



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA



Escola Tècnica
Superior d'Enginyeria
Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Diseño e implementación de una máquina Arcade con Raspberry Pi y Arduino

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: Álvaro Roig Coves

Tutor: Xavier Molero Prieto

15 de junio de 2015

*" Ad augusta
per angusta "*

Resumen

Mediante este proyecto planteamos la resolución completa de un trabajo de ingeniería, cuyo objetivo es crear un dispositivo electrónico con el que emularemos las conocidas máquinas arcade de las salas recreativas. El sistema construido se usará por el Museo de Informática de la Universitat Politècnica de València, con el fin de difundir el patrimonio cultural informático de las últimas décadas del siglo XX.

Durante el transcurso de este proyecto abordaremos diferentes fases de desarrollo, desde el diseño conceptual de nuestra máquina arcade, hasta la implementación de su software o el montaje hardware que conlleva. De esta forma, combinaremos diferentes tecnologías open-hardware y open-software tales como los sistemas Arduino y Raspberry Pi, los cuales trabajarán en colaboración el uno con el otro con el fin de emular una máquina arcade real, obtener una funcionalidad atractiva y una experiencia óptima para los usuarios de nuestro dispositivo.

Destacamos algunas características funcionales de nuestro dispositivo como, por ejemplo, la autenticación de usuarios mediante sistemas de radiofrecuencia, el control de usuarios mediante la gestión de una base de datos, o la posibilidad de configuración remota a través de Internet.

Palabras clave: Máquina arcade, Arduino, Raspberry Pi, difusión del patrimonio informático, Museo de Informática.

Resum

Mitjançant aquest projecte plantegem la resolució completa d'un treball d'enginyeria, l'objectiu del qual és crear un dispositiu electrònic amb el qual emularem les conegudes màquines arcade de les sales recreatives. El sistema construït s'utilitzarà per part del Museu d'Informàtica de la Universitat Politècnica de València amb la finalitat de difondre el patrimoni cultural informàtic de les últimes dècades del segle XX.

Durant el transcurs d'aquest projecte abordarem diferents fases de desenvolupament, des del disseny conceptual de la nostra màquina arcade, fins a la implementació del seu programari o el muntatge del maquinari que el conforma. D'aquesta forma, combinarem diferents tecnologies *open-hardware* i *open-software* com ara els sistemes Arduino i Raspberry Pi, els quals treballaran en col·laboració l'un amb l'altre amb la finalitat d'emular una màquina arcade real, obtenir una funcionalitat atractiva i una experiència òptima per als usuaris del nostre dispositiu.

Destaquem algunes característiques funcionals del nostre dispositiu com, per exemple, l'autenticació d'usuaris mitjançant sistemes de radiofreqüència, el con-

trol d'usuaris mitjançant la gestió d'una base de dades, o la possibilitat de configuració remota a través d'Internet.

Paraules clau: Màquina arcade, Arduino, Raspberry Pi, difusió del patrimoni informàtic, Museu d'Informàtica.

Abstract

Through this project we propose the complete resolution of an engineering work, which aims to create an electronic device to emulate the famous arcade machines. The built system will be used by the Museum of Informatics at the Universitat Politècnica de València, in order to spread the computing cultural legacy of the last decades of the twentieth century.

During the work on this project we will address different stages of development, from conceptual design of our arcade machine until the implementation of its software or its hardware installation. In this way, we will combine different open-hardware and open-software technologies like Arduino and Raspberry Pi systems, which will work in cooperation with each other in order to emulate a real arcade machine, getting an attractive functionality, and also an optimal experience for the users of our device.

We highlight some functional characteristics of our device as the user authentication by radio frequency systems, the control of users through a database, or the possibility of remote configuration through Internet.

Keywords: Arcade machine, Arduino, Raspberry Pi, computer legacy, Museum of Informatics.

Índice general

1. Motivación y objetivos del trabajo	1
1.1. Motivación	1
1.2. Objetivos	2
1.3. Estructura de la memoria	3
1.4. Uso de la bibliografía	3
2. Introducción a las máquinas arcade	5
2.1. Los inicios de las máquinas arcade	5
2.2. La Edad de Oro de las máquinas arcade	6
2.3. Máquinas recreativas arcade españolas	8
3. Elementos de diseño	13
3.1. Raspberry Pi	13
3.1.1. Hardware	14
3.1.2. Software	15
3.1.3. Especificaciones técnicas	17
3.2. Arduino	17
3.2.1. Hardware	19
4. Diseño de la máquina arcade	21
4.1. Presupuesto	21
4.2. Computador central	22
4.2.1. Sistema operativo	23
4.2.2. Mecanismo de alimentación	24
4.3. Emulador arcade	25
4.3.1. MAME	26

4.3.2. Configuraciones del emulador	26
4.4. Interconexión de controles	28
4.4.1. Emulación de controles	31
4.5. Identificación de usuarios	33
4.5.1. Tecnología RFID	33
4.5.2. RFID ISO Estándar	34
4.5.3. Montaje del sistema mediante Arduino	37
4.5.4. Identificación de transpondedores	37
4.5.5. Instalación de la base de datos	39
4.5.6. Comunicación con RaspberryPi	43
4.6. Gestión remota	46
4.6.1. Diseño del portal de administración remoto	46
4.6.2. Instalación de servidores en la RaspberryPi	52
4.7. Inicio automático del sistema	57
5. Construcción de la máquina arcade	59
5.1. Laboratorio de trabajo	59
5.2. Detalles de los componentes	60
6. Conclusiones	65
6.1. Contratiempos y problemas sufridos	65
6.2. Consideraciones finales	66
Bibliografía	67
Apéndice	70
A. Código inicio del sistema	71
A.1. Script de inicio	71
B. Código gestión de usuarios	77
B.1. Obtención de identificadores	77
B.2. Edición de objetos en la base de datos	79
B.3. Comunicación con la base de datos	81
B.4. Comunicación con Arduino	84

