal en un sistema comercial o industrial de red.

El TBM390 es una placa de 72 pines compacta (31.8 mm x 102.9 mm). Es una implementación hardware con soporte para *Ethernet*[9] y incluye todas las funcionalidades descritas en la figura 6.5:

- 512 kilobytes de memoria flash para código del sistema crítico
- 512 kilobytes de memoria no volátil (persistente) SRAM, expandible a 1 Mb

[9] *Ethernet*: Ver definición en página 9

- Controlador Ethernet 10Base-T
- Reloj de tiempo real
- Interfaz de Red 1-Wire Dual
- Controladores CAN Duales
- Puertos serie Duales (uno con niveles *RS-232*[10] y otro con niveles de +5V)
- Puerto de serie síncrono de 2 cables
- Expone los buses de direcciones y datos del microcontrolador para expansiones de E/S
- Solo necesita de una fuente de energía de +5V

6.2.6 El entorno de ejecución

Proporcionar el hardware esencial para desarrollar dispositivos de red empotrados es solo la mitad del trabajo. Se necesita un gran cantidad de software para liberar a los desarrolladores de tener que preocuparse de la infraestructura necesaria para ejecutar múltiples tareas, protocolos de red y una interfaz de programación. Un entorno de ejecución bien definido que ofrezca todas estas características permite al desarrollador centrarse principalmente en los detalles de la aplicación. Por esta razón desde el principio se ha desarrollado un entorno de ejecución como parte integral de toda la plataforma.

El software que compone el entorno de ejecución de TINI puede dividirse en dos categorías: Código nativo ejecutado directamente por el microcontrolador, y una API interpretada como bytecodes por la Maquina Virtual de Java (JVM). El código de las aplicaciones se escribe en Java y utiliza la API para acceder a las capacidades de código nativo y los recursos del hardware subyacente. También es posible escribir librerías nativas para cumplir requisitos estrictos de tiempo real. En la

[10] *RS-232*: Ver definición en página 8

API	Una API (del inglés Application Programming Interface - Interfaz de Programación de Aplicaciones) es un conjunto de especificaciones de comunicación entre componentes software. Representa un método para conseguir abstracción en la programación, generalmente (aunque no necesariamente) entre los niveles o capas inferiores y los superiores del software. Uno de los principales propósitos de una API consiste en proporcionar un conjunto de funciones de uso general, por ejemplo, para dibujar ventanas o iconos en la pantalla. De esta forma, los programadores se benefician de las ventajas de la API haciendo uso de su funcionalidad, evitándose el trabajo de programar todo desde el principio. Las APIs asimismo son abstractas: el software que proporciona una cierta API generalmente es llamado la implementación de esa API.

figura 6.8 se muestra una representación gráfica del entorno de ejecución.

Los programas Java que se ejecutan en TINI son aplicaciones y no Applets. Son programas autónomos que comienzan su ejecución con un método main con la siguiente estructura:

```
public static void main(String[] args)
```

Además, a diferencia de los applets, no tienen ningún tipo de restricciones de "caja de arena" como las máquinas virtuales disponibles en los PCs. En TINI, las aplicaciones Java tienen todos los privilegios y acceso a todos los recursos del sistema, incluso más que otras plataformas que incluyen soporte para un entorno de ejecución de Java. Esto es particularmente importante para aplicaciones empotradas porque estas están estrechamente relacionadas con dispositivos físicos. También, a diferencia de otras plataformas Java, en TINI no hay un administrador del sistema que se encargue de la configuración y el mantenimiento. Esto significa que la aplicación es completamente responsable de configurar además de controlar el sistema al completo. Por

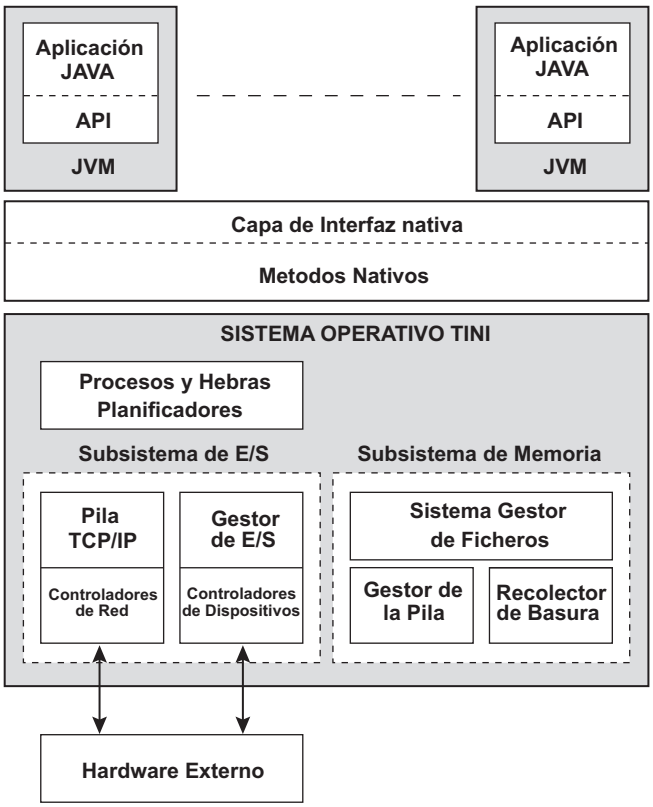


Figura 6.8: Entorno de ejecución de TINI

estas razones una aplicación que controle un sistema empotrado debe tener acceso completo incluso a funciones de bajo nivel proporcionadas por el sistema operativo.

6.3 Proceso de diseño y fabricación de los prototipos

Existen diferentes herramientas en el mercado que facilitan el diseño y la producción de sistemas electrónicos desde Placas de Circuito Impreso (*PCBs*[11]) a circuitos integrados, estos se conocen como programas de diseño electrónico automatizado (EDA por sus siglas en inglés), para distribuir e interconectar los componentes. Estos programas almacenan información relacionada con el diseño, facilitan la edición, y pueden también automatizar tareas repetitivas. Para el diseño del primer prototipo de ambas placas se usó el software P-CAD, pero

[11] *PCB*: Ver definición en página 24

la ausencia de herramientas de verificación adecuadas y de control de separación entre las pistas nos llevó a cometer varios errores que subsanamos en el segundo prototipo. Para este se usó el software PADS, mucho más complejo que P-CAD pero a la vez muchísimo más completo y usable.

A continuación se hace un análisis de los diferentes procesos de diseño y fabricación que se han aplicado en el proyecto una vez diseñados los esquemáticos de los circuitos.

6.3.1 Diseño

La primera etapa es convertir el esquemático en una lista de nodos (o "net list" en inglés). La lista de nodos es una lista de los pines y nodos del circuito, a los que se conectan las pines de los componentes. Normalmente el programa de captura de esquemáticos, utilizado por el diseñador del circuito, es responsable de la generación de la lista de nodos, y esta lista es posteriormente importada en el programa de ruteo.

En la figura 6.9 se muestra un fragmento de la lista de nodos de la puerta inteligente generada por PADS.

El siguiente paso es determinar la posición de cada componente. Normalmente los programas incluyen rutinas de posicionamiento automático o por cluster donde el diseñador puede asistir a estas especificando ciertas zonas de la tarjeta, donde determinados grupos de componentes deben ir. Por ejemplo, a las partes asociadas con el subcircuito de la fuente de alimentación se le podría asignar una zona cercana a la entrada al conector de alimentación. En otros casos, los componentes pueden ser posicionados manualmente, ya sea para optimizar el desempeño del circuito, o para poner componentes tales como interruptores y conectores, según lo requiera el diseño mecánico del sistema.

En la figura 6.10 se muestra una captura del programa PADS con

```
1  *NET*
2  *SIGNAL* VCC
3  J2.8 J2.1 J2.6 S1.2 R4.1
4  R1.A C1.PLUS J1.VDD U4.18 U3.8
5  B1.POS LED1.A U1.VCC4 U1.VCC3 U1.VCC1
6  U1.VCC2 R5.1 J1.BL U1.VPP U8.14
7  U2.14 U7.20 U9.20
8  *SIGNAL* GND
9  C3.MINUS C2.MINUS U4.9 U3.10 U3.4
10 U1.GND1 U1.GND2 U1.USB1 U1.GND3 U10.1
11 U6.G C1.MINUS J1.R/˜W J1.GND J1.BLG
12 B1.NEG R3.B J2.5 R2.1 U8.7
13 U2.7 U7.10 U9.10
14 *SIGNAL*           WR
15 U2.9 U1.W̃R U8.12
16 *SIGNAL* D2
17 U7.13 U9.4 R5.4 J1.D2 U1.D2
18 *SIGNAL* D1
19 U7.12 U9.3 R5.3 J1.D1 U1.D1
20 *SIGNAL* D5
```

Figura 6.9: Fragmento de la lista de nodos de la puerta inteligente

pin	Un pin es una patilla para conexiones eléctricas

los componentes de la puerta inteligente posicionados de forma manual. Aunque el programa dispone de opciones para agrupaciones de componentes mediante clusters, dado el bajo número de estos no merece la pena usarlas ya que se obtiene un diseño más eficiente colocándolos de forma manual.

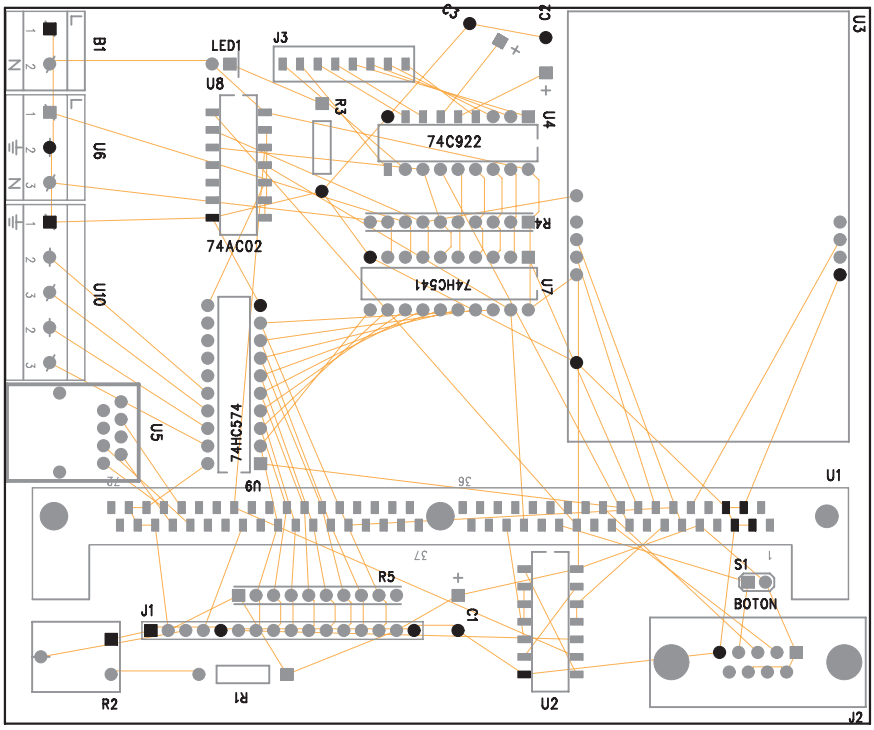


Figura 6.10: Componentes de la puerta inteligente en PADS

El software luego expande la lista de componentes en una lista completa de pines para la tarjeta, utilizando plantillas de una biblioteca de footprints asociados a cada tipo de componentes. La biblioteca permite que los footprints sean dibujados sólo una vez, y luego compartidos por todos los dispositivos de ese tipo. En la figura 6.11 se muestra un footprint de un componente genérico que se adapta a cualquier circuito integrado DIP de 18 patillas.

pad	Un pad es una superficie plana de cobre usada para hacer conexiones eléctricas. Opcionalmente pueden ser taladrados para introducir un pin antes de realizar la soldadura

vía	Una vía es un orificio practicado en la placa para permitir realizar una conexión eléctrica entre 2 pistas situadas en diferentes capas

En algunos sistemas, los pads de alta corriente son identificados en la biblioteca de dispositivos, y los nodos asociados son etiquetados para llamar la atención del diseñador del circuito impreso. Las corrientes elevadas requieren de pistas más anchas, y el diseñador normalmente determina este ancho.

Luego el programa combina la lista de nodos (ordenada por el nombre de los pines) con la lista de pines (también ordenada por nombre), transfiriendo las coordenadas físicas de la lista de pines a la lista de nodos. La lista de nodos es luego reordenada, por el nombre del nodo.

Algunos programas pueden optimizar el diseño al intercambiar la posición de las partes y puertas lógicas para reducir la longitud de las pistas de cobre. Otros también detectan automáticamente los pines de alimentación de los dispositivos, y generan pistas o vías conectadas al plano de alimentación o conductor más cercano.

Luego el programa trata de encaminar cada nodo en la lista de señales-pines, encontrando secuencias de conexión en las capas disponibles. A menudo algunas capas son asignadas a la alimentación y a la masa, y se conocen como plano de alimentación y masa respectiva-

Footprint	Un footprint (huella) es un mapa de los pines de un dispositivo, usualmente con la distribución de los pad y perforaciones recomendadas.

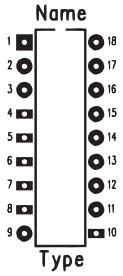


Figura 6.11: Footprint de un DIP de 18 pines

mente. Estos planos ayudan a blindar los circuitos del ruido.

El problema de ruteo es equivalente al problema del viajante de comercio, y es por lo tanto NP-completo, y es demasiado costoso la búsqueda de una solución perfecta. Un algoritmo práctico de ruteo es elegir el pin más lejano del centro de la tarjeta, y luego usar un algoritmo greedy para seleccionar el siguiente pin más cercano con la señal del mismo nombre.

Después del encaminamiento automático, suele quedar una lista de nodos que deben ser ruteados manualmente.

Una vez ruteado, el sistema puede disponer de un conjunto de estrategias para reducir el costo de producción del circuito impreso. Por ejemplo, una rutina podría eliminar las vías innecesarias (cada vía es una perforación, que cuesta dinero). Otras podrían redondear los bordes de las pistas, y ensanchar o mover las pistas para mantener el espacio entre éstas dentro de un margen seguro. Otra estrategia podría ser ajustar grandes áreas de cobre de tal forma que ellas formen nodos, o juntar áreas vacías en áreas de cobre. Esto permite reducir la contaminación de los productos químicos utilizados durante el grabado y acelerar la velocidad de producción.

En la figura 6.12 se muestra el resultado final de la puerta inteligente.

Algunos sistemas tienen comprobación de reglas de diseño para

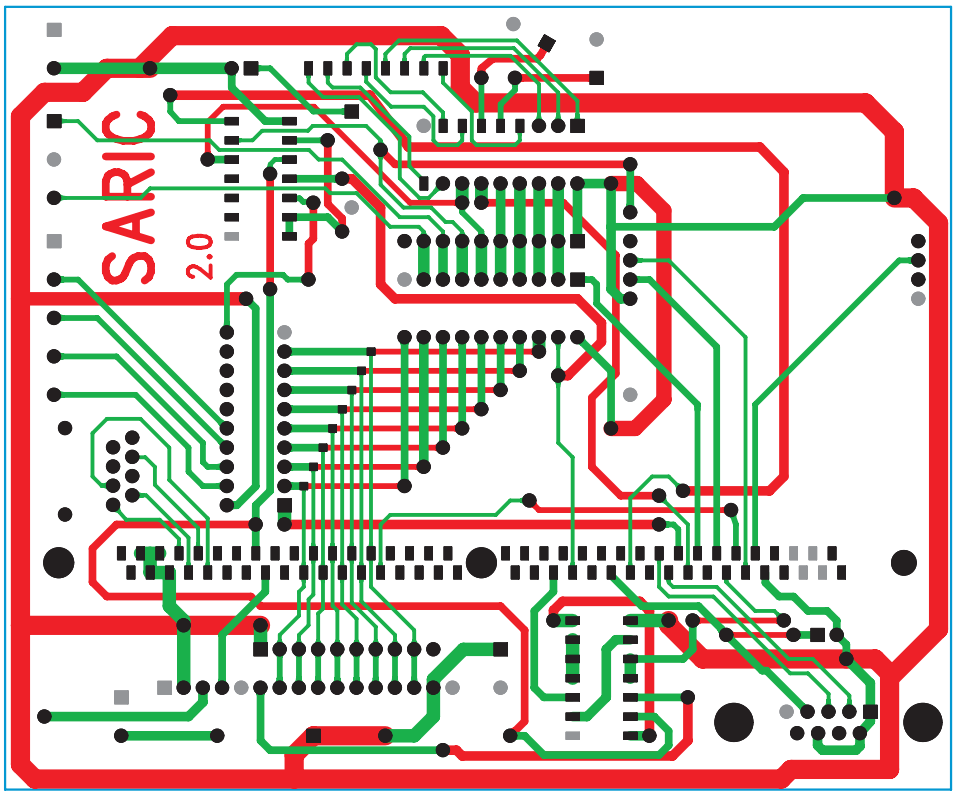


Figura 6.12: Enrutado de todas las pistas de la puerta inteligente

validar la conectividad eléctrica y separación entre las distintas partes, compatibilidad electromagnética, reglas para la fabricación, ensamblaje y prueba de las tarjetas, flujo de calor y otro tipo de errores. En PADS es posible validar el diseño especificando unas reglas determinadas, y así, por ejemplo, verificar si alguna pista esta demasiado cerca de otra o es demasiado fina para la frecuencia de reloj requerida. **El diseño de ambas puertas pasó todas las pruebas de forma satisfactoria**.

La serigrafía, máscara antisoldante y plantilla para la pasta de soldar, a menudo se diseñan como capas auxiliares (Soldering Mask Layer).

6.3.2 Fabricación

La gran mayoría de las tarjetas para circuitos impresos se hacen adhiriendo una capa de cobre sobre todo el sustrato, a veces a ambos lados (creando un circuito impreso virgen), y luego eliminando el cobre no deseado tras aplicar una máscara temporal (por ejemplo, grabándola con percloruro férrico), dejando sólo las pistas de cobre deseado. Algunos circuitos impresos se fabrican agregando el cobre al sustrato dibujando directamente las pistas a través de un proceso complejo de electrorecubrimiento múltiple. Algunos circuitos impresos tienen capas con pistas en el interior de éste, y son llamados circuitos impresos multicapas. Éstos se construyen agrupando tarjetas delgadas que son procesadas de forma separada. Después de que la tarjeta haya sido fabricada, los componentes electrónicos se sueldan a ésta.

Hay varios métodos típicos para la producción de circuitos impresos:

1. La impresión serigráfica utiliza tintas resistentes al grabado para proteger la capa de cobre. Los grabados posteriores eliminan el cobre no deseado. Alternativamente, la tinta puede ser conductora, y se imprime en una tarjeta virgen no conductora. Esta última técnica también se utiliza en la fabricación de circuitos híbridos.

2. El fotograbado utiliza un *Fotolito*[12] y grabado químico para eliminar la capa de cobre del sustrato. El fotolito normalmente se prepara con un fotoplotter, a partir de los datos producidos por un programa para el diseño de circuitos impresos. Es posible usar transparencias impresas en una impresora Láser como fotolitos de baja resolución.

3. El fresado de circuitos impresos utiliza una fresa mecánica de 2 o 3 ejes para eliminar el cobre del sustrato. Una fresa para circuitos impresos funciona en forma similar a un plotter, recibiendo comandos desde un programa que controla el cabezal de la fresa los ejes **x**, **y** y **z**. Los datos para controlar la máquina son generados por el programa de diseño, y son almacenados en un archivo en formato HPGL o Gerber.

4. La impresión en material termosensible para transferir las pistas a la placa de cobre, mediante calor. En este proceso se puede utilizar papel glossy (fotográfico) o papel con cera.

Los primeros prototipos de ambas puertas se han realizado utilizando la técnica de fotograbado, en la figura 6.13 se puede observar la insoladora del departamento de Electrónica y Tecnología de los Computadores utilizada en la fabricación de los prototipos de ambas placas.

El segundo prototipo ha realizado utilizando la técnica de fresado, más precisa que la anterior pero que necesita disponer de acceso a una máquina fresadora.

Tanto el recubrimiento con tinta, como el fotograbado requieren de un proceso de atacado químico, en el que se elimina el cobre excedente, quedando únicamente el patrón deseado.

Atacado

El atacado de la placa virgen se puede realizar de diferentes maneras. La mayoría de los procesos utilizan ácidos o corrosivos para eliminar el cobre excedente. Existen métodos de galvanoplastia que funcionan de manera rápida, pero con el inconveniente de que es necesario

[12] *Fotolito*: Ver definición en página **??**

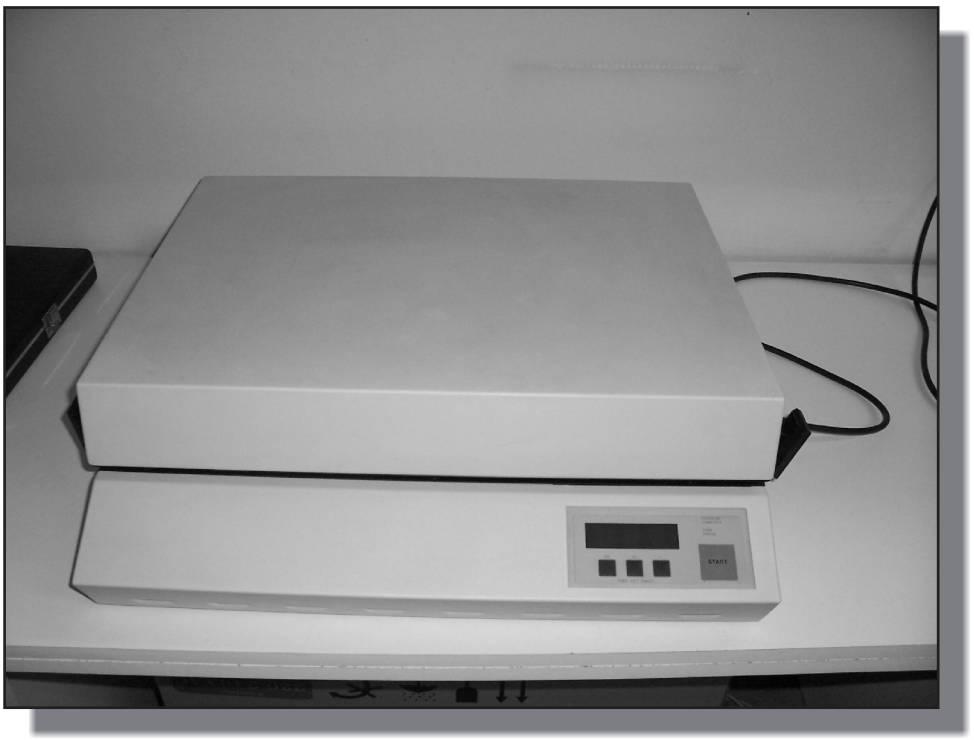


Figura 6.13: Insoladora del Dpto de ETC de la UGR

atacar al ácido la placa después del galvanizado, ya que no se elimina todo el cobre.

Los químicos más utilizados son el Cloruro Férrico, el Sulfuro de Amonio, el Ácido clorhídrico mezclado con Agua y Peróxido de hidrógeno. Existen fórmulas de ataque de tipo alcalino y de tipo ácido. Según el tipo de circuito a fabricar, se considera más conveniente un tipo de formulación u otro.

Para la fabricación industrial de circuitos impresos es conveniente utilizar máquinas con transporte de rodillos y cámaras de aspersión de los líquidos de ataque, que cuentan con control de temperatura, de presión y de velocidad de transporte. También es necesario que cuenten con extracción y lavado de gases .

En la figura 6.14 se muestra el ácido y el resultado de la placa una vez finalizado el proceso.

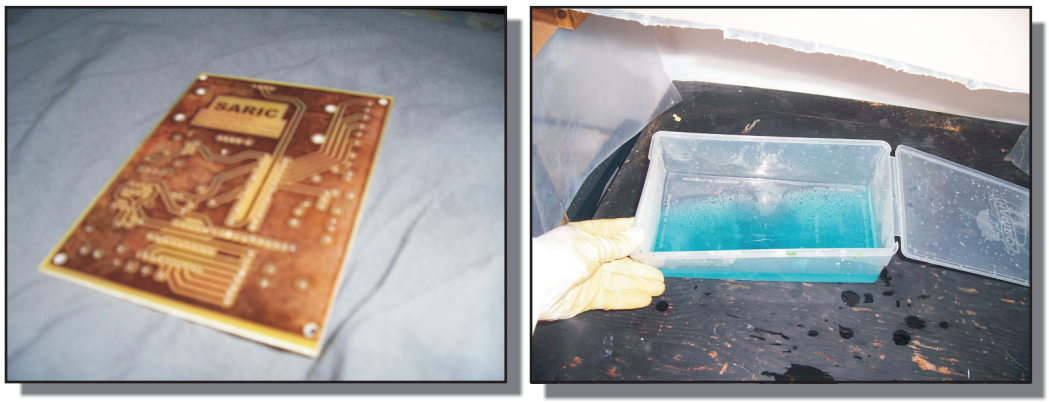


Figura 6.14: Atacado químico sobre la puerta inteligente

Perforado

Las perforaciones, o vías, del circuito impreso se taladran con pequeñas brocas hechas de carburo de tungsteno. El perforado es realizado por maquinaria automatizada, controlada por una cinta de perforaciones o archivo de perforaciones. Estos archivos generados por computador son también llamados taladros controlados por computador (NCD por sus siglas en inglés) o archivos Excellon. El archivo de perforaciones describe la posición y tamaño de cada perforación taladrada.

Cuando se requieren vías muy pequeñas, taladrar con brocas es costoso, debido a la alta tasa de uso y fragilidad de éstas. En estos casos, las vías pueden se evaporadas por un láser. No obstante las vías perforadas de esta forma suelen tener una terminación de menor calidad al interior del orificio. Estas perforaciones se llaman micro vías.

También es posible usando taladrado con control de profundidad, con perforado láser, o pre-taladrando las láminas individuales antes de la laminación, producir perforaciones que conecten sólo algunas de las capas de cobre, en vez de atravesar la tarjeta completa. Estas perforaciones se llaman vías ciegas cuando conectan una capa interna con una de las capas exteriores, o vías enterradas cuando conectan dos ca-

pas internas.

Las paredes de los orificios, para tarjetas con dos o más capas, se metalizan con cobre para formar orificios metalizados, que conectan eléctricamente las capas conductoras del circuito impreso. Para este proceso es necesario disponer de maquinaria especial y puede sustituirse soldando un pequeño cable entre los dos orificios de la vía.

En los primeros prototipos de ambas placas, los taladros se hicieron de forma manual como se puede observar en la figura 6.15, por no disponer de la maquinaria adecuada. En el segundo prototipo sí se utilizó maquinaria automática y en la figura 6.16 se puede ver un fragmento del fichero de perforaciones.

Figura 6.15: Taladrado de la placa manual

```
1         Drill Listing
2 -----------------
3 Drill: .02  Tool: 1  Feed: 197  Speed: 550
4 X 122500  Y 140500
5 X 162500  Y 130500
6 X 212500  Y 130500
7 X 262500  Y 130500
8 X 300000  Y 130500
9 X 330000  Y 123000
10 X 365000  Y 130500
11 X 397500  Y 130500
12 X 397500  Y 140500
13 X 427500  Y 140500
14 X 427500  Y 130500
15 X 437500  Y 130500
16 X 508900  Y 143000
17 X 514400  Y 131800
```

Figura 6.16: Fragmento del fichero de taladros

Estañado y máscara antisoldante

Los pads y superficies en las cuales se montarán los componentes, se suelen metalizar, ya que el cobre al desnudo no es fácilmente soldable. Tradicionalmente, todo el cobre expuesto era metalizado con soldadura. Esta soldadura solía ser una aleación de plomo-estaño, sin embargo, se están utilizando nuevos compuestos para cumplir con la directiva RoHS de la UE, la cual restringe el uso de plomo. Los conectores de borde, que se hacen en los lados de las tarjetas, a menudo se metalizan con oro. El metalizado con oro a veces se hace en la tarjeta completa.

Las áreas que no deben ser soldadas pueden recubrirse con un polímero resistente a la soldadura, el cual evita cortocircuitos entre los pines cercanos de un componente.

Serigrafía

Los dibujos y texto se pueden imprimir en las superficies exteriores de un circuito impreso mediante serigrafía. Cuando el espacio lo permite, el texto de la serigrafía puede indicar los nombres de los componentes, la configuración de los interruptores, puntos de prueba, y otras características útiles en el ensamblaje, prueba y servicio de la tarjeta.

Montaje

En las tarjetas los pines de los componentes se insertan en los orificios (pads), y son fijados eléctrica y mecánicamente a la tarjeta con soldadura.

Con la tecnología de montaje superficial (surface mount en inglés), los componentes se sueldan a los pads en las capas exteriores de la tarjetas. A menudo esta tecnología se combina con componentes through hole, debido a que algunos componentes están disponibles sólo en un formato. (Ver figura 6.17

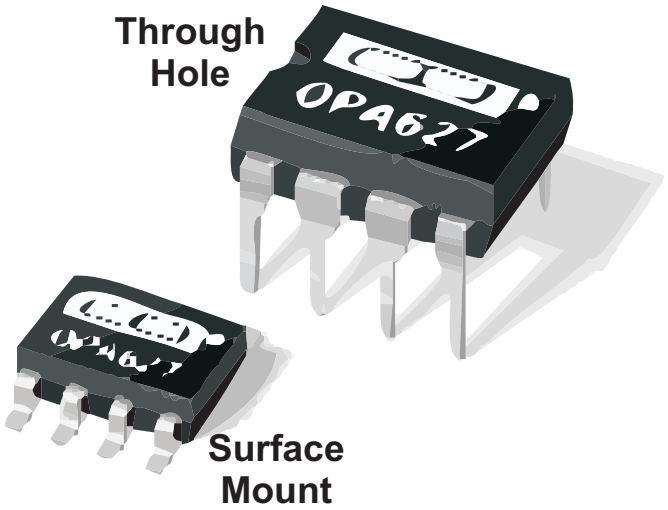


Figura 6.17: Tecnologías de montaje superficial (surface mount) y through hole

Pruebas y verificación

Las tarjetas sin componentes pueden someterse a pruebas al desnudo, donde cada conexión definida en el netlist se verifica en la tarjeta finalizada. Para producciones en grandes volúmenes, se utiliza una cama de clavos para hacer contacto con las áreas de cobre u orificios en ambos lados de la tarjeta para facilitar las pruebas. Un ordenador le indicará a la unidad de pruebas eléctricas, que envíe una pequeña corriente a través de cada contacto en la cama de clavos, y que verifique que esta corriente se recibe en el otro extremo del contacto. Para volúmenes medianos o pequeños, se utilizan unidades de prueba con un cabezal flotante para hacer los contactos.

Protección y empaquetamiento

Los circuitos impresos que se utilizan en ambientes extremos, normalmente tienen un recubrimiento protector que se aplica o bien usando un aerosol o directamente sumergiendo la tarjeta, después de que se hayan soldado los componentes. El recubrimiento previene la corrosión y las corrientes de fuga o cortocircuitos producto de la condensación. Los primeros recubrimientos utilizados eran ceras. Los recubrimientos modernos están constituidos por soluciones de goma silicosa, poliuretano, acrílico o resina epóxica. Algunos son plásticos aplicados en una cámara al vacío.

6.4 Diseño de la puerta aislada

En la introducción de este capítulo se hace una introducción al microprocesador que usa este dispositivo, el PIC16F878 de Microchip. Éste nos permite conectar una gran cantidad de dispositivos gracias a que dispone de 3 puertos de E/S de propósito general. Es necesario conectar a éste una serie de periféricos de forma que el dispositivo realice todas las funciones para las que se ha diseñado, permita fácilmente su expansión y mantenga el funcionamiento lo mas sencillo posible. No obstante hay partes del diseño que quedan abiertas conservando la filosofía de mantener el coste de la Puerta Aislada lo más bajo posible, de

forma que con las mismas especificaciones, se pueda variar sus capacidades, conectar diferentes periféricos y adaptar las posibilidades del dispositivo a restricciones de coste o funcionalidad.

Lector de Tarjetas Para permitir conectar el chip de la tarjeta al microcontrolador es necesario incluir un dispositivo adaptador de forma que podamos integrarlo en la placa. Este dispositivo tiene como función principal: conectar ciertos pines del microcontrolador con los de la tarjeta e indicar cuando se encuentra introducida ésta.

Display Alfanumérico Mediante este dispositivo podemos ofrecer al usuario ayuda y permitir una mejor interactividad.

In-Chip-Debugger Para utilizar las capacidades de depuración del chip es necesario incluir un conector para conectarlo al depurador externo y de este al PC.

Relé Para enviar la señal de apertura a la puerta, con un relé podemos cambiar fácilmente el voltaje que se suministra a la cerradura.

Optoacoplador Incluimos un optoacoplador para utilizarlo como interfaz de entrada de otros dispositivos: Sensores infrarrojos, lectores biométricos serie, detectores de proximidad... Con esto ampliamos las capacidades del dispositivo y lo hacemos fácilmente extensible

Interfaz para 3 botones Se incluye un conector que permite conectar hasta 3 botones o pulsadores para dotar a la interfaz de funciones mas complejas, también es posible conectar un teclado matricial mediante un chip paralelo-serie i2c.

Selector de modo de operación Para hacer el dispositivo y el software más rehusable, se incluye un selector de configuraciones que permite hasta 16 modos diferentes de operación, este puede usarse por ejemplo para permitir 16 tipos de smartcards diferentes, o para seleccionar el tipo de dispositivo conectado al optoacoplador o a la interfaz de 3 botones.

Zumbador Un zumbador opcional es un pequeño altavoz capaz de emitir pitidos a una frecuencia determinada, es útil para informar de errores o como alarma.

LED Un led opcional para indicar que el aparato esta en funcionamiento.

En la figura 6.18 se muestra un diagrama de bloques con la conexión de todos estos periféricos al microprocesador y en la figura 6.19 una fotografía con algunos de los componentes antes de ser soldados.

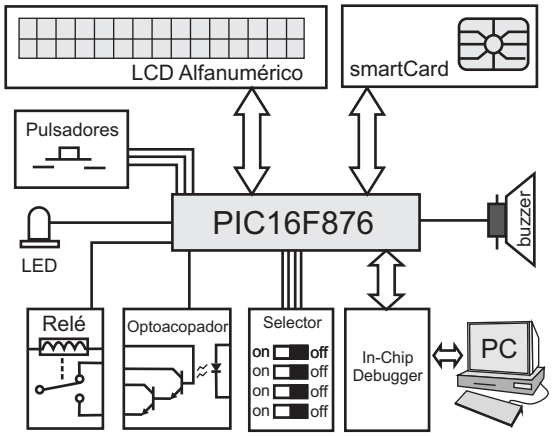


Figura 6.18: Diagrama de Bloques de la puerta Aislada

6.4.1 Diseño eléctrico

Una vez definidos todos los periféricos a conectar, el siguiente paso es asignarle a cada uno patillas en el microcontrolador, definir los buses y las señales de interrupción. En las figuras 6.20 y 6.21 se muestra el esquemático final de la puerta Aislada dividido en dos hojas, en la primera está toda la lógica y periféricos externos, en la segunda los reguladores de tensión con selectores para que pueda adaptarse fácilmente a diferentes voltajes de operación.

A los periféricos comentados en la sección anterior se ha añadido un botón de reset útil durante el desarrollo del firmware y el cristal de cuarzo a 4Mhz. Las etiquetas identificadas como *MACROCLEMA1* y *MACROCLEMA2* son los puertos de expansión que permiten conectar

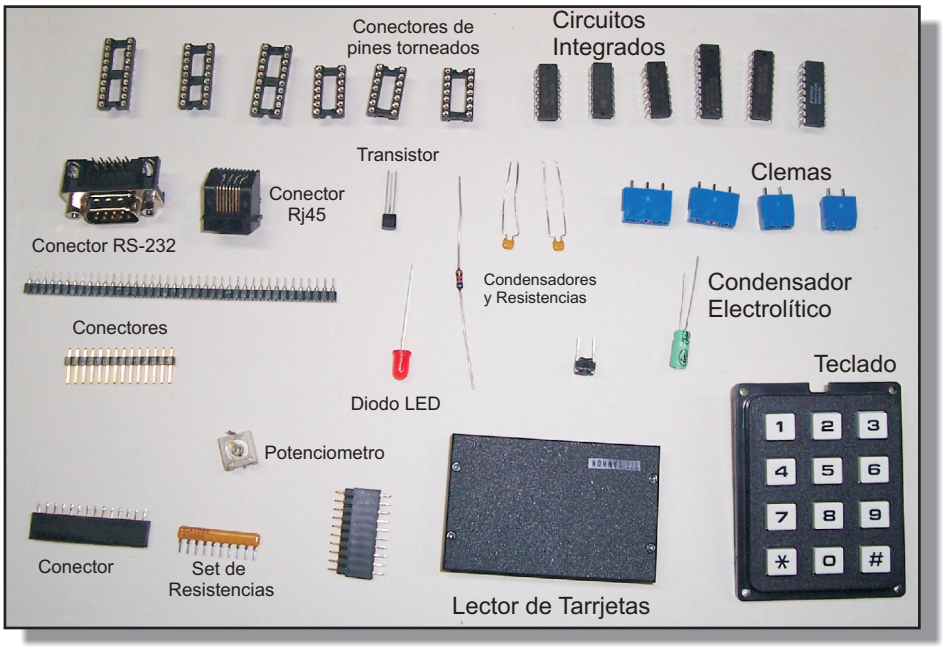


Figura 6.19: Algunos de los componentes que componen la puerta

los periféricos adicionales, botones, un zumbador, leds... En el selector de modo de operación identificado en el diagrama como *SWITCHDIP*, se han colocado resistencias de pull-up ya que son necesarias por el tipo de puertos disponibles en el *PIC16F876* al ser estos de tipo colector abierto.

Para facilitar la detección de la inserción de una tarjeta, se ha conectado el lector de tarjetas a la señal de interrupción externa del PIC, de este modo, cada vez que se inserte una tarjeta se producirá una interrupción. La elección del *Puerto A* para conectar los botones tampoco es una decisión arbitraria, ya que estos pueden configurarse para provocar interrupciones en flancos de subida o bajada, de esta forma es más sencillo detectar la pulsación de estos evitando tener que sondear el puerto constantemente.

Por último, es necesario señalar que todo el diseño se ha pensado para admitir diferentes configuraciones de periféricos de modo que con la misma placa sea posible montar desde el dispositivo más básico

hasta el más avanzado. Son opcionales el buzzer, los botones, el LED y el relé. Además tanto el relé como el buzzer es posible soldarlos a la placa o conectarlos a una de las clemas si se necesita colocar estos a cierta distancia de la placa.

6.4.2 Diseño de la PCB

Una vez terminado el esquemático, hay que trasladar éste al diseño real para fabricar la Placa de Circuito Impreso. Se ha optado por un diseño en dos capas, ya que sin equipo profesional es muy complicado incluir capas internas. En las figuras 6.32 y 6.33 se muestran los *Fotolitos*[13] de ambas caras. Es fácil observar las zonas sombreadas que destacan del resto, esto son los planos de masa que ayudan a garantizar el correcto funcionamiento de todo el dispositivo.

El tamaño final del dispositivo es de 108x75 milímetros.