C. Código del portal de administración telemática	87
C.1. Página de acceso	87
C.2. Página de inicio	91
C.3. Conexión a la base de datos	109
C.4. Cerrar sesión de usuario	109

Índice de figuras

2.1. Galaxy Game (1971)	6
2.2. Publicidad de Computer Space (1971) y Pong (1972)	7
2.3. Juego arcade Space Invaders (1978)	8
2.4. Juego arcade Pacman (1980)	9
2.5. Juego arcade Donkey Kong (1981)	10
2.6. Logo de la empresa española Gaelco	10
2.7. Juego arcade Big Karnak (1991)	11
2.8. Juego arcade World Rally (1993)	11
3.1. Raspberry Pi model B	15
3.2. Logos de Moebius, Raspbian, ArchLinux, y Pidora	16
3.3. Arduino Uno Rev.3 y Arduino Mega 2560 Rev.3	19
3.4. Esquema del Arduino Mega 2560 Rev.3	20
4.1. Presupuesto de los materiales necesarios	22
4.2. Logo de Raspbian	24
4.3. Mecanismo de alimentación de la Raspberry Pi	25
4.4. Logo del emulador MAME	26
4.5. Interfaces originales del emulador MAME	28
4.6. Interfaces editadas del emulador	28
4.7. Joystick arcade	29
4.8. Botones pulsadores arcade	29
4.9. Pulsador <i>microswitch</i>	30
4.10. Puertos <i>GPIO</i> de la Raspberry Pi Modelo B	31
4.11. Cableado de los controles arcade	32
4.12. Lector y transpondedor RFID	34

4.13. Espectro de frecuencias	36
4.14. Componentes necesarios para el montaje RFID	38
4.15. Montaje Arduino RFID	40
4.16. Logo de SQLite	40
4.17. Estado de la tabla <i>identifications</i>	43
4.18. Sistema de autenticación	45
4.19. Página de acceso al portal de trámites telemáticos	47
4.20. Página de inicio del portal de trámites telemáticos	48
4.21. Menú lateral del portal de trámites telemáticos	49
4.22. Menú superior del portal de trámites telemáticos	49
4.23. Acciones asociadas a la gestión de identificadores	50
4.24. Listado de identificadores de radiofrecuencia	50
4.25. Dar de alta un nuevo identificador de radiofrecuencia	51
4.26. Seleccionar un identificador para ser modificado	51
4.27. Modificar la información asociada a un identificador	52
4.28. Acciones asociadas a la gestión de ROMs	53
4.29. Listado de las ROMs presentes en el sistema	53
4.30. Formulario para añadir una ROM en el sistema	54
4.31. Reporte de la subida de una ROM en el sistema	54
4.32. Eliminar ROM del sistema	55
4.33. Tabla de <i>MySql</i> para guardar credenciales de acceso	56
4.34. Interfaz de consola para autenticar al usuario	58
5.1. Laboratorio de trabajo	59
5.2. Sistema de autenticación Arduino	60
5.3. Módulo RFID Arduino	60
5.4. Tarjetas de radiofrecuencia utilizadas	61
5.5. Raspberry Pi utilizada	61
5.6. Sistema de control arcade	62
5.7. Sistema de alimentación de la Raspberry Pi	62
5.8. Montaje final del hardware	63

Índice de tablas

3.1. Comparativa entre modelos de Raspberry Pi	17
3.2. Características técnicas del Arduino Mega 2560 Rev.3	20
4.1. Líneas a añadir en el fichero de configuración	27
4.2. Sección del código <code>retrogame.c</code>	33
4.3. Órdenes para la instalación de SQLite	41
4.4. Órdenes para la creación de la tabla <i>identifications</i>	42
4.5. Instrucciones para la inserción de elementos en la tabla <i>identifications</i>	43
4.6. Órdenes para crear y dar permisos al grupo de trabajo que utiliza <i>Apache</i>	55
4.9. Órdenes para crear ficheros de configuración	55
4.7. Órdenes para actualizar los repositorios e instalar <i>Apache</i>	56
4.8. Órdenes para instalar <i>MySQL</i> y <i>PHPMyAdmin</i>	56

CAPÍTULO 1

Motivación y objetivos del trabajo

En este primer capítulo exponemos las razones que nos han conducido a la elaboración de este trabajo, así como los objetivos que pretendemos alcanzar, y una pequeña descripción de los capítulos que forman la memoria del trabajo realizado.

1.1 Motivación

El Museo de Informática¹ de la Escola Tècnica Superior d'Enginyeria Informàtica de la Universitat Politècnica de València fue inaugurado el 11 de diciembre del año 2001 y ofrece un fondo museográfico que permite el recorrido histórico a través de las últimas décadas de la informática.

Cabe destacar el reconocimiento de su actividad por parte de la Conselleria d'Educació, Cultura i Esport de la Generalitat Valenciana, que el 13 de mayo de 2013 lo incluyó como museo oficial de la Comunitat Valenciana² así como su incorporación dentro del Consejo Internacional de Museos (ICOM) de la UNESCO en el año 2015.

Dentro de las actividades culturales que ofrece, tales como la visita a su exposición permanente, exposiciones temporales o conferencias, incluye un apartado dedicado a la educación. En este ámbito se encuentra la realización de visitas guiadas para alumnos de educación primaria y secundaria, bachillerato y ciclos formativos, así como talleres de retroprogramación para acercar a los visitantes tanto la historia de la informática como despertar en ellos el interés por la programación. Dentro de esta visión, el Museo de Informática ofrece un apartado más de actividades, orientadas al mundo de la programación y la educación, por las cuales puedan acercar a sus visitantes, muchos nacidos en la era de las nuevas

¹Página web disponible en <http://museo.inf.upv.es>.