6.5 Diseño de la puerta inteligente

La filosofía de diseño a seguir en esta puerta es completamente diferente a la anterior, ahora trabajamos con un microcontrolador más avanzado, con buses de direcciones y datos y menos pines de E/S que el anterior, por lo que será necesario usar técnicas para multiplexar el bus de datos y poder conectar varios periféricos simultáneamente. Los dispositivos y periféricos externos que se deben conectar son similares a los de la puerta aislada: un lector de tarjetas, el display alfanumérico, el relé para la apertura de la puerta y el LED. Adicionalmente y ya que esta puerta es un modelo más avanzado se incluye:

Conector Ethernet RJ-45 Un conector de red estándar par integración en *PCBs*[14]

Conector RS-232 Un conector *RS-232*[15] para permitir comunicación

[13] *Fotolito*: Ver definición en página **??**
[14] *PCB*: Ver definición en página 24
[15] *RS-232*: Ver definición en página **??**

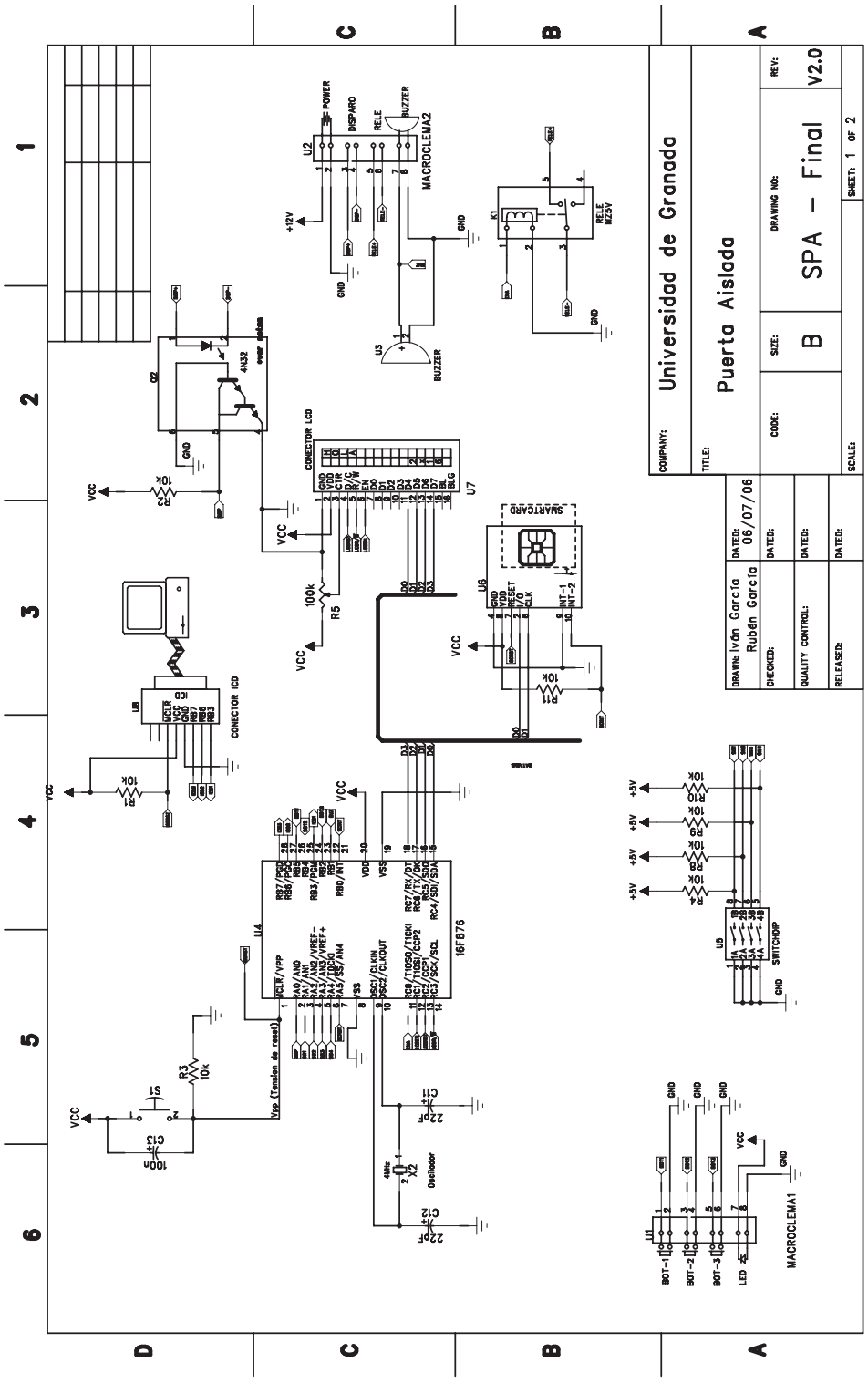


Figura 6.20: Esquemático de la Puerta Aislada. Hoja 1

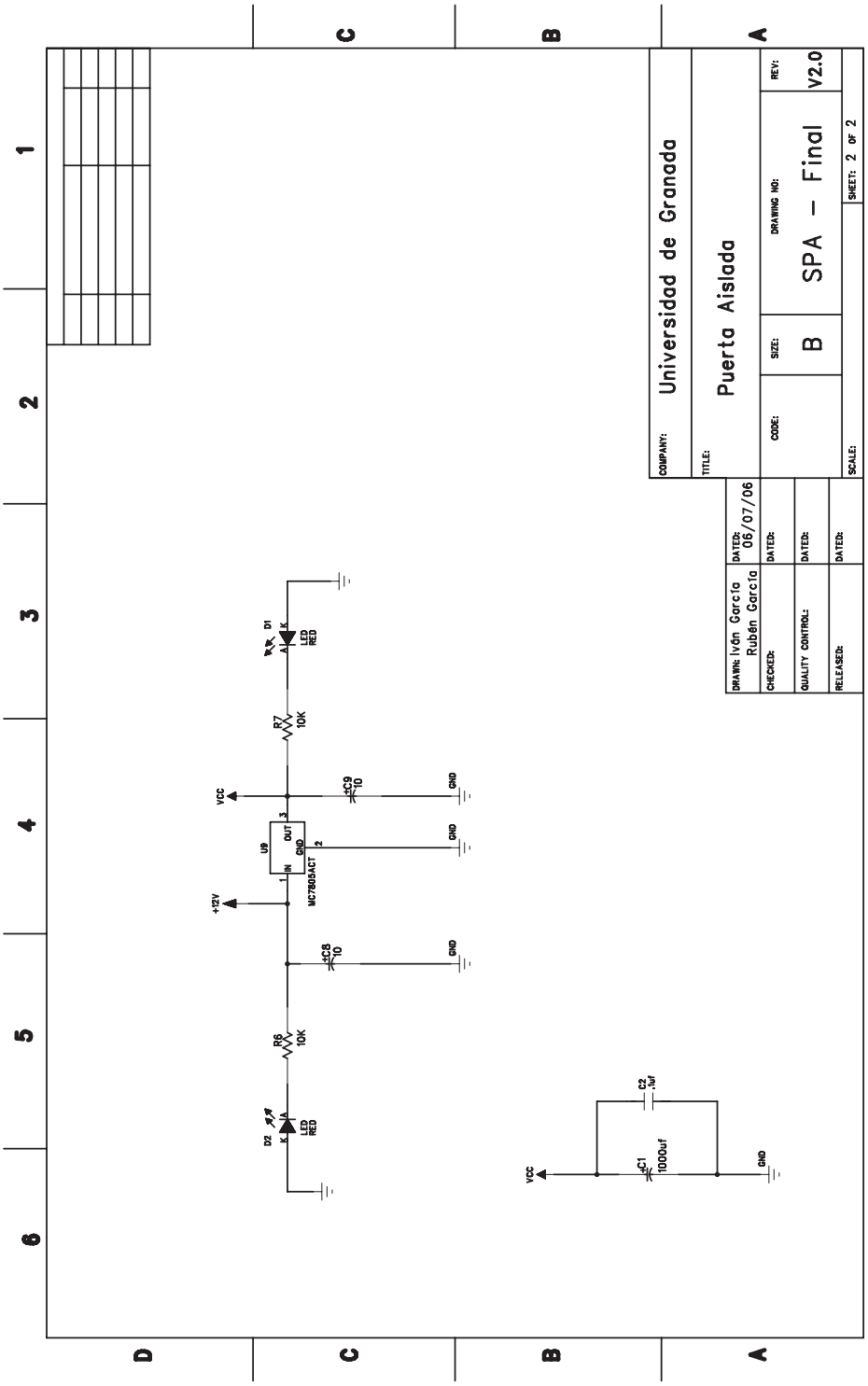


Figura 6.21: Esquemático de la Puerta Aislada. Hoja 2

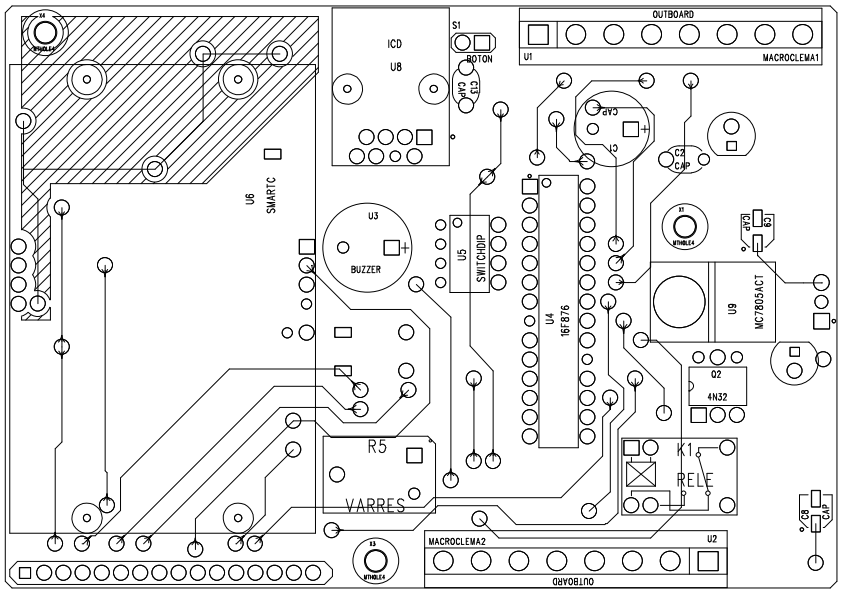


Figura 6.22: Fotolito de la capa TOP de la puerta Aislada

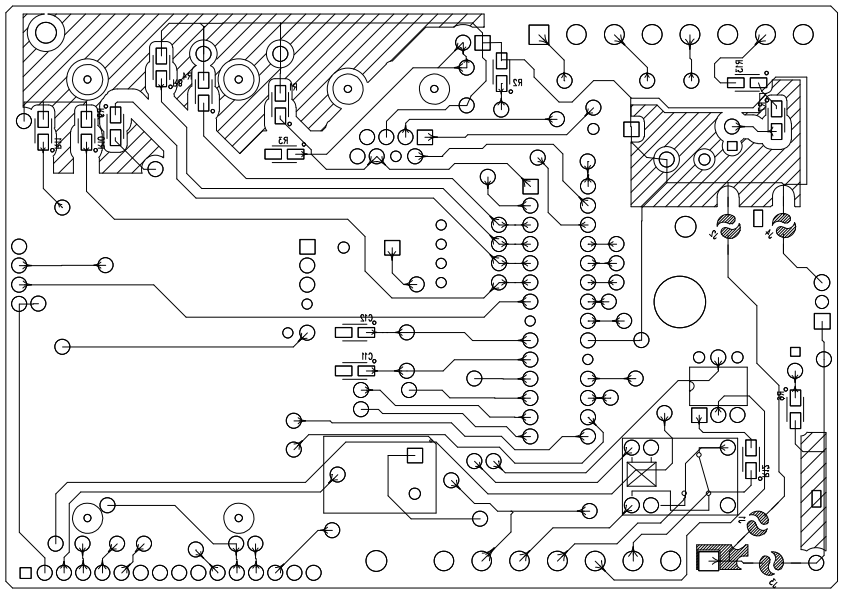


Figura 6.23: Fotolito de la capa BOTTOM de la puerta Aislada

por un puerto de serie mediante el estándar RS232. Esto es necesario para actualizaciones de firmware y configuración inicial de la interfaz de red: Dirección IP, Mascara de Red...

Teclado de 4x3 Para permitir introducir contraseñas se incluye en esta puerta un teclado numérico. Es necesario incluir también un chip que decodifique las señales recibidas por el teclado.

Lógica para el puerto paralelo Latch y buffers que permitan realizar lecturas y escrituras en el puerto paralelo. Esto se explica con más detalle en el siguiente apartado.

En la figura 6.24 se muestra un diagrama de bloques con la conexión de todos estos periféricos a TINI omitiendo detalles como resistencias y lógicas de decodificación. En el siguiente apartado se profundiza en el diseño eléctrico cubriendo estos aspectos.

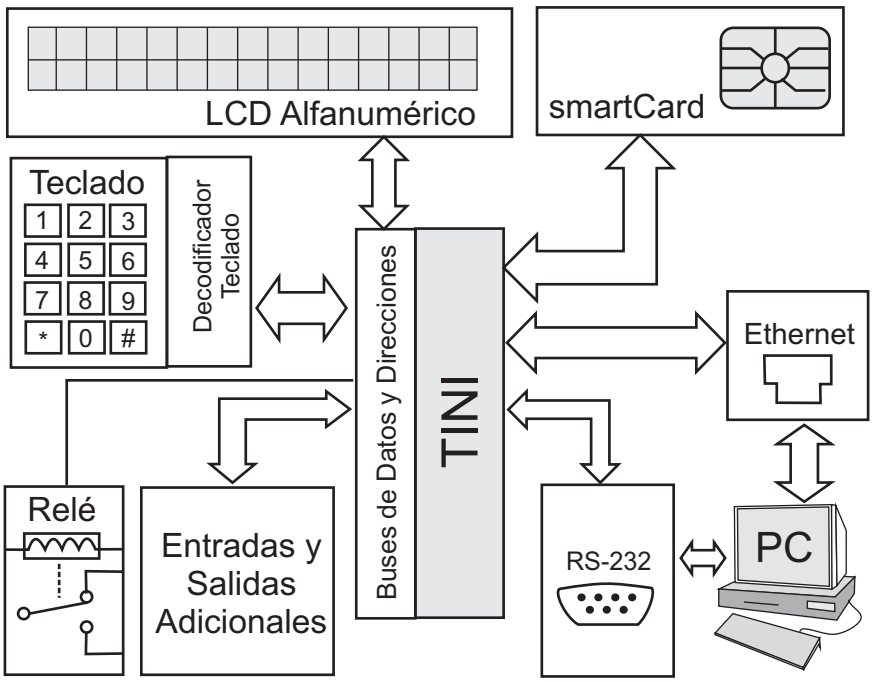


Figura 6.24: Diagrama de Bloques de la puerta Inteligente

bus	Bus es el conjunto de conductores eléctricos en forma de pistas metálicas impresas sobre la placa base del computador, por donde circulan las señales que corresponden a los datos binarios del lenguaje máquina con que opera el Microprocesador. Bus es una palabra inglesa que significa "transporte"; aplicada a la informática, se relaciona con la idea de las transferencias internas de datos que se dan en un sistema computacional en funcionamiento. En el bus todos los nodos reciben los datos aunque no se dirijan a todos estos, los nodos a los que no van dirigidos los datos simplemente los ignoran.

6.5.1 Diseño eléctrico

El bus paralelo

El bus paralelo de TINI se usa, como mínimo, para comunicarse con chips de memoria externa para el almacenamiento de código y datos. También se accede a los dispositivos periféricos como el controlador *Ethernet*[16] y el reloj de tiempo real mediante el bus paralelo. El diagrama de bloques mostrado en la figura **??** constituye una configuración bastante genérica para conectar dispositivos externos al bus. Como sus nombres indican, el bus de direcciones específica la dirección destino de la lectura o las operaciones de escritura, mientras que el bus de datos transfiere los datos binarios hacia y desde el dispositivo.

La combinación de ciertas señales de control y posiblemente ciertas líneas de dirección puede usarse en conjunción con la lógica de descodificación para actual como una señal de habilitación (del inglés: enable) para el periférico. El propósito de la señal de habilitación es asegurar que solamente las operaciones del bus que van dirigidas al dispositivo son vistas por el dispositivo. Hay que hacer notar que algunos dispositivos, incluyendo muchos chips de memoria, pueden conec-

[16] *Ethernet*: Ver definición en página 9

buffer	Un buffer es un espacio de memoria, en el que se almacenan datos para evitar que el recurso que los requiere, ya sea hardware o software, se quede en algún momento sin datos.

tarse directamente al bus sin necesidad de usar ninguna lógica externa.

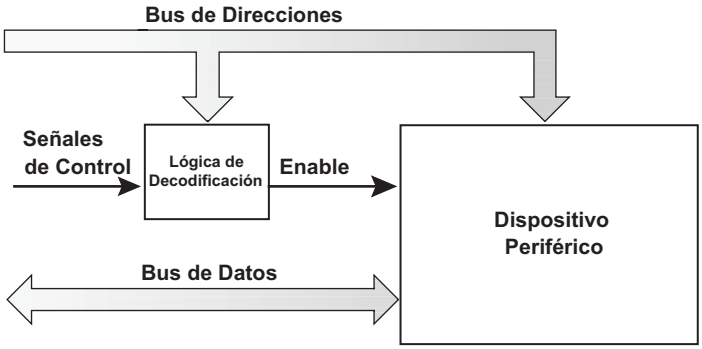


Figura 6.25: Interfaz de dispositivos externos con el bus

Un punto importante a resaltar es que la figura 6.25 muestra las señales de dirección y de datos conectadas directamente al dispositivo externo directamente. Esto es apropiado a veces. Sin embargo, dependiendo del número total de dispositivos en el bus paralelo, tanto las señales de dirección como los datos pueden necesitar el uso de memoria intermedia para asegurar la fiabilidad de la operación. Los buffers son chips que permiten retener un valor temporalmente y así liberar al bus antes para no sobrecargarlo. Si es necesario o no usar buffers, y como conectarlos con el bus, son cuestiones que dependen del diseño. Las señales que constituyen el bus paralelo del microcontrolador pueden agruparse en las siguientes categorías:

- Bus de datos bidireccional
- Bus de direcciones unidireccional
- Señales de control para distinguir entre operaciones de escritura o de lectura y de selección de dispositivo (chip)

Todos los datos, direcciones y señales de control de las que dispone TINI se listan, junto con pequeñas descripciones en la figura 6.26. El bus de datos (D0-D7) es un bus bidireccional de 8-bits. Todas las transferencias de datos se realizan mediante este bus, incluyendo las capturas de código de flash ROM, capturas de datos de RAM estática y operaciones de lectura o escritura a periféricos conectados al bus. Los dispositivos externos son direccionados usando el bus de direcciones de 20-bits (A0-A19) junto con una de las 8 señales precodificadas "chip select". El bus de datos de 20-bits proporciona un rango de direcciones de 1-megabyte. Sin embargo, este rango se expande mediante las ocho señales de selección de chip codificando cada una un espacio de 1-megabyte. La selección de chip es de dos tipos: chips enables (CEs) y Periferial Chip Enables (PCEs). Hay 4 señales de cada una.

El mapa de memoria, mostrado en la figura 6.27, está dividido en dos extensiones de 4 megabyte distintas. El espacio CE contiene todos los chips de memoria usados como almacenamiento de programa y de datos para el entorno de ejecución. También contiene un área de 1-megabyte para periféricos que sirve para direccionar dispositivos de alta velocidad que soporten una interfaz de bus paralela. La diferencia principal entre las señales CE y PCE es que las señales PCE solo pueden ser usadas para las lecturas y escrituras de datos. En otras palabras, el microcontrolador no puede capturar código nativo de dispositivos de memoria que sean activados usando las señales PCE. Por esto es por lo que se accede a la memoria Flash ROM y la RAM estática utilizando las señales CE. Las direcciones de CE se corresponden con direcciones físicas reales del mapa de memoria del microcontrolador.

La dirección de inicio de las memorias controladas por señales PCE es de alguna forma arbitraria ya que se mapea en direcciones virtuales. Esto requiere de un mayor uso del procesador para llevar a cabo la conversión a direcciones reales, por lo que las transferencias de datos usando el espacio de PCE son hasta tres veces más lentas que usando CE.

Identificador	Nombre Completo	Descripción
D0-D7	Bus de Datos	Bus de datos bidireccional de 8 bits
A0-A19	Bus de Direcciones	Bus de direcciones de 20-bits
$\overline{CE0}$-$\overline{CE3}$	Chip Enables Habilitadores de Chips	Las lineas de habilitación se usan para seleccionar memoria o periféricos. Las capturas de código deben realizarse de chips de memoria habilitados por una de estas señales
$\overline{PCE0}$-$\overline{PCE3}$	Periferial Chip Enables Habilitadores de Chips para periféricos	Las lineas de habilitación para periféricos se usan normalmente para almacenaje de datos. No se puede capturar código nativo de chips de memoria habilitados por estas señales.
$\overline{PSEN}$	Program Store Enable	Lineas sonda usadas para controlar las capturas de código de dispositivos de memoria externa habilitados por lineas CE. También puede usarse para captura de datos.
$\overline{RD}$	Sonda de lectura	Linea sonda de lectura usada para capturas de datos de memoria y otros dispositivos periféricos habilitados por lineas PCE
$\overline{WR}$	Sonda de escritura	Linea sonda usada para escritura de datos en memoria y otros dispositivos periféricos.
DRST	Reset del dispositivo	El pin 3.4 del microcontrolador. Esto no es una señal definida formalmente en el bus paralelo. En TINI se usa para resets externos.