²Resolución publicada en el DOCV, el 28 de mayo de 2013.

tecnologías, la opción no solo de entender estas tecnologías sino de usarlas como herramientas para crear y expresarse.

Además, debido a su constante evolución y creciente auge, el museo se plantea establecer una sección orientada a la difusión del conocimiento de los videojuegos clásicos de la década de los años ochenta, lo que ha originado el planteamiento de este trabajo.

1.2 Objetivos

Uno de los elementos de distracción más férreamente acogidos por el ser humano, han sido y son, los videojuegos. El origen de estos sistemas computacionales destinados al entretenimiento se encuentra en las máquinas recreativas. Dichas máquinas son uno de los elementos más representativos de los famosos salones recreativos de los años ochenta y noventa, a los que la mayoría de personas acudían con fines lúdicos.

En los salones recreativos podíamos encontrar tanto máquinas de videojuegos arcade, como juegos de mesa y de azar; los cuales, enmarcados en la sociedad de las últimas décadas del siglo XX, debido a su creciente auge, tenían una importancia relevante en los ámbitos económico y cultural.

En este trabajo nos disponemos a diseñar e implementar una de estas conocidas máquinas arcade, con la peculiaridad de hacer uso de componentes más modernos, tales como microcontroladores y dispositivos empotrados de las familias Arduino y Raspberry Pi. Además, nuestra máquina arcade será capaz de emular más de un videojuego, de tal forma que podremos disponer de varios juegos en un mismo dispositivo.

Como elementos distintivos del dispositivo podemos destacar que ofreceremos funcionalidades tales como la identificación de usuarios mediante tarjetas de radiofrecuencia, o la implementación de un portal de gestión telemático desde el cual los administradores de la máquina arcade podrán realizar todo tipo de trámites asociados a la gestión del dispositivo.

Finalmente, cabe destacar como objetivo adicional, la inclusión de esta máquina arcade en el Museo de Informática de la Universitat Politècnica de València, el cual financia el proyecto, y ofrecerá dicho dispositivo como atractivo para las visitas guiadas al museo, en las cuales los visitantes podrán hacer uso de la máquina arcade, así como descubrir e interactuar con los componentes que la forman.

1.3 Estructura de la memoria

El presente trabajo se compone de seis capítulos, y en esta sección, nos disponemos a realizar una breve descripción del trabajo realizado en cada uno de ellos.

- ★ **Capítulo 1, Motivación y objetivos del trabajo:** En este capítulo se expone la motivación por la cual se ha llevado a cabo este trabajo, así como los objetivos esenciales a abarcar.
- ★ **Capítulo 2, Introducción a las máquinas arcade:** Este capítulo está destinado a realizar un pequeño trabajo de indagación histórica acerca de las máquinas arcade, enmarcándolas dentro de la época en la que presentaron su mayor auge.
- ★ **Capítulo 3, Elementos de diseño:** En esta parte mostramos una descripción detallada de los componentes que vamos a utilizar con el fin de construir nuestra máquina arcade.
- ★ **Capítulo 4, Diseño de la máquina arcade:** Este capítulo se centra en el diseño de las estructuras hardware que componen la máquina arcade, así como el desarrollo software necesario para ofrecer todas sus características funcionales y su correcta configuración. Además, se presenta un presupuesto aproximado del coste asociado a la elaboración del proyecto.
- ★ **Capítulo 5, Construcción de la máquina arcade:** Aquí mostramos, mediante una serie de imágenes tomadas en el laboratorio de trabajo, el conjunto de montajes y pasos seguidos a través del proceso de construcción de la máquina arcade.
- ★ **Capítulo 6, Conclusiones:** El último capítulo recoge las principales conclusiones obtenidas, así como un breve resumen del conjunto de funcionalidades implementadas en nuestro dispositivo.

Finalmente, cabe destacar la presencia de varios anexos al final de la memoria, en los cuales se incluyen los códigos de más relevancia implementados durante la consecución del trabajo.

1.4 Uso de la bibliografía

En esta sección realizamos una breve descripción de los materiales bibliográficos utilizados, así como su uso y relación en cada uno de los apartados del trabajo.

- Inicialmente, con el fin de situarnos dentro del contexto histórico de las máquinas arcade, hacemos uso de la referencia bibliográfica [7], mediante la cual realizamos un pequeño trabajo de análisis histórico acerca de estos dispositivos.
- Uno de los objetivos de este trabajo es construir una máquina arcade con componentes más modernos que los utilizados para este propósito en la época de mayor auge de dichos dispositivos, con tal de realizar una descripción de las características de los componentes que usaremos, utilizamos las referencias [2, 4, 12, 14, 15, 16, 18].
- Además, para construir nuestro sistema de autenticación, necesitamos conocimientos sobre la tecnología RFID (*Radio Frequency IDentification*), por lo que hacemos uso de las referencias bibliográficas [1, 8, 10, 17].
- Por otra parte, con el fin de almacenar los identificadores de radiofrecuencia en la máquina arcade, precisamos de información referente a un sistema de base de datos seguro, por lo que hacemos uso de la referencia [11]; y para conectar los controles arcade al sistema necesitamos información referente a sus mecanismos físicos, por lo que utilizamos la referencia [13].
- Finalmente, para lograr administrar la máquina arcade de forma remota, necesitamos instalar una serie de servidores y diseñar un portal online, por lo que hacemos uso de las referencias bibliográficas [3, 5, 6, 9].

CAPÍTULO 2

Introducción a las máquinas arcade

Arcade es el término genérico de las máquinas recreativas de videojuegos disponibles en lugares públicos de diversión, centros comerciales, restaurantes, bares, o salones recreativos especializados. Son similares a los *pinballs* y a las tragamonedas o máquinas tragaperras de los casinos, pero no son juegos de azar ni de apuestas, ya que se basan en la pericia del jugador.

2.1 Los inicios de las máquinas arcade

Entre algunas de las más famosas máquinas arcade que han existido, podemos destacar las siguientes.

Galaxy Game (1971)

El Galaxy Game, mostrado en la Figura 2.1, fue programado por Bill Pitts y Hugh Tuck, y es el primer juego comercial del que se tiene constancia. El juego se basaba en dos naves espaciales que tratan de destruirse mutuamente, y que se ven afectadas por el campo gravitatorio de una estrella. Se instaló en la Universidad de Stanford en septiembre de 1971, dos meses antes del lanzamiento de Computer Space. Al principio solo se creó una máquina, aunque posteriormente se fabricarían algunas más y el juego sería adaptado a multitud de sistemas domésticos. La máquina costaba 20000 \$, y el juego permaneció unos años en el campus formando colas de más de una hora para jugar hasta que en mayo de 1979 fue retirado. Ahora reside en el *Computer Museum* de California.



Figura 2.1: Galaxy Game (1971)

Computer Space (1971)

Computer Space no sería la primera máquina arcade comercial, pero sí la primera fabricada a gran escala. En el juego se controlaba un cohete que podía disparar a platillos voladores, y fue lanzado en noviembre de 1971 de la mano de la compañía Nuttin Associates. La máquina la diseñó Nolan Bushnell y Ted Dabney. Con el dinero obtenido fundarían su propia compañía, Atari. A pesar del relativo éxito de Computer Space, por culpa de su control confuso, comenzaba a asentarse un nuevo tipo de negocio que daría muchos beneficios a las compañías en los años posteriores.

Pong (1972)

Un año más tarde Atari se estrenaba con el arcade PONG, el cual era un videojuego que simulaba una partida de tenis de mesa, en la que los jugadores tienen que devolverse mutuamente una pelota. Nolan Bushnell situó la primera máquina en una gasolinera local y cuando volvió a ver qué tal había ido, la máquina ya no funcionaba. Estaba completamente llena de monedas. A diferencia de Computer Space, el PONG tenía un manejo muy sencillo y todo el mundo podía jugar y divertirse desde la primera pantalla. El éxito de PONG fue enorme, popularizó los videojuegos y asentó definitivamente a las máquinas arcade como un modelo de negocio rentable.

2.2 La Edad de Oro de las máquinas arcade

Con Space Invaders daría comienzo la Edad de Oro de las máquinas arcade, que duraría aproximadamente desde 1979 hasta 1984. Un período en el que los nuevos avances informáticos darían lugar a gráficos más realistas y mejores



Figura 2.2: Publicidad de Computer Space (1971) y Pong (1972)

sonidos. Y estos avances serían aprovechados por parte de los desarrolladores para realizar una serie de clásicos que sentarían las bases para todos los juegos posteriores inaugurando géneros que prevalecen hoy en día. A continuación exponemos tres de los juegos más representativos de esta época.

Space Invaders (1978)

Diseñado por Toshihiro Nishikado en 1978 para Taito, Space Invaders supondría una revolución tanto en las máquinas arcade como en los sistemas de entretenimiento doméstico. En un principio, Space Invaders iba a ser un videojuego bélico, sin embargo la idea se rechazó dada la dificultad de animar los tanques y aviones. Toshihiro buscó entonces inspiración en el libro *La guerra de los mundos*¹, y así creó el matamarcianos por excelencia.

El juego incluía una nueva forma de jugar basada en puntos, un aumento progresivo de la dificultad, y un marcador de máxima puntuación que registraba las iniciales de los jugadores. Todo unido incentivaba a los jugadores a superarse a sí mismos y a los demás.

Pacman (1980)

Pacman nació de la mano de un diseñador de Namco llamado Tohru Iwatani mientras comía pizza. El nombre original de Pacman es Puck-man (que en japonés significa comer), pero al entrar en Estados Unidos procedente de Japón y de la mano de Midway, su nombre fue cambiado debido a las similitudes con las palabras *puck* y *fuck*. En el juego el jugador controlaba un círculo amarillo al que le faltaba un sector (por lo que parecía tener boca), mediante el cual debía ir

¹La guerra de los mundos es una novela de ciencia ficción escrita por Herbert George Wells y publicada por primera vez en 1898, que describe una invasión marciana a la Tierra.



Figura 2.3: Juego arcade Space Invaders (1978)

comiendo los puntos situados a lo largo de un laberinto, mientras evitaba encontrarse con unos fantasmas que rondaban por el mismo.

Donkey Kong (1981)

En el año 1981 Nintendo lanzó el arcade Donkey Kong. En el juego controlábamos a Jumpman, un personaje que evolucionaría hasta transformarse en Mario, y nuestro objetivo sería salvar a Pauline de las manos del gorila gigante Donkey Kong. El juego fue el primer trabajo de diseño encargado a Shigeru Miyamoto por petición de Hiroshi Yamauchi (presidente de Nintendo), y formaba parte de una serie de intentos por parte de la compañía nipona para introducirse en el mercado americano.

Este juego estaba inspirado en la película King Kong (1933, *Merian C. Cooper y Ernest B. Schoedsack*) y en el arcade Popeye, y sería uno de los primeros exponentes del género plataformas.