Figura 6.26: Señales del bus paralelo en TINI

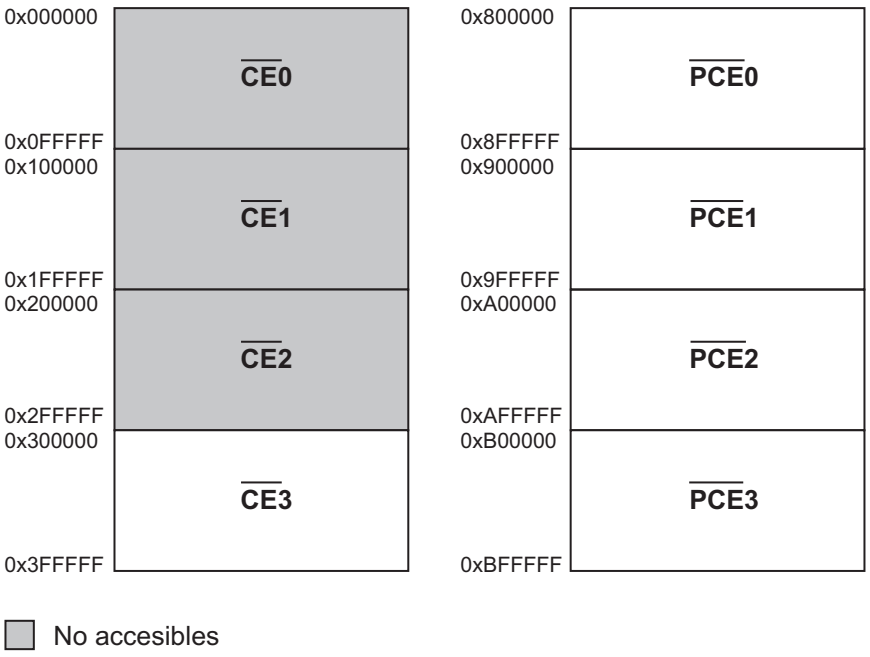


Figura 6.27: Mapeado de las direcciones PC y PCE

El entorno de ejecución de TINI no reserva espacio PCE para dispositivos periféricos. Esto implica que las cuatro señales PCE, y los cuatro megabytes de espacio de direcciones que controlan, están enteramente disponibles para los diseñadores de sistemas. Sin embargo, muchos dispositivos de alta velocidad están mapeados en el espacio CE3 porque el microcontrolador puede acceder a ellos de forma más eficiente. Si no se mapea ningún dispositivo en el espacio PCE, los 4 pines PCE pueden usarse como pines de propósito general. El diseñador es libre de usar el área de periféricos (pines PCE0-PCE3) para conectar hardware directamente al bus paralelo del microcontrolador, sin embargo también es posible usar estos pines como señales TTL de E/S, pero no ambas a la vez.

Para conectar todos los periféricos descritos al principio de esta sección necesitamos más señales que las que ofrece el microcontrolador. La solución es hacer uso del bus de datos para obtener entradas y salidas adicionales. En la figura 6.28 se muestra un fragmento del esquemático de la puerta inteligente donde se ha usado este concepto para obtener 8 entradas y 8 salidas independientes usando un latch

octal (74HC574) y un buffer también octal (74HC541). Cada entrada se lleva a Vcc usando resistencias de 10 kohm. Ambos chip se decodifican en el espacio de 1-megabyte controlado por PCE1.

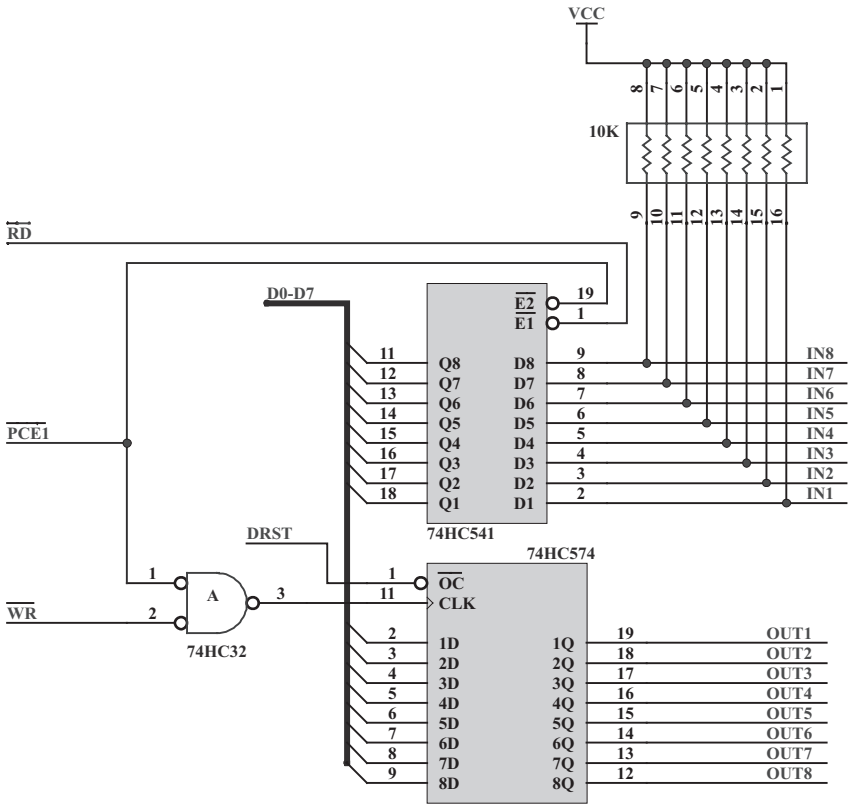


Figura 6.28: 8 líneas de E/S decodificadas en el espacio PCE

Para obtener la dirección en el espacio de direcciones debemos hacer las siguientes consideraciones:

- La dirección es un número de 32 bits (a0-a31)
- Como el espacio PCE termina en 0xC00000, los bits mas significativos a31-a24 deben ser 0.
- Además, del mapa de memoria mostrado en la figura 6.27 obtenemos que la dirección más baja en el espacio PCE es 0x800000. Esto implica que a32 debe ser 1.
- Debido a que hay un agujero en el mapa de memoria entre las direcciones 0x400000 y 0x800000, a22 debe valer 0. Este rango

es tierra de nadie porque el microcontrolador mapea varias áreas del sistema incluyendo la pila en este área.

- Por último, los bits a20 y a21 quedan determinados por la señal PCE que se use. De la figura 6.29 obtenemos que ya que hemos usados PCE1, a20 es 1 y a21 es 0.
- Los bits restantes a19-a0 no se tienen en cuenta ya que no se han usado en la lógica de decodificación.

$\overline{PCE0}$	$\overline{PCE1}$	$\overline{PCE2}$	$\overline{PCE3}$	a20	a21
0	1	1	1	0	0
1	0	1	1	1	0
1	1	0	1	0	1
1	1	1	0	1	1

Figura 6.29: Equivalencias de PCE a a20-a21

La dirección resultante se muestra en la figura 6.30. Si por ejemplo elegimos los bits sin importancia como 0, tenemos que la dirección de acceso es **0x900000**.

$\overline{a31 - a24}$	$\overline{a23}$	$\overline{a22}$	$\overline{a21}$	$\overline{a20}$	$\overline{a19 - a0}$
00000000	1	0	0	1	XXXXXXXXXXXXXXXXXXX

X => no se tiene en cuenta

Figura 6.30: Dirección de acceso a las 8 líneas decodificadas

El esquemático completamente terminado se muestra en la figura 6.31.

6.5.2 Diseño de la PCB

Al igual que con la puerta aislada, una vez terminado el esquemático trasladamos el diseño a una Placa de Circuito Impreso también en

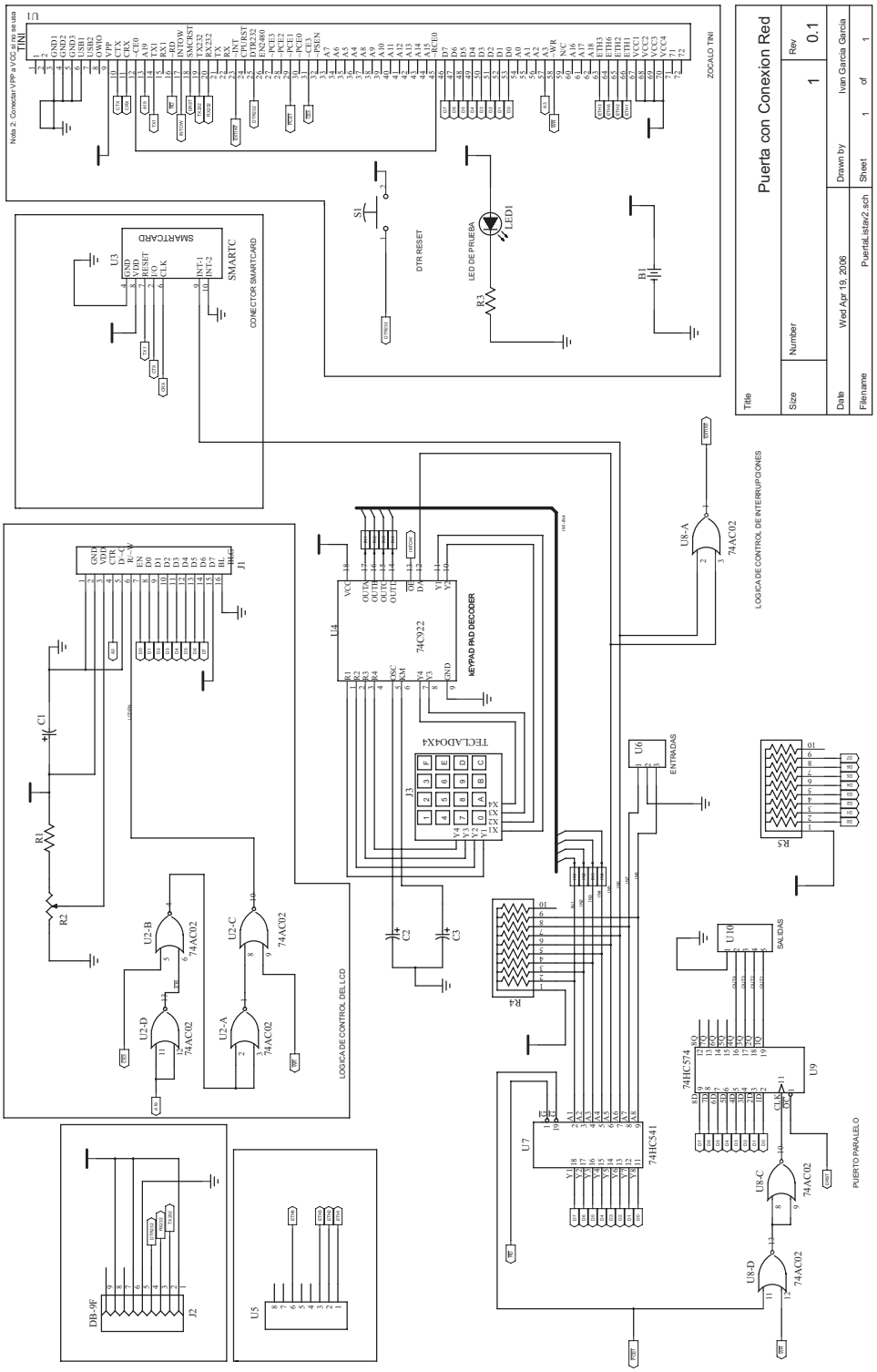


Figura 6.31: Esquemático de la puerta inteligente

dos capas. En las figuras **??** y **??** se muestran los *Fotolitos*[17] de ambas caras. Es este caso se ha optado por enrutar todas las pistas correspondientes a masa en la capa BOTTOM. Este fotolito se corresponde con el último prototipo en el que se uso una maquina fresadora para la fabricación de las placas.

Este dispositivo es un poco mayor que el anterior, en las figuras **??** y **??** se muestra a tamaño real.

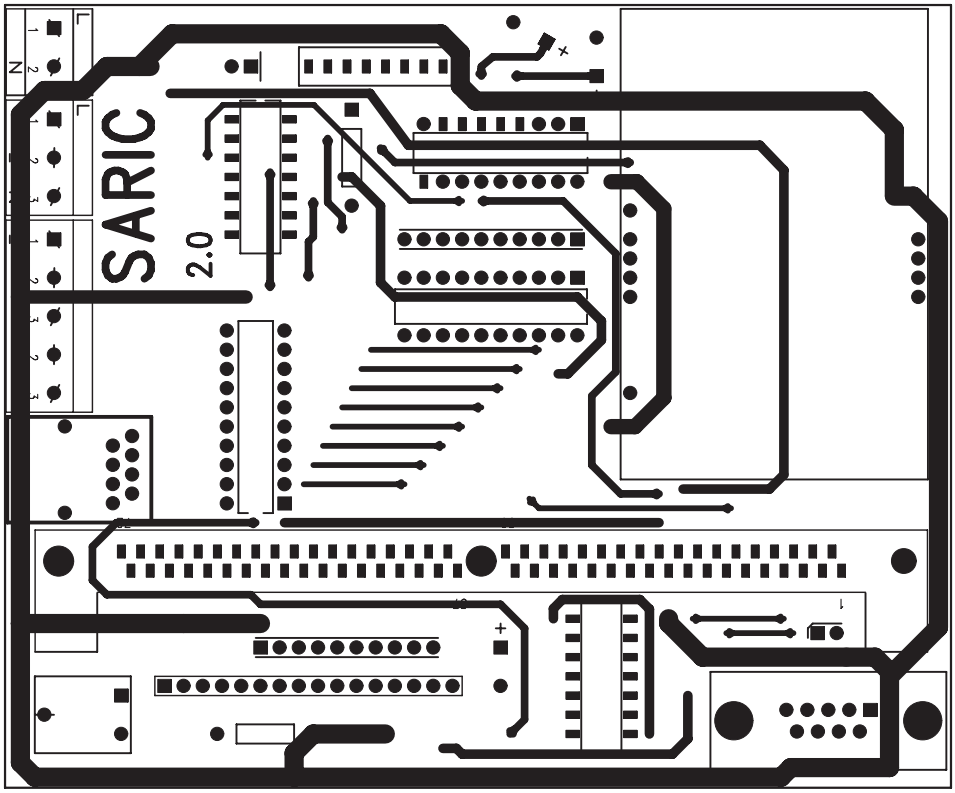


Figura 6.32: Fotolito de la capa TOP de la puerta Inteligente

Por último en la figura 6.36 se muestra el prototipo terminado y funcionando.

[17] *Fotolito*: Ver definición en página **??**

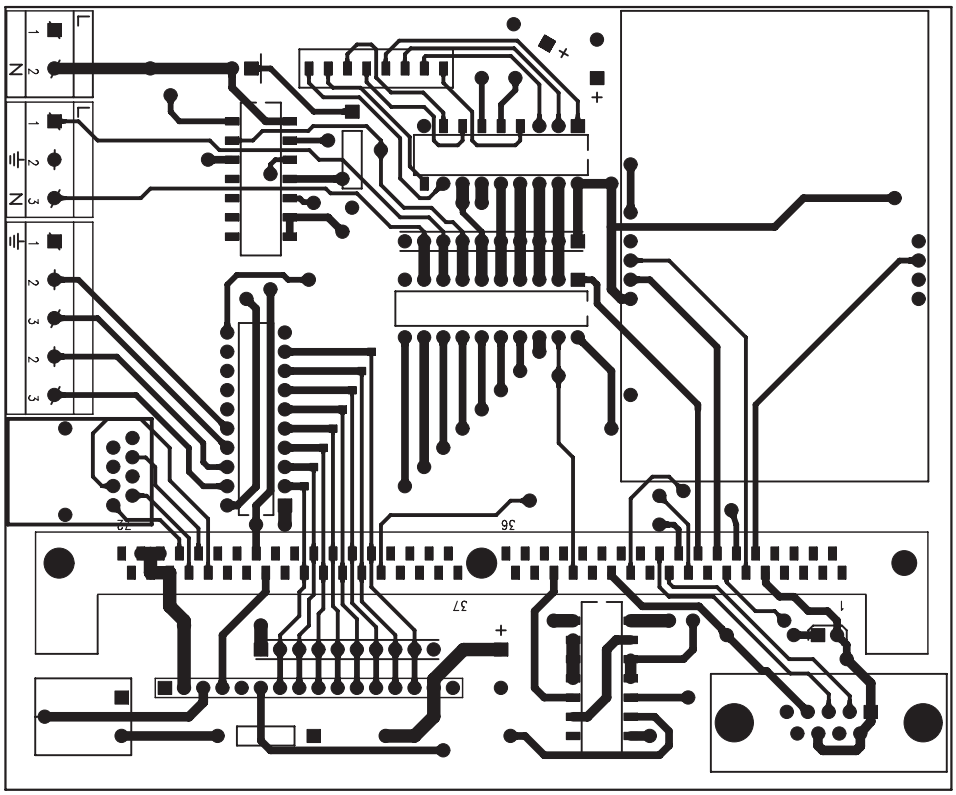


Figura 6.33: Fotolito de la capa BOTTOM de la puerta Inteligente

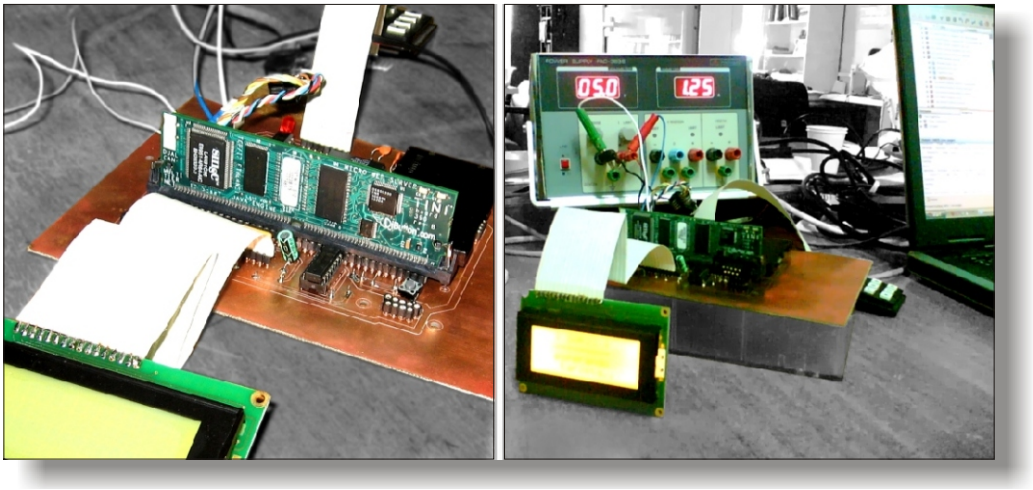


Figura 6.34: Prototipo de la puerta inteligente terminado

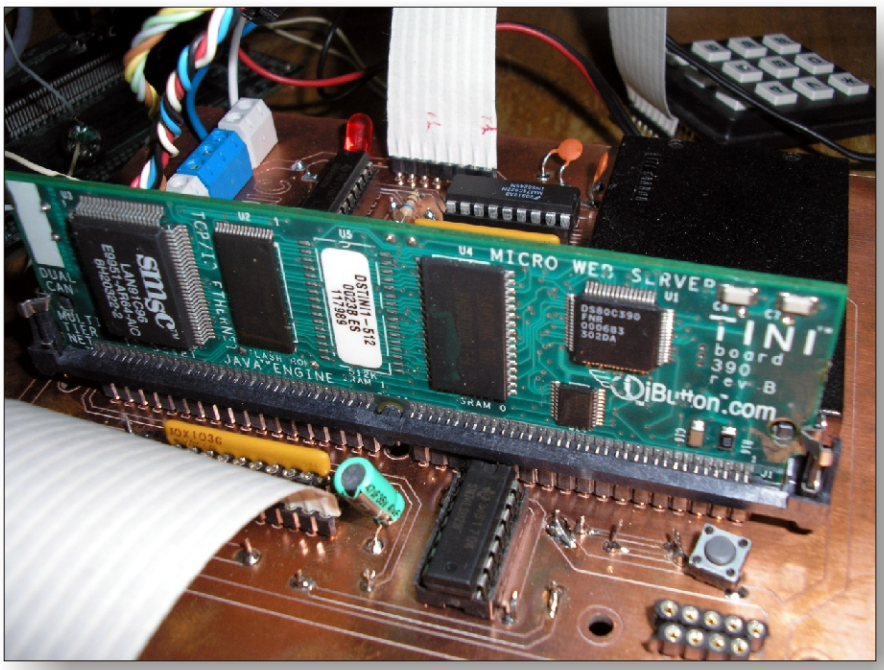


Figura 6.35: Vista superior del prototipo de la puerta inteligente

Figura 6.36: LCD conectado a uno de los dispositivos

7 Firmware

El microcontrolador con el que van equipados ambos sistemas de acceso es el cerebro de todo el sistema ya que se encarga de controlar todos los periféricos, procesar la información leída de la tarjeta y accionar los mecanismos de aviso y de apertura de la puerta si procede.

Todo esto se realiza siguiendo las órdenes de un programa que hay que incorporar al microcontrolador ya que, por si solo, éste no realiza ninguna función. Para incorporar el programa hay que grabarlo en la memoria interna del microcontrolador o bien en una RAM externa de forma que este sea capaz de cargarlo una vez sea iniciado.

El termino firmware, que podríamos traducir al castellano como "microprograma", fue acuñado originalmente para indicar la unión entre el hardware y el software en un mismo concepto. El firmware es una combinación de software (programa) y hardware (en este caso, chips). Estos chips normalmente son de los siguientes tipos:

- ROM Read Only Memory (Memoria de solo lectura)
- PROM Programmable read-only memory (memoria programable de solo lectura)
- EPROM Erasable programmable read-only memory (memoria programable y con capacidad de ser borrada)
- EEPROM Electronicaly Erasable programmable read-only memory

(memoria programable y con capacidad de ser borrada mediante señales eléctricas)

Actualmente y dados los avances producidos en las tecnologías de fabricación de circuitos integrados, los microprocesadores suelen incluir como partes funcionales estas memorias dentro del mismo encapsulado y métodos para poder actualizarlas de forma relativamente sencilla. En términos prácticos, las actualizaciones de firmware pueden mejorar la fiabilidad y caracteristicas de un dispositivo e incluso dotarlo de nuevas funcionalidades.

Existen múltiples lenguajes de programación con los que desarrollar el firmware para el microcontrolador. El lenguaje original o nativo que comprende es el código máquina, compuesto exclusivamente por unos y ceros. Como es de suponer, este tipo de programación no se usa.

El siguiente nivel que se acerca al lenguaje de los humanos es el ensamblador, que sustituye los códigos numéricos del código maquina por códigos nemónicos, más sencillos de recordar y manejar por los programadores. Pero aun así este tampoco nos simplifica en demasía el trabajo ya que es necesario conocer a fondo la arquitectura interna del microprocesador para escribir código eficiente. Se estima que el tiempo de desarrollo se aumenta en un 400% si se decide usar este lenguaje.

Otra posibilidad es el uso de lenguajes de alto nivel y su posterior compilación o traducción al lenguaje maquina. Ejemplos de estos lenguajes de alto nivel son el Basic y el C, siendo este último el que se utilizará para desarrollar el firmware de la puerta Aislada. La elección de este lenguaje se justifica en la gran cantidad de documentación existente, así como su uso generalizado por numerosos programadores para este tipo de aplicaciones. Otro motivo es la existencia de compiladores muy eficientes que producen código máquina de alta calidad.

En un nivel superior se encuentran los lenguajes interpretados

como es el caso de Java que una vez compilados producen código que puede ser interpretado directamente por una máquina virtual; este código se denomina: bytecodes. Este lenguaje se usará en la puerta Inteligente ya que la plataforma TINI dispone de un entorno de ejecución JAVA. Utilizar un lenguaje orientado a objetos presenta muchas ventajas entre las que cabe destacar:

- Fomenta la reutilización y extensión del código.
- Permite crear sistemas más complejos.
- Relacionar el sistema al mundo real.
- Facilita la creación de programas visuales.
- Construcción de prototipos
- Agiliza el desarrollo de software
- Facilita el trabajo en equipo
- Facilita el mantenimiento del software

En la figura 7.1 se presenta la jerarquía de lenguajes descrita anteriormente.

7.1 El firmware en la puerta Aislada

Como se ha comentado en el punto anterior, se ha elegido el lenguaje C para desarrollar el firmware de esta puerta. Primero se desarrollarán bibliotecas para los dos componentes esenciales: display alfanumérico y lector de tarjetas. Como compilador se usará el programa PICC, de Custom Computer Services; a pesar de la gran cantidad de compiladores disponibles, se ha elegido este porque es un IDE que además del compilador integra programación y depuración en el mismo programa.

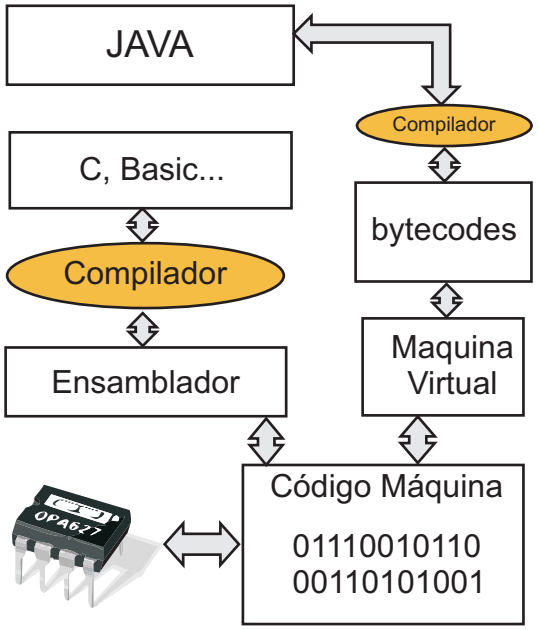


Figura 7.1: Jerarquia de lenguajes de programación

Crear controladores para dispositivos es una tarea compleja, puesto que requiere estudiar detenidamente los diagramas de tiempos del dispositivo y crear rutinas de espera precisas para ajustarse fielmente a estos. Por su rol de protagonista en este proyecto, se analizará detenidamente el controlador del lector de tarjetas inteligentes.

7.1.1 Controlador de SmartCard

El modelo usado es el Siemens SLE4442, con EEPROM Inteligente

IDE	Un entorno de desarrollo integrado o en inglés Integrated Development Environment (IDE) es un programa compuesto por un conjunto de herramientas para un programador. Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica GUI. Los IDEs pueden ser aplicaciones por si solas o pueden ser parte de aplicaciones existentes.

de 256-Byte, protección de escritura y código de seguridad programable. Sus principales características son:

- Memoria EEPROM distribuida en 256 bloques de 8-bits
- Direccionamiento por byte
- Protección de escritura irreversible por byte de los 32 bloques inferiores (Bytes 0 ... 31)
- Memoria de protección distribuida en 32 bloques de 1-bit
- Protocolo de comunicación con 2 cables
- Aviso de final de procesamiento en la salida de datos
- Answer-to-Reset (ATR) cumpliendo con la normativa ISO 7816-3
- Tiempo de programación de 2.5 ms por byte tanto para escritura como borrado
- 10^4 ciclos de escritura/borrado mínimos garantizados
- Retención de los datos mínima de diez años
- Configuración de los contactos e interfaz serie cumpliendo con el estándar ISO 7816 (transmisión síncrona)
- Los datos solo pueden modificarse después de introducir correctamente el código de seguridad programable de 3-bytes

En la figura 7.2 se presenta un esquema de la memoria de la tarjeta inteligente. El bloque de la derecha representa la memoria principal de 256x8bits. La memoria principal se borra y se escribe byte a byte. El borrado consiste en establecer todos los bits de un byte de datos a 1 lógico. La lectura puede realizarse directamente sin ningún tipo de seguridad adicional, sin embargo la escritura pasa antes por la lógica de seguridad. Esta dispone de un contador de errores que inutilizará la tarjeta tras introducir tres códigos erróneos. Una vez pasada esta lógica, se comprueba si la escritura va dirigida a los 32 primeros bytes y en ese caso se comprueba en la memoria de protección que los bits

correspondientes a los bytes que se desean escribir no estén puestos a 1.

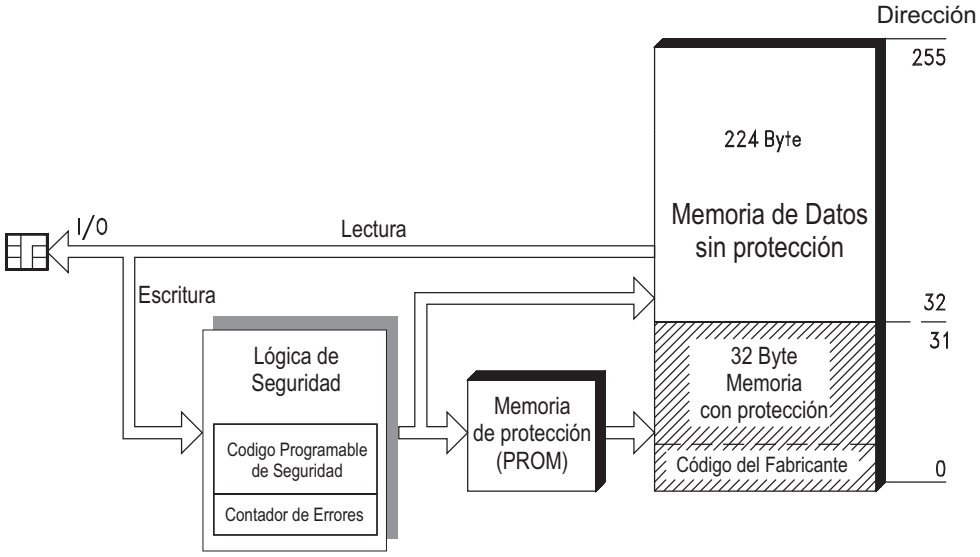


Figura 7.2: Visión general de la Memoria en el chip SLE4442

Como protocolo de transmisión se usa un enlace de dos cables entre el dispositivo de interfaz y el circuito integrado. Es idéntico al protocolo tipo "S = A". Todas los transferencias de datos de E/S se inicializan por el flanco de bajada de CLK. El protocolo de transmisión consta de 4 modos:

Reset y Answer-to-Reset La respuesta al reset (ATR) sigue el estándar ISO 7816-3. Se puede enviar la orden de reset en cualquier momento. Para empezar, se pone a 0 el contador de dirección junto con un pulso de reloj y el primer bit de datos (Bit menos significativo) se envía a la línea I/O cuando la señal RST pasa de nivel alto a bajo. Tras los siguientes 31 pulsos de reloj se reciben los 4 primeros bytes de la EEPROM. El pulso número 33 cambia I/O a alta impedancia y termina el procedimiento ATR.

Modo Comando Después del Answer-To-Reset el chip espera un comando. Cada comando empieza con una condición de inicio, incluyen un comando de 3 bytes seguido de un pulso de reloj adicional y

una condición de parada:

- Condición de inicio: Flanco de bajada en I/O cuando CLK está a nivel alto.
- Condición de parada: Flanco de subida en I/O cuando CLK está a nivel alto.

Tras la recepción del comando, hay 2 posibles modos: Modo de salida de datos para lectura, y modo de procesamiento de datos para escritura.

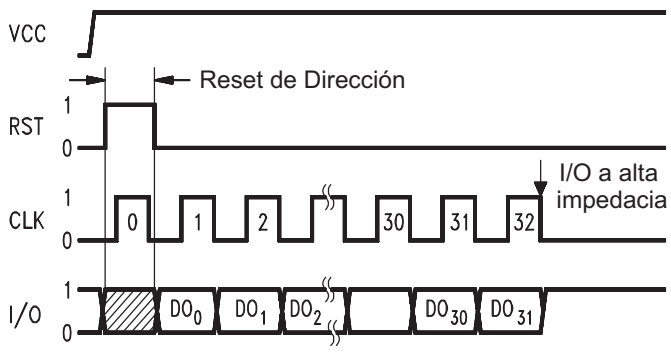
Modo salida de datos En este modo, el circuito integrado envía datos al dispositivo interfaz. El primer bit es válido tras el primer flanco de bajada en CLK. Tras el último bit de datos, se necesita un pulso de reloj adicional para establecer I/O a alta impedancia y dejar el chip listo para el siguiente comando.

Modo de procesamiento En este modo el circuito integrado actúa internamente. Es necesario enviar pulsos de reloj continuamente hasta detectar que I/O pasa a alta impedancia.

El controlador para el lector de tarjetas implementado dispone de funciones que cubren todos modos del protocolo descrito anteriormente:

```
void ATR(uchar *buffer);
void LeerMemoriaPrincipal(uchar dir,uchar* buffer,uchar long);
void LeerMemoriaProtegida(uchar rdBuf[4]);
void EscrituraMemoriaPrincipal(uchar direccion, uchar valor);
void EscrituraMemoriaProtegida(uchar direccion, uchar valor);
void LecturaMemoriaSeguridad(uchar buffer[4]);
void EscrituraMemoriaSeguridad(uchar direccion, uchar valor);
bit VerificaCodigo(uchar valor[3]);
```

A modo de ejemplo, en la figura 7.3 se presenta el diagrama de tiempos del modo ATR y el código fuente de su implementación en C.




```
void ATR(uchar *buffer){
  uchar len;
  CLK=0;
  RST=0;
  IO=1;
  Delay(800);
  RST=1;
  Delay(10);
  CLK=1;
  Delay(10);
  CLK=0;
  Delay(50);
  RST=0;
  Delay(10);
  for(len = 0;len <4;len++)
     buffer[len]=LeeByte();
}
```

Figura 7.3: Diagrama de tiempos e implementación en C del comando Answer-To-Reset

7.1.2 Rutina Principal

La rutina principal de ejecución es muy simple, ya que el dispositivo queda en estado de espera hasta que se introduzca una tarjeta. Esto nos permite ahorrar energía ya que en este modo, el microprocesador consume muchísimo menos. En la figura 7.4 puede verse un diagrama de flujo de dicha rutina. En primera instancia se inicializan variables globales y el estado de todos los periféricos, luego se muestra un mensaje en el display para indicar que se encuentra en funcionamiento, se lee el modo de operación seleccionado y pasa a estado StandBy hasta que se produzca alguna interrupción.

Hay que recordar que este dispositivo esta pensado para actuar de forma diferente según los periféricos que se hayan conectado, esto requiere que el firmware se comporte igualmente de modo diferente para contemplar todos los modos de operación. Aunque el dispositivo está preparado para soportar hasta 16 modos diferentes, en este proyecto solo se van a implementar cuatro que son:

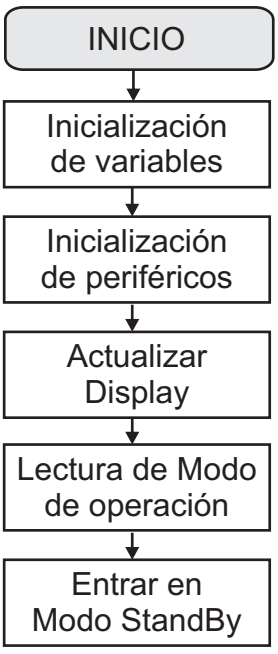


Figura 7.4: Rutina Principal

- 0: Modo normal de operación

- 1: Modo normal de operación con teclado numérico
- 2: Modo parking
- 3: Modo parking con teclado numérico

En los modos 2 y 3 es necesario incluir detectores de proximidad del coche, por ejemplo mediante una bobina, y un detector de infrarrojos para detectar cuando el coche ha terminado de pasar por la barrera y dar la orden de bajarla.

7.1.3 Rutina de Interrupción por SmartCard

El dispositivo se ha diseñado para que cuando se produzca la inserción de una tarjeta inteligente se produzca una interrupción. En la figura 7.5 se muestra el diagrama de flujo correspondiente a esta rutina. El primer paso es obtener todos los datos de la tarjeta. Estos están codificados mediante el algoritmo TEA (Ver apéndices). A pesar de que se han descubierto ataques de criptoanálisis contra este algoritmo, solo con los 256bytes de información que disponen las tarjetas usadas, es prácticamente imposible que estos ataques resulten satisfactorios ya que requieren de muchos datos codificados con la misma clave.

El siguiente paso es obtener el pin de la tarjeta para verificar la autenticidad del usuario. Dependiendo del modo de operación seleccionado la captura de la contraseña se hace con un teclado numérico o con pulsadores. En el caso del teclado numérico, la obtención de los números es mucho más cómoda para el usuario final, ya que solo es necesario introducir los 4 dígitos. A nivel de hardware se requiere que la puerta disponga de un teclado numérico y un chip conversor que retenga el dato pulsado y lo codifique de forma que el microcontrolador pueda leerlo. Esto incrementa ligeramente el coste de la puerta. Si se opta por no usar el teclado, el proceso será más tedioso para el usuario ya que será necesario introducir cada número de forma manual. Con los dos primeros botones incrementamos/decrementamos la cifra

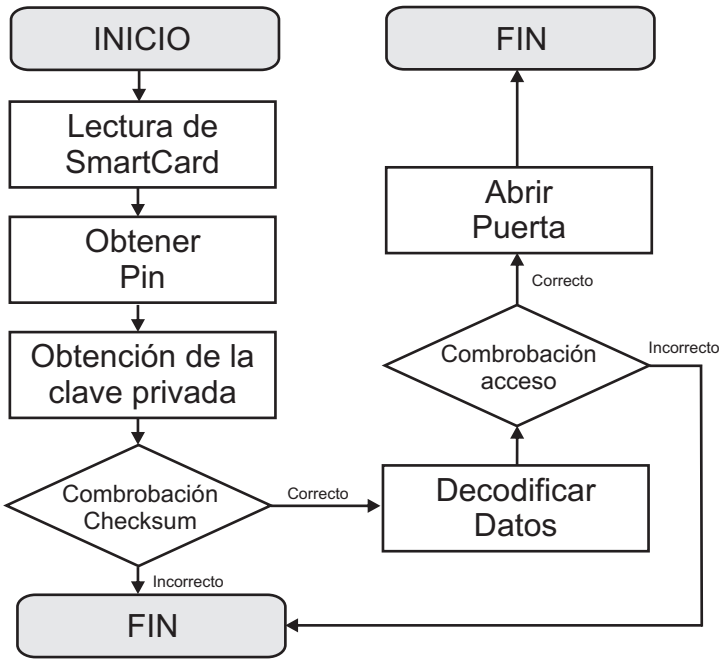


Figura 7.5: Rutina de Interrupción de tarjeta

y con el tercer botón pasamos a la siguiente. En la figura 7.6 se muestran ambas rutinas.

Una vez tenemos el pin que es un número de 4 cifras, este se codifica en binario usando 14bits. Posteriormente se combinará con una clave privada prefijada que contienen todas las puertas de 114bits para formar la clave privada de TEA de 128bits. De esta forma, aun conociendo la parte fija de la clave, no es posible obtener directamente la información. Este proceso es descrito en la figura 7.7.