2.3 Máquinas recreativas arcade españolas

Si los situamos en su contexto, a mediados de la década de los 80, en la época en la que estaban realizándose algunos de los mejores juegos que hemos visto en los salones, los títulos de las compañías españolas suponen poco más que algunos ejemplos de segunda fila, por buenas que puedan ser algunas de sus propuestas. Hacía falta algo más, y ese algo vino, en la mayoría de casos, de la decadencia de



Figura 2.4: Juego arcade Pacman (1980)

las desarrolladoras que a finales de los ochenta y primeros noventa luchaban por adaptarse al nuevo panorama del mundo del videojuego. Los ordenadores de 16 bits eran el nuevo horizonte, con una potencia radicalmente superior a la que podía ofrecer, por ejemplo, el viejo Sinclair ZX Spectrum. Pero también suponían un reto para el que muchas de nuestras compañías demostraron no estar preparadas, ya que llevaban aparejado un cambio enorme en la manera de desarrollar un videojuego y, sobre todo, en el coste del mismo. Una tras otra, todas ellas fueron languideciendo ante los nuevos retos, cada una con sus propios motivos. Pero hubo una de ellas que consiguió reconvertirse con éxito y pasar a formar parte de un todo mucho más grande, que llevaría al videojuego español por primera vez al verdadero éxito internacional.

La empresa barcelonesa **Gaelco**, cuyo logo podemos observar en la Figura 2.6, es, con muchísima diferencia, el máximo exponente del desarrollo de máquinas recreativas en España. El diseñador de hardware Javier Valero, alma máter de Gaelco y líder de un pequeño equipo, era capaz de desarrollar placas recreativas de potencia equiparable a las de las grandes del arcade.

El componente que necesitaba esta empresa lo aportó la presencia de programadores de la extinta empresa Zigurat - Made in Spain, que aportaron su experiencia como desarrolladores de juegos exitosos al aliarse con Gaelco y participaron en algunos de los desarrollos más exitosos de la compañía. Aunque la propia Gaelco contaba con sus propios equipos de artistas y programadores, era



Figura 2.5: Juego arcade Donkey Kong (1981)

una época en la que los equipos eran mucho más reducidos, y fueron capaces de lanzar desarrollos que plantaban cara decididamente en el mercado internacional.



Figura 2.6: Logo de la empresa española Gaelco

Esta combinación de hardware mucho más que solvente y programadores con experiencia tuvo como resultado la llegada al mercado de una serie de títulos que son, con muchísima diferencia, lo mejor que ha dado de sí la industria española del videojuego en su vertiente para recreativas.

Algunos ejemplos de los videojuegos arcade desarrollados por esta compañía española son Big Karnak (1991), o World Rally (1993).

Big Karnak (1991)

Ambientado en el antiguo Egipto, es un título de excelente factura técnica y aguanta el tipo sin problemas ante lo que se estaba haciendo por esta época. Y

no era cualquier época. Recordemos que estamos en 1991, y que este mismo año Capcom lanzaba el primer Street Fighter 2, cambiando para siempre el mundo de las recreativas.



Figura 2.7: Juego arcade Big Karnak (1991)

World Rally (1993)

Fruto de la integración entre la Zigurat que ya había lanzado un juego de Rallies para ordenadores de 8 bits, y el poderoso hardware que proporcionaba Gaelco, llegó finalmente un triunfo internacional. Uno con mayúsculas, premio al juego más vendido de Europa incluido y distribuido en todo el mundo. De nuevo estamos ante un juego en el que se une una técnica solvente y jugabilidad para dar como resultado un producto de primera división.



Figura 2.8: Juego arcade World Rally (1993)

CAPÍTULO 3

Elementos de diseño

A continuación mostramos una introducción a los sistemas RaspberryPi y Arduino, mediante los cuales acometeremos la realización de este proyecto.

3.1 Raspberry Pi

La Raspberry Pi es una placa computadora (SBC¹) de bajo coste desarrollada desde 2012 en Reino Unido por la Raspberry Pi Foundation, con el objetivo de estimular la enseñanza de ciencias de la computación en las escuelas. De esta forma fue diseñada con el fin de ser lo más barata posible y poder llegar al máximo número de usuarios.

Con unas dimensiones de 85,60 mm x 56 mm x 21 mm, un peso de 45 g, y un precio de 35 \$ en su versión más completa, la Raspberry Pi se presenta como una revolución dentro del mundo de los dispositivos empotrados. Dentro del repertorio de posibilidades que encontramos con este dispositivo podemos destacar algunas como usarlo a modo de NAS, de servidor de aplicaciones distribuidas, o de servidor de correo. También hace las veces de computador de sobremesa, por lo que podemos realizar con él tareas tales como reproducir vídeos en alta definición, reproducir música, utilizar aplicaciones de ofimática, o navegar por la red, eso sí, debido a su limitada potencia no podemos esperar un gran rendimiento. Además, unos de los usos más extendidos y que más éxito han tenido con este dispositivo han sido su integración en sistemas robóticos a modo de controlador central, o usarlo para emular sistemas recreativos arcade.

La popularidad de Raspberry Pi con estos fines reside en la disminución de los costes de un proyecto en el que debido a su naturaleza, o a las necesidades tecnológicas del mismo, suelen acarrear grandes costes durante su desarrollo; además,

¹Un SBC (*Single Board Computer*) es un computador completo construido en una única placa de circuitos, con microprocesador, entradas y salidas de datos, y otras características propias de un computador funcional.

su reducido tamaño es un punto a su favor, ya que actualmente uno de los objetivos de la tecnología no es solo la eficiencia de los productos, sino que además, es muy valorado el tamaño de los mismos.