Con esta clave ya es posible decodificar el contenido de la tarjeta pero como última comprobación de seguridad, y para investigar si los datos de la tarjeta han sufrido modificaciones no autorizadas o se han corrompido por algún tipo de interferencia electromagnética, se calcula el compendio CRC16 de todo el contenido de la tarjeta excepto los últimos 16bits que contendrán el compendio auténtico. Si al comparar el obtenido con el auténtico se descubre que estos son diferentes, automáticamente se deniega el acceso y se aborta la operación con el

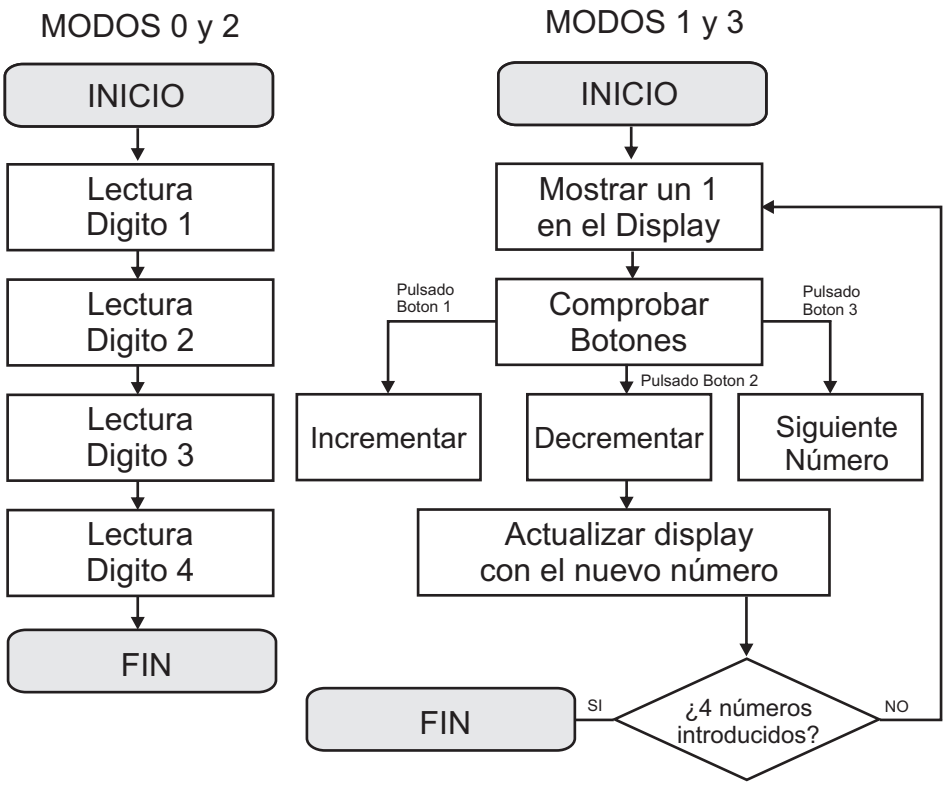


Figura 7.6: Rutina de Interrupción de captura de PIN

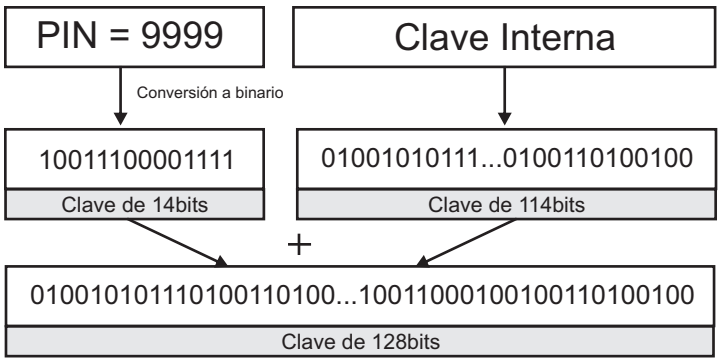


Figura 7.7: Construcción de la clave privada de TEA

mensaje “Tarjeta corrupta”. En la figura 7.8 se describe este otro proceso.

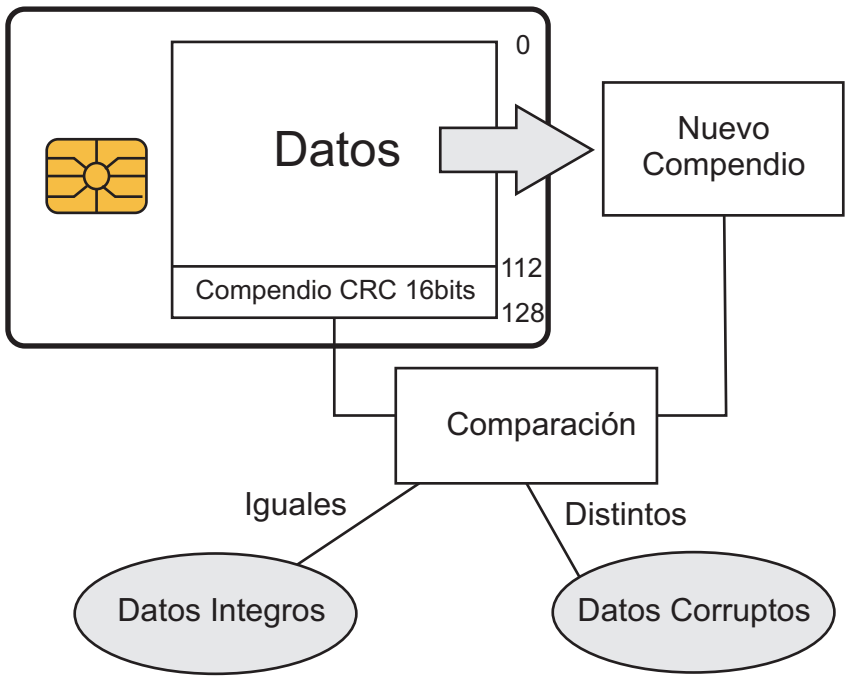


Figura 7.8: Comprobación de la integridad de los datos almacenados

La última comprobación que resta por hacer es comprobar si en la lista de puertas a las que el propietario de la tarjeta tiene acceso, se encuentra el código de la puerta a la que se quiere acceder. En caso afirmativo, se muestra el mensaje “Hola, nombre, Acceso Concedido”, en caso contrario el mensaje será simplemente “Acceso denegado” y se emitirán dos sonoros pitidos consecutivos. Tras esperar unos cinco segundos el sistema volverá al estado de StandBy. El nombre del usuario se obtiene de la misma tarjeta una vez decodificados los datos.

Este es el curso normal de operación para los modos 0 y 1, en el caso de los modos 2 y 3 hay que hacer algunas comprobaciones extras y bajar la barrera una vez se detecte que el coche ha pasado. Cuando se introduce una tarjeta, por la bobina podemos saber si efectivamente hay un coche esperando pasar por la barrera, en caso de no detectar

ningún coche se cancelará la operación. Además es necesario bajar la puerta una vez el coche salga para lo que se hace uso del detector de infrarrojos detectando el cambio de activado a desactivado. En la figura 7.9 se muestra el diagrama de flujo de los modos 2 y 3.

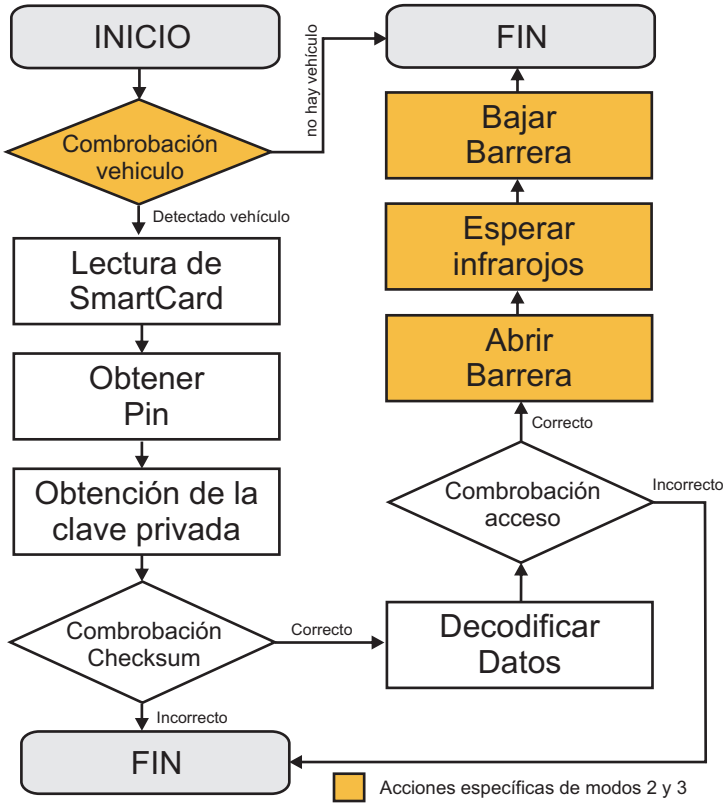


Figura 7.9: Rutina de Interrupción de tarjeta para modos 3 y 4

Se ha implementado un método especial para poder actualizar tanto la clave que comparten todas las puertas como el código de la puerta. Esto se hace con una tarjeta que incluye en los primeros 16bytes un código especial que permite acceder a las funciones de administración. Tras estos 16bytes, en los siguientes 16 se almacena la nueva clave, y en los siguientes el código de la puerta. Este método permite la actualización de los datos sin necesidad de acceder al interior del dispositivo para modificar la EEPROM.

7.2 El firmware en la puerta inteligente

Para este dispositivo, la puerta inteligente, el paradigma de programación es totalmente diferente. Como se comentó al comienzo de este capítulo se usará el lenguaje JAVA aprovechando que la plataforma Tini Net Interfaces incluye un entorno de ejecución de JAVA (*JRE*[1] del inglés, Java Runtime Environment).

En este caso, se usará un enfoque orientado a objetos, y se diseñarán clases para cada uno de los componentes hardware, de igual modo que se ha explicado en el caso de la puerta aislada. En la figura 7.10 se muestra un diagrama de clases con la estructura del firmware de este dispositivo.

Este software es un poco más complejo que el anterior, ya que no sigue una estructura lineal. Se puede dividir en tres procesos independientes que se ejecutarán concurrentemente en el dispositivo.

Proceso principal: Se encargará de inicializar el resto de procesos así como del control general del punto de acceso: lectura de la tarjeta, lectura del código de acceso e interacción con los diferentes dispositivos que se pueden conectar para la apertura de la puerta

Servidor Web: Este proceso se encargará de atender las peticiones http que reciba mediante el puerto 80 y proporcionar una página web con información de los últimos accesos.

Servidor de operación remota: Este proceso, que escucha conexiones entrantes por el puerto 51000, atenderá las peticiones que reciba de un applet según un protocolo muy sencillo descrito más adelante para permitir la interacción remota con algunas de las funciones del punto de acceso como el bloqueo y apertura de la puerta o el cambio de la configuración de la misma.

En la figura 7.11 se muestra un diagrama con estos tres procesos y con qué o quién interactúan.

[1] *JRE*: Ver definición en página 14

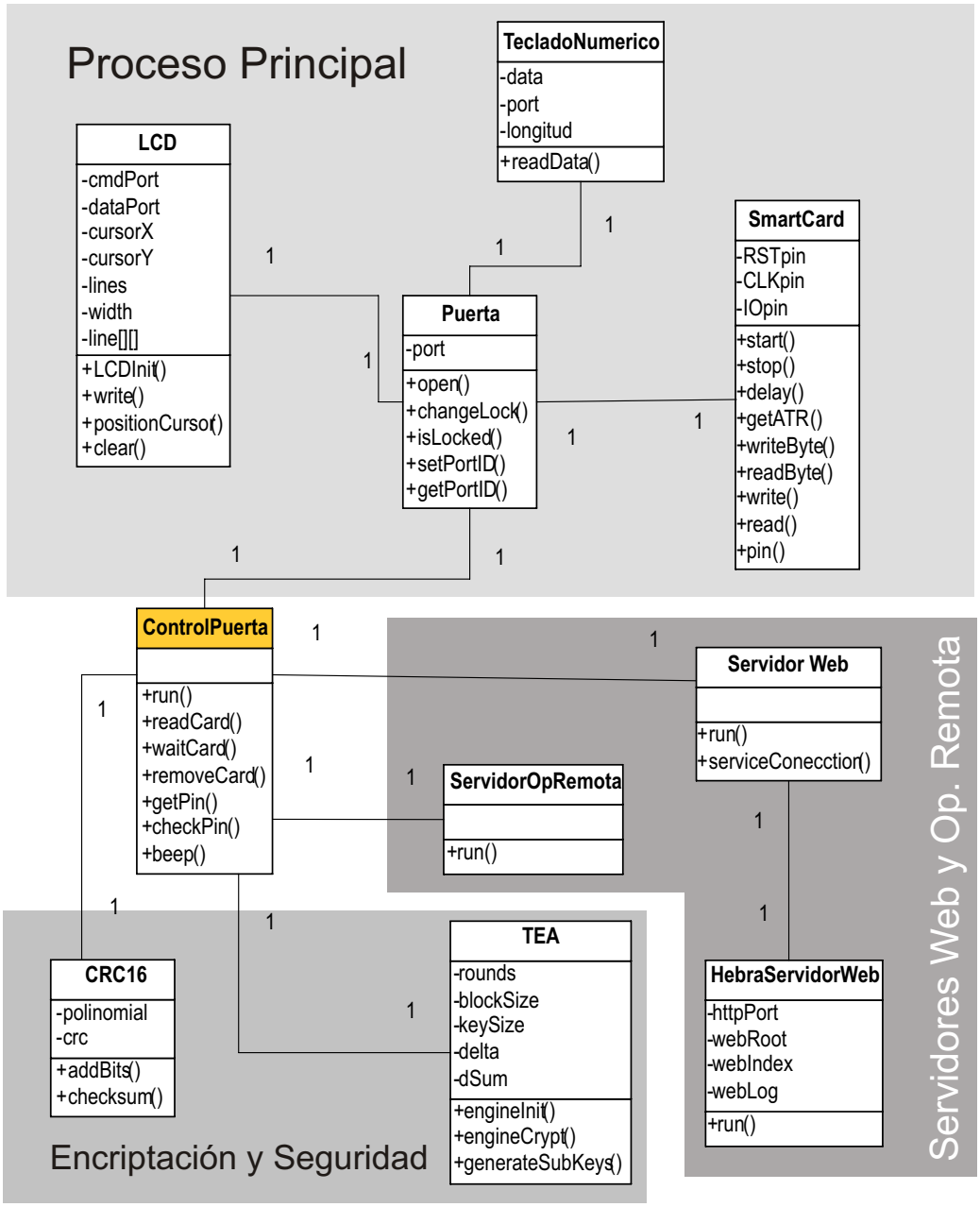


Figura 7.10: Diagrama de clases del firmware de la puerta inteligente

applet	Un applet es un componente de software que corre en el contexto de otro programa, por ejemplo un navegador web. El applet debe correr en un contenedor, que es proporcionado por un programa anfitrión, mediante un plugin, o en aplicaciones como teléfonos celulares que soportan el modelo de programación por applets. A diferencia de un programa, un applet no puede correr de manera independiente, ofrece información gráfica y a veces interactua con el usuario, típicamente carece de sesión y tiene privilegios de seguridad restringidos. Un applet normalmente lleva a cabo una función muy específica que carece de uso independiente. El término fue introducido en AppleScript en 1993.

7.2.1 Proceso principal

El proceso principal es el encargado de inicializar la aplicación, crear dos hebras para cada uno de los servidores y comenzar la ejecución del proceso principal. De todo esto se encarga la clase ControlPuerta. En el diagrama de clases de la figura 7.10 se listan los métodos de esta clase:

- run()
- waitCard()
- readCard()
- removeCard()
- getPin()
- checkPin()
- beep()

Este proceso es similar al caso de la puerta aislada, espera a que el usuario introduzca una tarjeta mediante el método waitCard(), una vez detectada esta, obtiene el pin del usuario mediante el metodo getPin() que lo lee del teclado usando la clase TecladoNumerico.

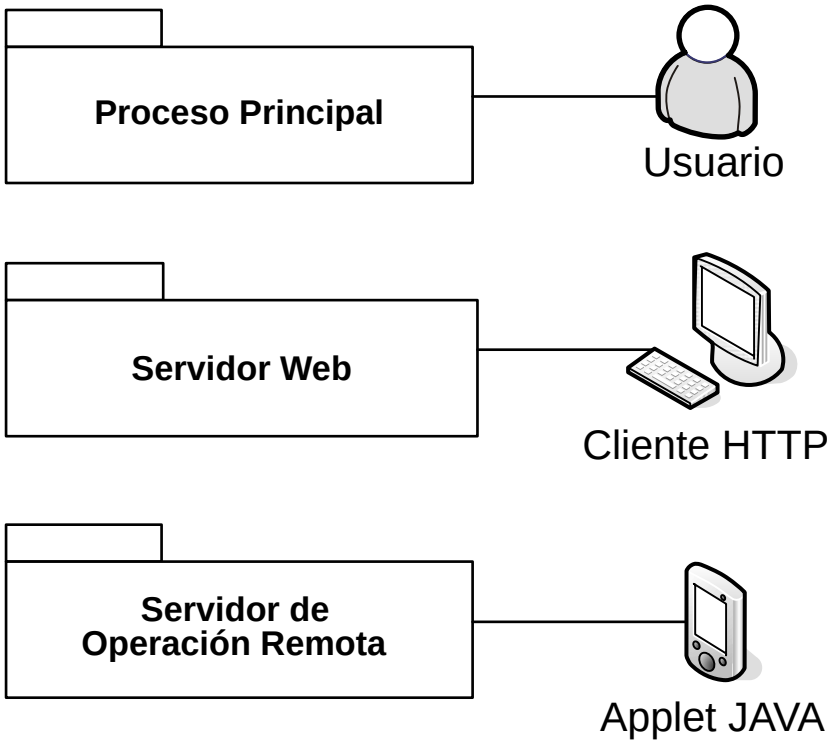


Figura 7.11: Procesos en la puerta inteligente

Para comprobar si el pin es correcto se usa el método checkPin() y en caso de que lo sea, se obtiene el contenido de la tarjeta. Este contenido es solo una cadena de bytes encriptados por lo que será necesario decodificarlos mediante la clase TEA.

Por último se comprueba que la tarjeta no haya sido modificada de forma externa con la clase CRC y se abre la puerta si todo es correcto. En la figura 7.12 se muestra este proceso de forma gráfica.

7.2.2 Servidor Web

Un servidor web es un programa que implementa el protocolo HTTP (hypertext transfer protocol).

Cabe destacar el hecho de que la palabra servidor identifica tanto al programa como a la máquina en la que dicho programa se ejecuta. Existe, por tanto, cierta ambigüedad en el término, aunque no será difícil diferenciar a cuál de los dos nos referimos en cada caso. En este artículo nos referiremos siempre a la aplicación.

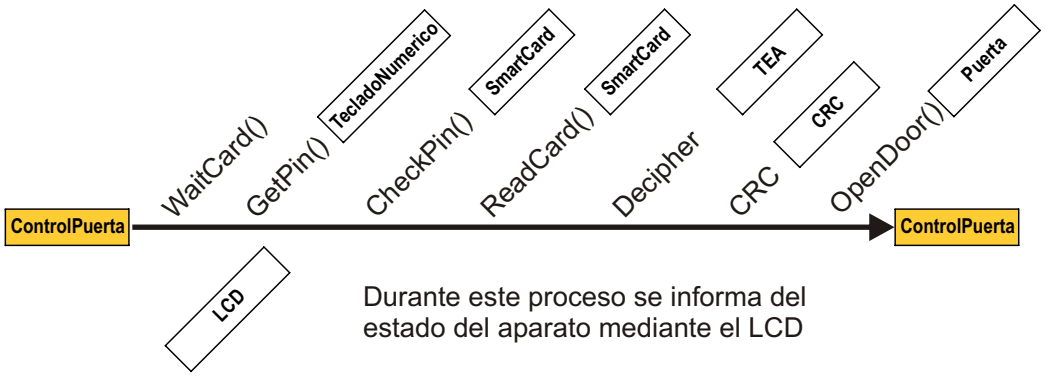


Figura 7.12: Proceso de lectura de tarjeta

HTTP	Potocolo diseñado para transferir lo que llamamos hipertextos, páginas web o páginas HTML (hypertext markup language): textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de sonidos.

Un servidor web se encarga de mantenerse a la espera de peticiones HTTP llevada a cabo por un cliente HTTP que solemos conocer como navegador. El navegador realiza una petición al servidor y éste le responde con el contenido que el cliente solicita.

El servidor web implementado sirve una página con un applet para controlar la puerta a distancia y una lista de los últimos accesos a modo de bitácora. Se ha diseñado como una hebra que se ejecuta concurrentemente con el proceso principal, de esta forma, las peticiones se procesan en tiempo real.

Por defecto este se ejecuta en el puerto 80, que es el puerto por defecto del protocolo HTTP.

HTML	El HTML, acrónimo inglés de HyperText Markup Language (lenguaje de marcas hipertextuales), es un lenguaje de marcación diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas web.

socket	Socket designa un concepto abstracto por el cual dos programas (posiblemente situados en computadoras distintas) pueden intercambiarse cualquier flujo de datos, generalmente de manera fiable y ordenada.

7.2.3 Servidor de operación remota

Además del servidor web, se ha implementado un protocolo para comunicarse con el dispositivo y poder reconfigurarlo de forma remota, desde cualquier dispositivo que pueda crear sockets *TCP/IP*[2] . Se ha diseñado con su futura ampliación en mente basándolo en números enteros:

0x01 Abrir puerta

0x02 Bloquear acceso

0xFF Obtener estado

De igual forma que el servidor web, este se ha implementado en una hebra que escucha en este caso en el puerto 51000.