Finalmente cabe destacar que la Raspberry Pi no viene con un manual de fábrica, ni con unas instrucciones de instalación ni con nada parecido, por lo que para el usuario final puede aparentar la vuelta a una era en la que cada uno debía apañárselas para administrar sus dispositivos, era en la cual se aprendía investigando, tocando ajustes y probando qué pasaba, cómo funcionaba mejor, o investigando cuáles eran las mejores soluciones para hacer que nuestras máquinas hicieran lo que nosotros pretendíamos. En definitiva, la Raspberry Pi nos remonta a otra época, en la que el aprendizaje y la investigación eran necesarios para conseguir un buen rendimiento de los dispositivos que construimos o administramos.

3.1.1. Hardware

En el modelo B, que es el que vamos a utilizar en nuestro proyecto, y que se muestra en la Figura 3.1, podemos encontrar unas características muy interesantes. En su corazón nos encontramos con un chip integrado Broadcom BCM2835, que contiene un procesador ARM11 con varias frecuencias de funcionamiento y la posibilidad de subirla (*overclocking*) hasta 1 GHz sin perder la garantía, un procesador gráfico VideoCore IV, y además, las últimas Raspberry Pi cuentan con 512 MB de memoria RAM. Todo ello equivale en la práctica a un ordenador con unas capacidades gráficas similares a la XBOX de Microsoft y con la posibilidad de reproducir vídeo en 1080 p.

En la placa nos encontramos además con una salida de vídeo y audio a través de un conector HDMI, con lo que conseguiremos conectar la tarjeta tanto a televisores como a monitores que cuenten con dicha conexión. En cuanto a vídeo se refiere, también cuenta con una salida de vídeo compuesto y una salida de audio a través de un conector *minijack*. Posee una conexión Ethernet 10/100 y, si bien es cierto que podría echarse en falta una conexión Wi-Fi, gracias a los dos puertos USB incluidos podremos suplir dicha carencia con un adaptador Wi-Fi USB de otros distribuidores si lo necesitamos. Los puertos tienen una limitación de corriente, por lo que si queremos conectar discos duros u otro dispositivos debemos pensar en hacerlo a través de un hub USB con alimentación.

En cuanto al tipo de unidades de almacenamiento de este dispositivo, hay que comentar que no tiene ninguno. Pero en su parte inferior, cuenta con un lector de tarjetas SD, lo que abarata enormemente su precio y da la posibilidad de instalar un sistema operativo en una tarjeta de memoria de 2 GB o más (clase 4 o mejor). De esta forma tenemos también la posibilidad de minimizar el espacio que necesitamos para tener todo un ordenador en un volumen mínimo. Para su alimentación podemos usar un cargador con conexión micro USB que pueda dar una corriente de al menos 750 mA.

RASPBERRY PI MODEL B

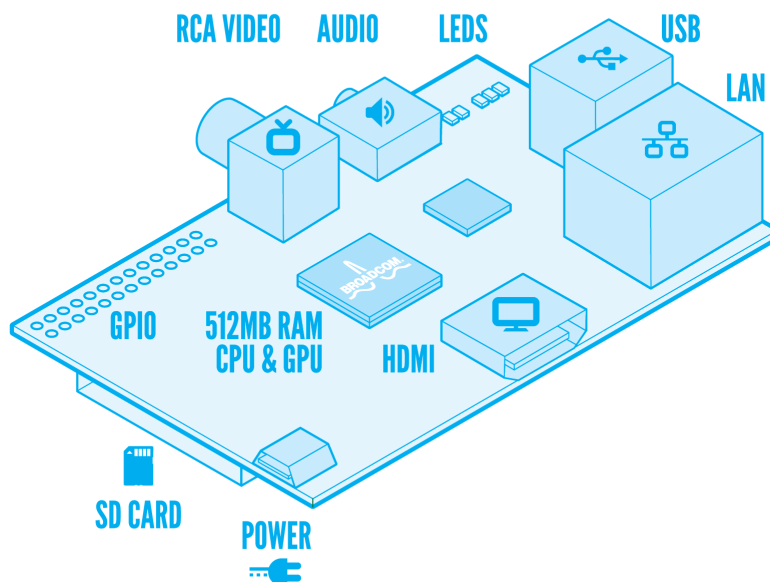


Figura 3.1: Raspberry Pi model B

3.1.2. Software

La Raspberry Pi utiliza mayoritariamente sistemas operativos basados en el núcleo Linux, aunque podemos meter con calzador cualquier sistema al que le basten los 512 MB de la memoria RAM para funcionar con fluidez. Entre las distribuciones más comunes podemos destacar las de propósito general:

- Raspbian: quizás la distribución Linux más conocida para Raspberry Pi, un sistema operativo basado en Debian que nos ofrece un entorno tanto en modo consola como en escritorio, con el que podremos programar o jugar. A través de la Pi Store, la tienda de aplicaciones de Raspberry Pi, podremos acceder a múltiples aplicaciones tanto gratuitas como de pago y, por ejemplo, podremos instalar aplicaciones de ofimática tales como LibreOffice.
- Pidora: es la versión de Fedora para procesadores ARM, y concretamente, la versión orientada a Raspberry Pi.
- Moebius: es una distribución que se caracteriza por ser extremadamente ligera, y por tener el servicio ssh activado por defecto.
- Arch Linux ARM: es otra opción simple y ligera que también podemos usar como sistema operativo base para nuestros proyectos.



Figura 3.2: Logos de Moebius, Raspbian, ArchLinux, y Pidora, respectivamente.