[2] *TCP/IP*: Ver definición en página 7

8 Conclusiones y futuras líneas de trabajo

SARIC es el resultado de un complejo desarrollo, en el cual se han utilizado las últimas tecnologías. Desde el principio, se han aplicado técnicas de ingeniería y procesos de análisis y desarrollo de software muy novedosos como la programación extrema, metodología que se ha creído pertinente usar por razones de productividad. Esto, unido al uso minucioso de estándares y herramientas CASE, ha permitido que el proceso en general se haya presentado fluido, minimizando los problemas de comunicación entre los dos autores del proyecto.

Desde el principio se fijaron ciertas metas a nivel personal, aparte de la puramente didáctica, entre ellas el uso de herramientas libres o la portabilidad. En este sentido, se ha hecho todo lo posible por escribir un código capaz de ejecutarse en la mayor cantidad de sistemas operativos posible y gracias al uso de JAVA, esto no ha supuesto un problema demasiado grave. La portabilidad junto a la programación orientada a objetos y la disponibilidad de excelentes entornos de programación ha reafirmado la elección de JAVA como lenguaje de programación para la mayor parte del proyecto.

No obstante, ha sido necesario el uso de otros lenguajes de bajo nivel como C o Ensamblador a la hora de programar los dispositivos de acceso. El proceso de programación en estos dispositivos es algo más complicado que el de software más genérico, debido a que se deben tener en cuenta detalles precisos del funcionamiento electrónico de los componentes y temporizaciones de señales eléctricas. La ejecución y prueba del firmware contribuye a complicar aún más el desarrollo,

ya que es necesario transferir el software compilado a la plataforma hardware. En este punto el uso de simuladores ha facilitado la tarea permitiendo probar diferentes componentes antes de la fabricación del prototipo.

Debido a la naturaleza multidisciplinar del proyecto, y gracias a la colaboración interdepartamental ha sido posible aprender y aplicar conocimientos de muy diversas áreas. Además, desde un principio se encontró apoyo por parte del Vicedecano de Infraestructuras de la Facultad de Ciencias, interesado en la posibilidad de utilizar los propios carnés de estudiantes como llaves de acceso, lo que resultaría impracticable con sistemas comerciales cerrados. Además este apoyo quedó plenamente demostrado al ser cedida la base de datos de profesores de la Facultad de Ciencias facilitando que el proyecto se enfocase hacia una posible implementación futura en este centro. Esto propició iniciar un ritmo de trabajo serio y comprometido que se continuó hasta la finalización del proyecto.

El sistema funciona según lo esperado y cumple todos los puntos que se plantearon en el anteproyecto, mejorando en muchos aspectos la mayoría de ellos. No obstante puede ampliarse de muy diversas formas, lo que podría conducir a la creación de nuevos Proyectos Fin de Carrera que continuasen este trabajo. Algunas posibles líneas de trabajo podrían ser:

- Incorporación de nuevos dispositivos biométricos como lectores de retina o técnicos como el reconocimiento facial que permitiesen dotar al sistema de métodos de identificación más prácticos y potentes.
- Interfaces más accesibles para personas con alguna discapacidad como interfaces en lenguaje natural, teclados adaptados, lenguaje braille o avisos sonoros. (Figura 8.1)
- Mejoras en las capacidades de administración remota.
- Interfaces inalámbricas como sustitución a la actual *Ethernet*[1] .

[1] *Ethernet*: Ver definición en página 9

- Utilización de smartcards más potentes de forma que sea posible mezclar varios servicios en la misma tarjeta inteligente o que incluyan un chip propio de encriptación de datos.

- Protocolos de intercambio de datos con servidores capaces de realizar cálculos mas complejos como reconocimiento de voz

- Adaptaciones a otros entornos diferentes de la Facultad de Ciencia: hoteles, aparcamientos, etc.

- Uso del carné universitario como SmartCard de acceso.

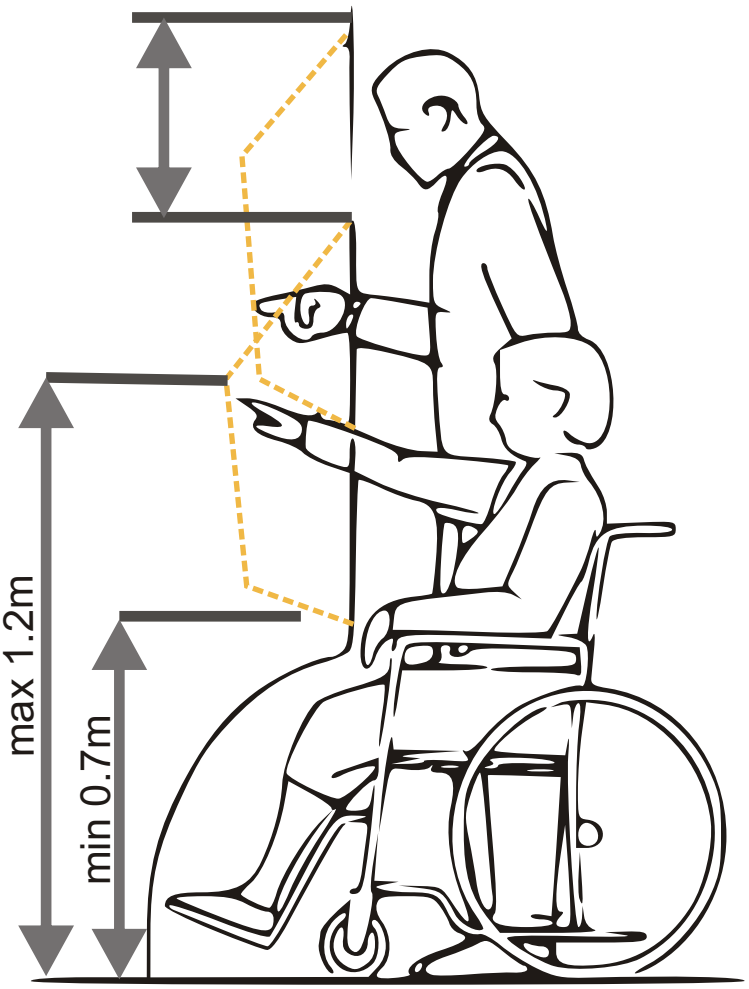


Figura 8.1: Estudio de accesibilidad para colocación de puntos de acceso

Por otro lado, el uso de las tarjetas inteligentes no tiene porqué limitarse a sistemas de acceso. Es posible desarrollar aplicaciones paralelas para éstas de forma que compartiesen los datos, por ejemplo, a nivel de la Universidad de Granada sería posible usarlas de forma simultánea como método de acceso a ciertos laboratorios, el pago en cafeterías, carné de biblioteca, consulta de expedientes, fotocopias, etc.

Figura 8.2: Resumen de diferente técnicas biométricas

A Manual de Usuario

SARIC es un sistema de control de accesos integral basado en tarjetas inteligentes, que incluye todo el hardware y software necesario para su puesta en funcionamiento. La instalación de los puntos de acceso debe de ser realizada por un profesional ya que implica tareas de albañilería y electricidad.

El sistema dispone de dos dispositivos de acceso diferentes: puertas inteligentes y puertas aisladas.

A.1 Gestor de SARIC

Para utilizar SARIC es necesario disponer de un ordenador personal que cumpla los siguientes requisitos:

- Sistema Operativo compatible con Java (Ms Windows, GNU/Linux, Solaris, Mac OS)
- 30MB de espacio libre en disco
- Monitor con resolución igual o superior a 800x600 y 16 bits de color
- Entorno de Ejecución de Java (*JRE*[1]) con la biblioteca multimedia Java Media Framework (JMF).
- Lector de Tarjetas Inteligentes

[1] *JRE*: Ver definición en página 14

- WebCam (en caso de ser administrador del sistema).

A.1.1 Uso de la Aplicación - Usuario común

Identificación

Al arrancar la aplicación, se le mostrará una ventana que le servirá para autentificarse en el sistema. Ver imagen A.1.

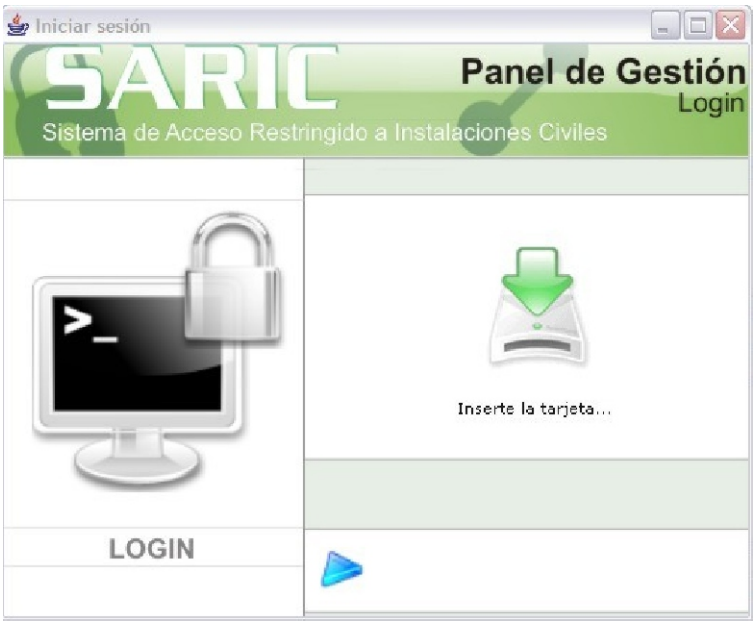


Figura A.1: Pantalla de identificación

Se le solicitará introducir la tarjeta y, si esta es reconocida, el código pin. Si ambos son válidos le será mostrado el menú principal. El sistema permite únicamente dos intentos erróneos seguidos. Al tercer error la tarjeta quedará invalidada. En la figura **??** se muestra un lector de tarjetas conectado a un PC portátil.

Menú Principal

El menú principal está dividido en tres paneles: el panel superior muestra el logo de la aplicación y es meramente decorativo. El panel izquierdo, se encuentra visible en todo momento y es el que permite navegar por la aplicación. El panel central cambia según la operación

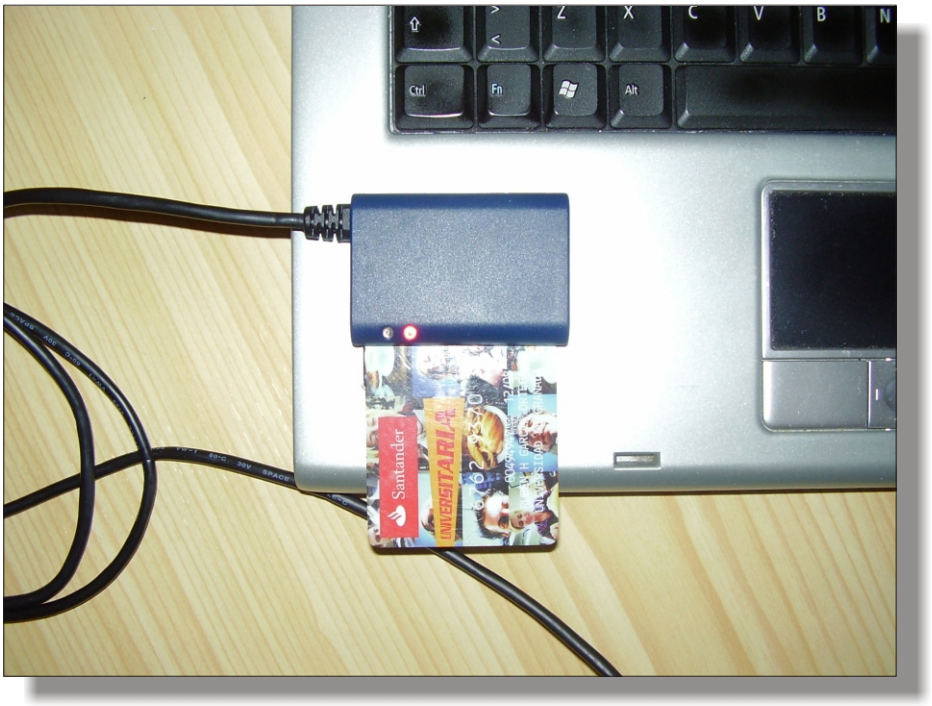


Figura A.2: Lector de tarjetas conectado a un pc portatil

a realizar mostrando las diferentes opciones con las que el usuario puede interactuar. En la figura **??** se puede ver el panel principal. Se muestra un panel de bienvenida y presentación de la aplicación para que usuarios noveles puedan empezar a usar esta sin necesidad de leer este manual, proporcionando en todo momento ayuda contextual.

Si la fecha de caducidad de la tarjeta del usuario ha expirado, el panel central cambia su apariencia indicándolo al usuario que contacte con el administrador. En la figura **??** se muestra esta situación.

A continuación se describen cada una de las 4 opciones principales del menu izquierdo:

Panel "Mis Datos"

Este panel muestra la información personal del usuario que ha accedido al sistema. Estos datos son meramente informativos y en la figura A.5 puede verse su aspecto. Para ver los permisos disponibles,

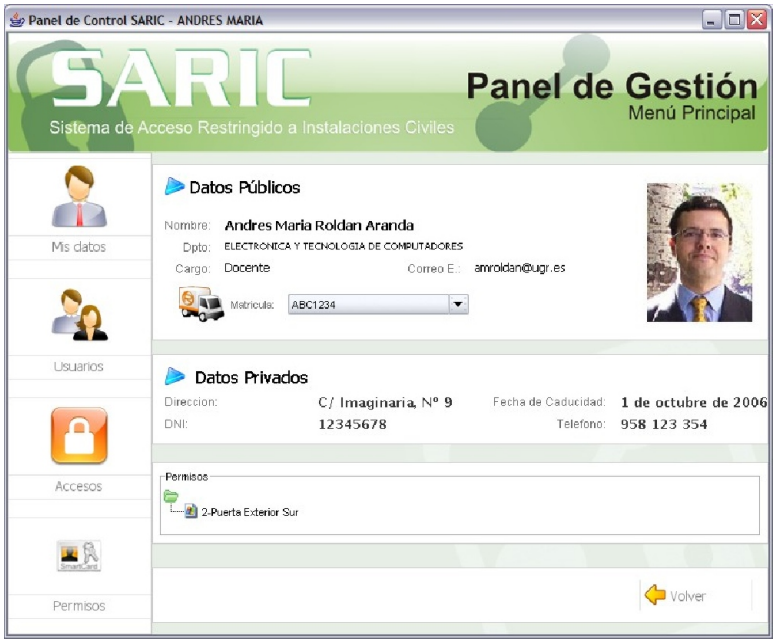


Figura A.3: Pantalla principal

Figura A.4: Situación de usuario caducado

basta con hacer doble clic en la pestaña de permisos. Si el usuario dispone de más de un vehículo asociado, puede consultarse desplegando la lista identificada como “Matricula”.

Figura A.5: Panel Mis Datos

Permisos

Antes de describir el panel es necesario conocer el sistema de permisos de la aplicación. Se dispone de tres tipos de permisos diferentes:

1. Permiso de acceso: Es el permiso más básico permitiendo únicamente el acceso. En el sistema se identifica por una pequeña carpeta de color gris.

2. Permiso para otorgar: Con este permiso, además del acceso, ofrece la posibilidad de otorgar dicho permiso a otro usuario. Este nivel está pensado para delegar responsabilidades en zonas concretas y no mantener todo bajo un único administrador. Se identifica con una carpeta de color amarillo.

3. Permiso para revocar: Este es el nivel superior, y además de los privilegios de los dos anteriores, incluye la posibilidad de revocar dicho permiso. Su color es el verde.

Este panel cumple dos funciones: por un lado permite ver los permisos disponibles y por otro, si se dispone de permisos de nivel amarillo o verde, otorgar/revocar los mismos a otro usuario. El panel izquierdo representa los permisos disponibles mientras que el derecho los otorgados al usuario seleccionado. Para hacer esto, es necesario conocer su DNI o disponer de su tarjeta inteligente. Tras introducir sus datos, bastará con pulsar en las flechas izquierda y derecha para otorgar o revocar el permiso seleccionado en cada panel.

En la figura A.7 se muestra este panel.

Figura A.6: Panel Permisos

A.1.2 Uso de la Aplicación - Administrador

Las siguientes opciones solo están disponibles si se dispone de privilegios de administrador, ya que pueden modificar datos de todos los usuarios del sistema.

Menú Usuarios

Desde el menú de usuarios (Ver figura **??**) se puede acceder a las acciones de administración de usuarios si se dispone de privilegios de

administrador pudiendo realizar las siguientes funciones:

Figura A.7: Menú usuarios

- Consultar los datos de cualquier usuario
- Añadir, modificar o eliminar usuarios
- Generar una nueva SmartCard con la fotografía del usuario
- Alterar las fechas de caducidad de las tarjetas de los usuarios

Panel para Añadir Usuarios En la figura A.8 se puede ver el panel de inserción de nuevos usuarios. En él se muestran campos para rellenar todos los datos del usuario:

- Nombre
- Apellido primero
- Apellido segundo
- Correo Electrónico (opcional)
- Departamento (opcional), el departamento puede seleccionarse de una lista desplegable) (Ver figura A.8)

Figura A.8: Panel Añadir Usuario: Lista de departamentos

- Cargo (opcional), el cargo puede seleccionarse de una lista desplagable (ver figura A.9)
- Fotografía
- Documento Nacional de Identidad (DNI)
- Pin o clave de acceso
- Dirección (opcional)
- Teléfono (opcional)
- Caducidad de la tarjeta. Es posible seleccionar la fecha de caducidad mediante un calendario desplegable de forma totalmente visual (Ver figura A.10)
- Vehículos

Una vez rellenados todos estos campos, para almacenarlos en la base de datos, basta con pulsar el botón “Guardar”. Si los datos son correctos, se mostrará un mensaje indicando el éxito de la operación. De lo contrario se informará del error cometido. Es posible borrar todos los campos con el botón borrar campos y escribir los datos en la tarjeta mediante el botón “Escribir Tarjeta”.

Figura A.9: Panel Añadir Usuario: Lista de Cargos

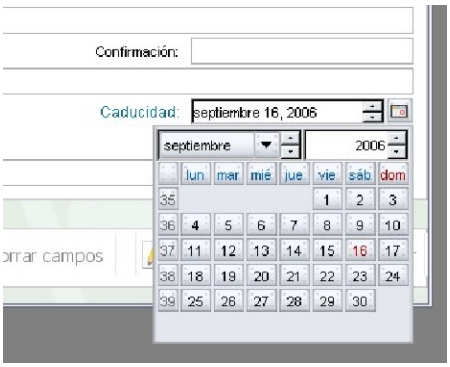


Figura A.10: Calendario desplegable

Es posible generar una tarjeta impresa para el usuario. Para ello es necesario pulsar el botón "Crear Tarjeta". Se mostrará un panel similar al de la figura A.11 con el nombre del usuario y el botón "Imprimir" para obtener una impresión de la tarjeta. Usando el botón "Capturar fotografía" se accede a un panel donde es posible, si se dispone de una webcam, capturar la foto del nuevo usuario *in situ*. Para ello se sigue el siguiente proceso (ver figura A.12)

Figura A.11: Panel Crear tarjeta

1. Pulsar el botón "Iniciar Cámara" y esperar a que esta esté lista
2. Cuando se vea una imagen de la persona de manera clara, pulsar el botón "Capturar Fotografía"
3. Mover el cuadro rojo, de forma que este esté centrado en la cara de la persona
4. Por último pulsar sobre volver para ver el resultado. En la figura A.13 se puede ver un ejemplo.

Panel para Buscar, Modificar o Borrar Usuario El panel está dividido en tres partes (Ver figura A.14):

Figura A.12: Panel Capturar Fotografía

Figura A.13: Panel Capturar tarjeta con una foto de ejemplo

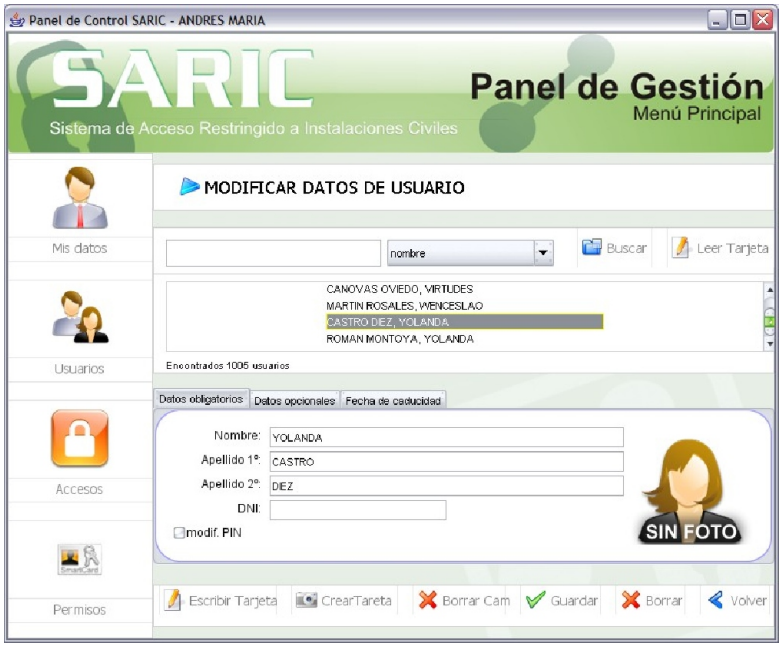


Figura A.14: Panel Buscar Usuario

- El panel superior sirve para buscar usuarios. Existen dos mecanismos diferentes para tal fin: si se dispone de la SmartCard del usuario que se desea consultar o modificar, el método más rápido es pulsar sobre "Leer tarjeta" e insertarla cuando se solicite. Si no se dispone de la SmartCard, puede realizar búsquedas por nombre, apellidos o DNI.
- El panel central muestra los resultados de la búsqueda realizada, indicando el número de coincidencias encontradas. Al pulsar sobre el nombre del usuario que se desea consultar, el formulario inferior mostrará sus datos.
- El panel inferior contiene tres pestañas que clasifican todos los datos del usuario. La primera pestaña muestra los obligatorios, la segunda (ver figura A.15) los opcionales y la tercera (ver figura A.16) permite alterar la fecha de caducidad mediante un calendario visual o botones que pueden ampliarla en 1 mes, 3 meses, 6 meses o un año.

El panel tiene funciones para "Crear tarjeta", "Guardar datos" y "Borrar usuario".

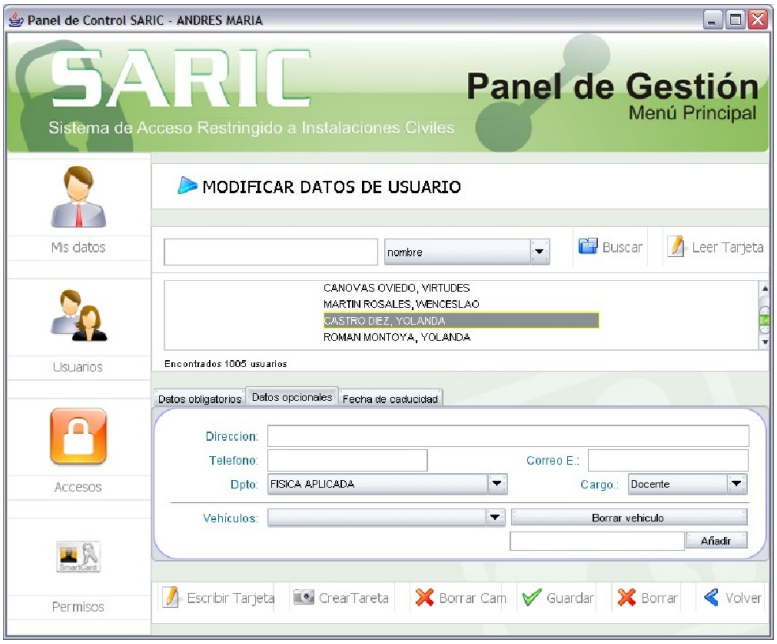


Figura A.15: Panel Buscar Usuario (Ficha datos opcionales desplagada)

Figura A.16: Panel Buscar Usuario (Ficha fecha de caducidad desplagada)

A.2 Puntos de acceso

Externamente ambos puntos de acceso operan de forma similar, utilizando los mismos mensajes. Cuando se inicia por primera vez, se muestran los siguientes mensajes:

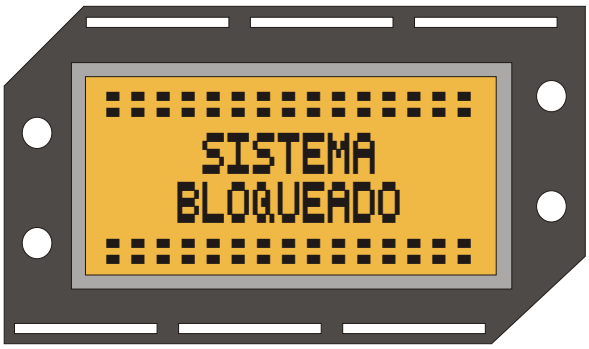
Seguidamente el sistema queda en estado de espera hasta que el usuario introduzca una tarjeta inteligente:

Cuando recibe la tarjeta, el sistema solicitará el código pin:

Si el código es incorrecto el sistema mostrará el siguiente mensaje:

En cambio, si es correcto, el dispositivo abrirá la puerta mostrando el siguiente mensaje:

Si algún administrador ha decidido bloquear la puerta, el mensaje mostrado será el siguiente:



A.3 Configuración del servidor

Lo primero es instalar un sistema gestor de bases de datos, en este proyecto se ha usado MySQL en su versión 5.0 y las instrucciones indicadas en este manual se corresponde con este sistema y para el sistema operativo Linux. No obstante es fácil adaptar todo el el sistema para usar cualquier otro SGBD como Oracle o Microsoft SQL Server.

A.3.1 Instalación y configuración del SGBD

En primer lugar es necesario obtener los paquetes para el servidor MySQL y el cliente MySQL para consola, como ejemplo se muestra su instalación para distribuciones con el sistema de paquetes apt, como Debian o Ubuntu.

1. Entrar al sistema como root y arrancar el demonio MySQL si este no esta ejecutándose

   
   ```
   >su
   ```

   
   ```
   >mysqld &
   ```

2. Si es la primera vez que se utiliza el servidor MySQL es imprescindible habilitar una contraseña para root, de lo contrario el sistema no seria seguro.

   
   ```
   >mysql -D mysql -u root -p
   ```

   (o bien si esto no funciona)

   
   ```
   >mysql -D mysql -u root
   ```

   
   ```
   >update user set
   password=PASSWORD('nueva_clave')
   where user='root';
   FLUSH PRIVILEGES;
   quit
   ```

   (nueva_clave se sustituirá por la clave de root. Así en nuestro ejemplo usaremos la clave svrpass de la siguiente forma: update user set password=PASSWORD('svrpass') where user='root';)

3. A continuación se crea una nueva base de datos para el uso de SARIC, nuestra saricbd, y un usuario al que se le permitirán los accesos a sus tablas.

   
   ```
   >mysqladmin -uroot -pPASSWORD
   create saricbd
   ```

   (el usuario ejemplo usa el login "cliente" y el password "clientepass")

A continuación entra a la base de datos MySQL) y ejecuta los siguientes comandos:

```
>insert into user
(host,user,password) values
('localhost','cliente',password('clientepass'));
>insert into user
(host,user,password) values
('%','cliente',password('clientepass'));
```

(se le permite el acceso a saricbd)

```
>insert into db values
('%','saricbd','cliente',
'Y','Y','Y','Y','Y','Y',
'Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y');
>quit
```

(se activan los cambios)

```
>mysqladmin -uroot -pPASSWORD
reload
```

4. Una vez que el usuario ha sido añadido, se accede a la base de datos mediante el cliente en consola "mysql" y se ejecuta el script para la generación e inicialización de la base de datos. Es necesario acceder desde el directorio donde se encuentre dicho archivo.

```
>mysql -ucliente -pclientepass
saricbd
```

```
>source saricbd.sql
>quit
```

B El algoritmo TEA

El Tiny Encryption Algorithm (TEA) o Algoritmo de Encriptación diminuto y sus variantes (XTEA; Block TEA, XXTEA) son cifradores de bloques célebres por su simplicidad de descripción y de implementación (normalmente unas pocas líneas de código) por lo que gozan de gran popularidad.

Los autores de TEA son David Wheeler y Roger Needham del Laboratorio de Informática de Cambridge y fue presentado por vez primera en 1994 en el Fast Software Encryption Workshop. No está sujeto a ningún tipo de patente.

TEA opera en bloques de mensajes de 64bits con una clave de 128 bits. Contiene una estructura de red Feistel con 64 rondas recomendadas, aunque los autores piensas que con solo 32 rondas es suficiente. Dos rondas Feistel de TEA se denominan **ciclo**. Posee una generación de claves extremadamente simple, mezclando todo el contenido de la clave de la misma manera para cada ciclo. En la figura B.1 se muestra un ciclo de TEA, en la figura B.2 se muestra el algoritmo en lenguaje C.

El algoritmo usa múltiplos de una "constante mágica´´, δ, obtenida de la proporción áurea, para asegurar que el encriptado en cada ciclo es diferente y prevenir así ataques basados en la simetría de las rondas; el valor preciso de δ es probamente poco importante, pero en el algoritmo TEA queda definido por:

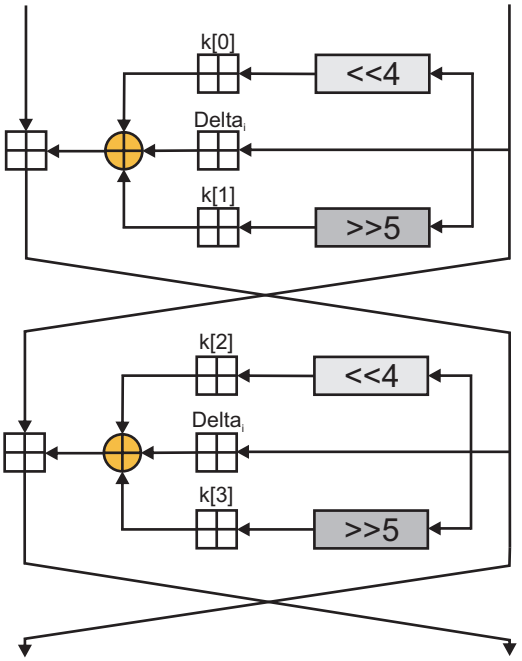


Figura B.1: Ciclo i del algoritmo TEA

```
void code(long* v, long* k)  {
  unsigned long y=v[0],z=v[1], sum=0,    /* inicialización */
  delta=0x9e3779b9, n=32 ;
  while (n-->0) {                        /* inicio del bucle */
    sum += delta ;
      y += (z<<4)+k[0] ^ z+sum ^ (z>>5)+k[1] ;
      z += (y<<4)+k[2] ^ y+sum ^ (y>>5)+k[3] ;
  }
  v[0]=y ; v[1]=z ;
}
```

Figura B.2: Código en C de la rutina de codificación del algoritmo TEA

$$\left[\left(\sqrt{5}-1\right)2^{31}\right]$$

TEA tiene algunas debilidades. La más notable es que padece de claves equivalentes: cada clave es equivalente a otras tres, y esto implica que la longitud de clave efectiva es solo de 126 bits. A consecuencia de esto, se han publicado diversas revisiones que hacen el algoritmo mas seguro: XTEA, BlockTEA y XXTEA orden cronológico.

B.1 XTEA

En respuesta a la publicación de estas debilidades, los diseñadores sugirieron una versión revisada de TEA a la que denominaron XTEA (puede aparecer en algunos textos cono tean). En la figura B.3 se muestra el funcionamiento del nuevo algoritmo.

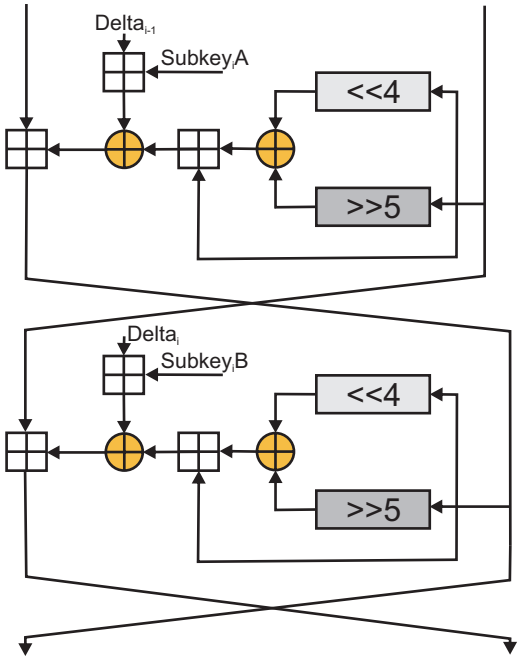


Figura B.3: Ciclo i del algoritmo XTEA

XTEA usa las mismas operaciones básicas que TEA (XOR, adición modulo 2^{32} y desplazamientos), pero el orden difiere considerablemente para prevenir ataques de clave-relacionada, las cuatro subclaves se mezclan de una forma menos regular y con una tasa mas lenta.

B.2 Block TEA y XXTEA

En el mismo informe que describe XTEA se introdujo además otra variante llamada Block TEA, que opera en bloques de longitud variable en múltiplos de 32bits. Este algoritmo aplica de forma secuencial la función de XTEA a cada palabra en el bloque y la combina mediante adición con su vecino. Esto se repite en varias rondas dependiendo

del tamaño del bloque, pero al menos 6 veces. Una ventaja de esta aproximación es que elimina la necesidad de modos de operación (CBC, OFB, CFB...), el cifrado se puede aplicar directamente a un mensaje. Además suele ser más eficiente que XTEA en mensajes largos. En la figura B.4 se ilustra este algoritmo.

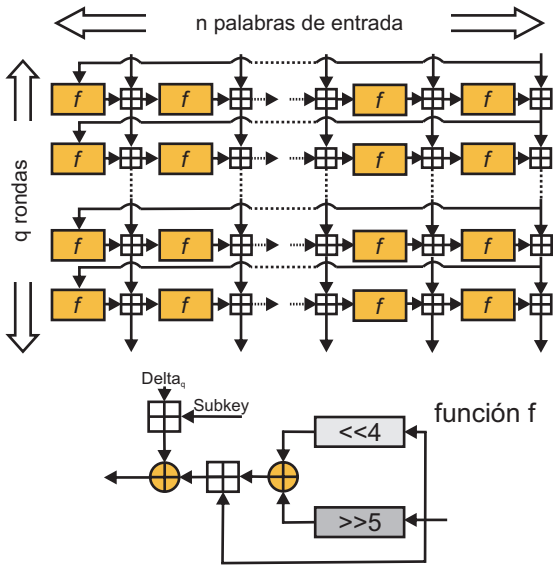


Figura B.4: La estructura de algoritmo Block TEA

B.3 XTEA

Cuando surgió un método de criptoanálisis de Block TEA, se publicó el algoritmo XXTEA. XXTEA es similar a Block TEA es estructura, pero hace uso de ambos vecinos cuando procesa cada palabra en el bloque. En vez de usar la función de XTEA, se introduce una nueva más compleja llamada MX que toma 2 entradas. En la figura B.5 puede verse la nueva estructura y la función MX.

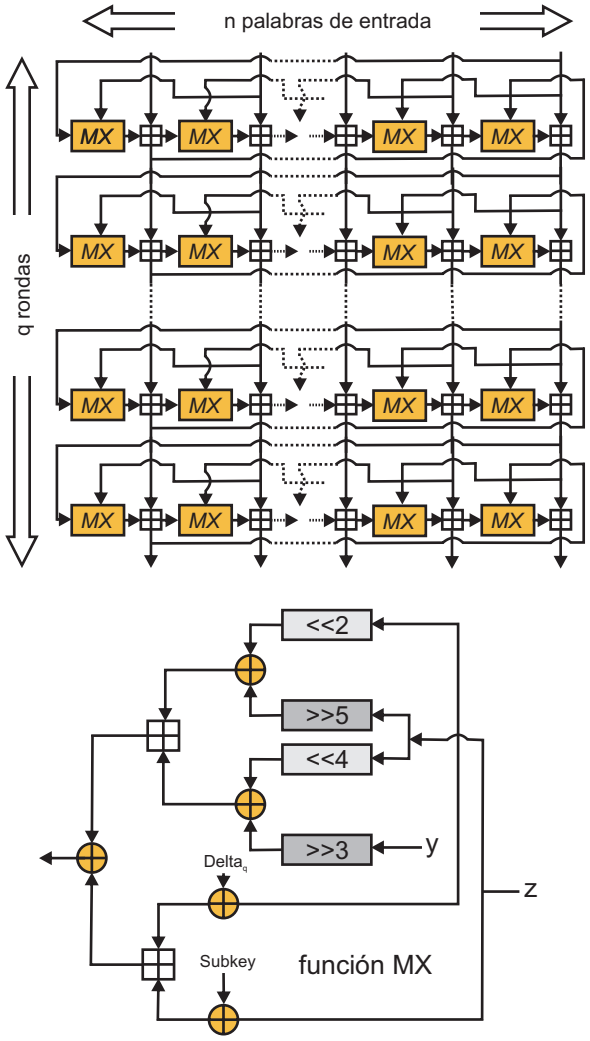


Figura B.5: La estructura de algoritmo XXTEA y la función MX

Bibliografía

[1] Ian Sommerville *Ingeniería del Sotware*. Pearson Educación, S.A., Madrid, 2005.

[2] James Newkirk, Robert C. Martin *Programación Extrema en la práctica*. Pearson Educación S.A., Madrid, 2002.

[3] *Wikipedia, la enciclopedia libre*. http://es.wikipedia.org .

[4] Christian Tavernier *Microcontroladores PIC*. Paraninfo, Madrid , 2001.

[5] José María Angulo Usategui *Microcontroladores PIC 16F87X*. McGraw-Hill, 1986.

[6] José M.ª Angulo Usategui, Ignacio Angulo Martínez, Susana Romero Yesa *Microcontroladores "PIC": diseño práctico de aplicaciones*. Mac Graw-Hill de España, Madrid, 2003.

[7] Eugenio Martín Cuenca, José Mª Angulo Usategui, Ignacio Angulo Martínez *Microcontroladores PIC : la solución en un chip*. Paraninfo, Madrid, 2001.

[8] Ed Sutter *Embedded systems firmware demystified*. Lawrence, Kan. : CMP Books, 2002.

[9] Herbert Schildt *Java 2 : manual de referencia*. Osborne Mac Graw-Hill, Madrid , 2001.

[10] Michael "Monty"Widenius, David Axmark, MySQL AB *MySQL reference manual: documentation from the source*. O'Reilly, Beijing, 2002.

[11] Vikram Reddy Andem *A cryptanalysis of the tiny encryption algorithm*. Master's thesis, Dept. of Computer Science, Graduate School of the University of Alabama, 2003.

[12] A. Biryukov and D. Wagner *Slide attacks*. In Lars Knudsen, editor, Fast software encryption: 6th International Workshop, FSE'99, Rome, Italy, March 24-26, 1999

[13] J. C. Hernández, J. M. Sierra, P. Isasi, and A. Ribagorda *Genetic cryptoanalysis of two rounds TEA*. Lecture Notes in Computer Science, 2331: 1024-1031, 2002.

[14] Julio César Hernández, Pedro Isasi, and Arturo Ribagorda *An aplication of genetic algorithms to the cryptoanalysis of one round tea*. In Proceedings of the 2002 Symposium on Artificial Intelligence and its Application, 2002.

[15] Julio César Hernández, José María Sierra, Pedro Isasi, and Arturo Ribargorda *Finding efficient distinguishers for cryptographic mappings, with an application to the block cipher TEA*. In Proceedings of the 2003 Congress on Evolutionary Computation, 2003.

[16] Julio César Hernández, José María Sierra, Arturo Ribagorda, Benjamín Ramos, and J. C. Mex-Perera *Distinguishing TEA from a random permutation: Reduced round versions of TEA do not have the SAC or do not generate random numbers*. In Proceedings of the IMA Int. Conf. on Cryptography and Coding 2001, pages 374-377, 2001.

[17] Deukjo Hong, Youngdai Ko, Donghoon Chang, Wonil Lee, and Jongin Lim *Differential cryptanalysis of XTEA*. Technical Report TR0313, Center for the Information Security and Technologies (CIST), Seoul, Korea, 2003a.

[18] Leslie Lamport *LaTex : A document Preparation System*. Addison-Wesley, 1986.

Índice alfabético