Por otra parte, también podemos encontrar distribuciones orientadas a usos más concretos. De la misma forma que podemos encontrar distribuciones Linux que se han diseñado para acometer funciones concretas (*firewalls*, sistemas de *backups*, etc), en el mundo de este dispositivo empotrado también podemos encontrar distribuciones optimizadas para un entorno o ámbito de aplicación concreto, que nos permiten transformar nuestra Raspberry Pi en una máquina arcade, o en un Media Center:

- ArkOS: es una distribución que ha nacido gracias al *crowdfunding*² con el objetivo de ofrecernos todas las herramientas necesarias para desplegar un servidor privado para nuestros archivos. Con ArkOS solamente necesitaremos una Raspberry Pi, una unidad de almacenamiento USB, y por supuesto, una tarjeta SD con la distribución. De esta forma, podemos disponer de un servidor privado que administramos según nuestras necesidades.
- OpenELEC (*Open Embedded Linux Entertainment Center*): es una distribución muy ligera con la que podremos implementar un Media Center sobre nuestra Raspberry Pi. Teniendo en cuenta que Raspberry Pi posee una salida HDMI, la idea es conectarla a nuestro televisor para usarla como reproductor multimedia de bajo coste con una distribución especialmente optimizada para este tipo de tareas y con un interfaz orientado también a hacernos sencillo su manejo.
- PiMAME: es una distribución que transformará nuestra Raspberry Pi en una máquina arcade; concretamente, esta distribución nos ofrece emuladores como MAME y algunos otros emuladores de consolas clásicas como Neo Geo, PlayStation, Super NES o Atari 2600. La distribución se apoya en Raspbian pero, cara al usuario, pone las cosas muy sencillas e implementa un servidor FTP para que nos sea fácil llevar las ROMs de los juegos a la tarjeta SD del computador.

²*Crowdfunding*, microfinanciación colectiva: es la cooperación colectiva llevada a cabo por diferentes personas para conseguir dinero u otros recursos.




	Model A	Model B	Model B+
			
SoC	Broadcom BCM2835		
CPU	700 Mhz ARM1175JZF-S		
GPU	VideoCore IV 250 Mhz		
RAM	256 MB	512 MB	
USB	1	2	4
Vídeo	RCA, HDMI		Jack, HDMI
Audio	Jack, HDMI		
Almacenamiento	SD		MicroSD
Red	-	Ethernet 10/100	
Consumo	300mA/1,5w/5v	700mA/3,5w/5v	600mA/3w/5v
Alimentación	MicroUSB / GIPO		
Tamaño	85,6x53,98x17 mm		85x56x17 mm
Precio	25 \$	35 \$	

Tabla 3.1: Comparativa entre los modelos A, B, y B+ de Raspberry Pi

3.1.3. Especificaciones técnicas

En la Tabla 3.1 mostramos las diferencias en cuanto a características técnicas existentes entre los tres modelos de Raspberry Pi más comunes en la actualidad.

3.2 Arduino

Arduino es una plataforma de prototipos electrónicos de código abierto (*open-source*³) basada en hardware y software flexibles y fáciles de usar. Arduino puede percibir características del entorno que le rodea mediante la recepción de señales analógicas haciendo uso de una gran variedad de sensores, y puede actuar e influir en el entorno mediante el control de luces, motores y otros artefactos.

El microcontrolador de la placa se programa usando el *Arduino Programming Language* (basado en Wiring⁴) y el *Arduino Development Environment* (basado en Processing⁵). Los proyectos de Arduino pueden ser autónomos o se pueden co-

³Código abierto (*Open-source*): es la expresión con la que se conoce al software distribuido y desarrollado libremente.

⁴Más información en <http://wiring.org.co>

⁵Más información en <http://www.processing.org>

municar con software en ejecución en un ordenador (por ejemplo con Flash, Processing, MaxMSP, etc.), las placas se pueden ensamblar a mano o encargarse preensambladas, y el software para programar el Arduino se puede descargar gratuitamente⁶. Los diseños de referencia del hardware (archivos CAD) están disponibles bajo licencia *open-source*, por lo que eres libre de adaptarlas a tus necesidades.

Hay muchos otros microcontroladores y plataformas microcontroladoras disponibles para computación física. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard, y muchas otras ofertas de funcionalidad similar. Todas estas herramientas toman los desordenados detalles de la programación del microcontrolador y la encierran en un paquete fácil de usar. Arduino también simplifica el proceso de trabajo con microcontroladores, pero ofrece algunas ventajas para profesores, estudiantes y aficionados interesados sobre los otros sistemas:

- **Barato:** Las placas Arduino son relativamente baratas comparadas con otras plataformas microcontroladoras. La versión menos cara del módulo Arduino puede ser ensamblada a mano, e incluso los módulos de Arduino preensamblados cuestan menos de 50 \$.
- **Multiplataforma:** El software de Arduino se ejecuta en sistemas operativos Windows, Macintosh OSX y GNU/Linux. La mayoría de los sistemas microcontroladores están limitados a Windows.
- **Entorno de programación simple y claro:** El entorno de programación de Arduino es fácil de usar para principiantes, pero suficientemente flexible para que usuarios avanzados puedan aprovecharlo también.
- **Código abierto y software extensible:** El software Arduino está publicado como herramienta de código abierto, disponible para extensión por programadores experimentados. El lenguaje puede ser expandido mediante APIs o librerías externas desarrolladas para extender la funcionalidad de nuestro sistema, o para utilizar placas compatibles con Arduino de expansión, comunicación, etc.
- **Código abierto y hardware extensible:** El Arduino está basado en microcontroladores ATMEGA8 y ATMEGA168 de Atmel. Los planos para los módulos están publicados bajo licencia Creative Commons, por lo que diseñadores experimentados de circuitos pueden hacer su propia versión del módulo, extendiéndolo y mejorándolo. Incluso usuarios relativamente inexpertos pueden construir la versión de la placa del módulo para entender como funciona y ahorrar dinero.

⁶Más información en <http://www.arduino.cc/en/Main/Software>

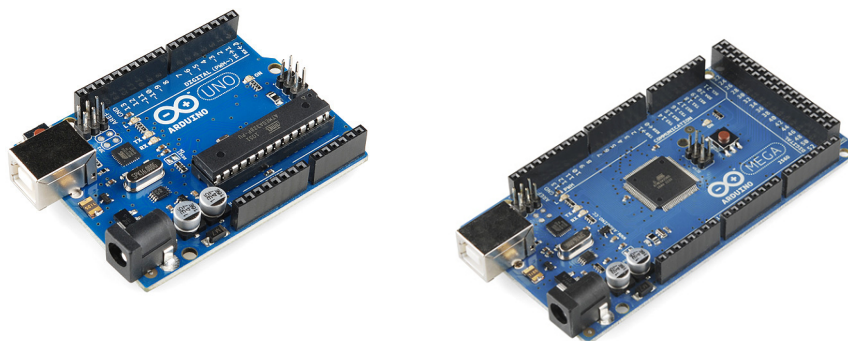


Figura 3.3: Arduino Uno Rev.3 y Arduino Mega 2560 Rev.3

3.2.1. Hardware

Hay múltiples versiones de la placa Arduino. Entre la gran variedad de versiones de estas placas microcontroladoras podemos encontrar algunas como la Arduino Uno o Mega (mostradas en la Figura 3.3), Leonardo, Due, Yún, Micro, Esplora, o Nano, entre otras⁷. En concreto, como la placa Arduino que vamos a utilizar en nuestro proyecto es la Arduino Mega 2560 Rev.3, vamos a analizar sus características técnicas.

El Arduino Mega 2560 es una placa microcontroladora basada en el ATmega2560. Tal y como observamos en la Figura 3.4, dispone de 54 pines digitales de entrada/salida (de los cuales 15 pueden ser usados como salidas analógicas), 16 entradas analógicas, 4 puertos UART de comunicación serie, un puerto USB, un conector de alimentación externa, un cabezal ICSP, y un botón de reset. Contiene todo lo necesario para mantener al microcontrolador, únicamente hay que proporcionarle una alimentación a través del USB, de una batería, o de los pines de alimentación (*Vin* y *GND*) para hacerlo funcionar. Además, destacamos que el ATmega2560 dispone de 256 KB de memoria flash para guardar código (de los cuales 8 KB se usan para el bootloader), 8 KB de SRAM, y 4 KB de EEPROM, que pueden ser leídos y escritos mediante la librería EEPROM.

⁷Para más información acerca de la variedad de placas Arduino visítase <http://arduino.cc/en/Main/Products>

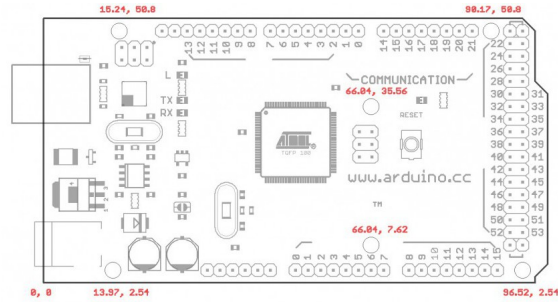


Figura 3.4: Esquema del Arduino Mega 2560 Rev.3

Arduino Mega 2560 Rev.3	
Microcontrolador	ATmega2560
Voltaje Operativo	5 V
Voltaje de entrada (recomendado)	7-12 V
Voltaje de entrada (límites)	6-20 V
Pines E/S digitales	54 (15 de salida PWM)
Pines de entrada analógica	16
Corriente DC por Pin E/S	40 mA
Corriente DC por Pin 3,3 V	50 mA
Memoria Flash	256 KB (8 KB para el bootloader)
SRAM	8 KB
EEPROM	4 KB
Velocidad de reloj	16 MHz

Tabla 3.2: Características técnicas del Arduino Mega 2560 Rev.3

CAPÍTULO 4

Diseño de la máquina arcade

En este capítulo vamos a hablar acerca de los elementos que van a formar parte de nuestra máquina arcade, entre los cuales distinguimos una Raspberry Pi, que realizará la función de computador central, un Arduino para controlar el sistema de autenticación, los controles de la máquina arcade, la gestión remota del computador central, o el diseño de los diferentes elementos software que interactuarán entre ellos para conseguir una experiencia óptima de juego.

4.1 Presupuesto

En esta sección realizamos un presupuesto detallado del conjunto de componentes que necesitaremos para llevar a cabo la construcción de nuestra máquina arcade.

Como podemos observar en la Figura 4.1, precisaremos de componentes tales como un Arduino Mega 2560 Rev.3, una *Communication Shield* y un Módulo RFID-125 Khz compatibles con Arduino, un combo de *tags* RFID, una Raspberry Pi, diferentes controles arcade, una tarjeta MicroSD de más de 4 GB de capacidad, o un adaptador WiFi USB.

Cabe destacar que el coste estimado de los muebles para máquinas arcade existentes en el mercado actualmente, oscilan entre la cifra de los trescientos y los cuatrocientos euros, lo que inflaría de forma considerable nuestro presupuesto; consecuentemente, con el fin de obtener una solución económica, al final del proyecto nos planteamos la construcción de nuestro propio mueble arcade, el cual, construido con materiales tales como la madera o el metacrilato, y sin contar la mano de obra de una empresa experta especialista, reducen de forma considerable nuestro presupuesto. De tal forma, en caso de ser llevada a cabo la construcción del mueble para nuestra máquina arcade, aumentaría el importe del presupuesto de los materiales de la Figura 4.1 en una cantidad aproximada de cien euros.

Además, en caso de querer comercializar este dispositivo, haciendo una estimación de las horas de trabajo realizadas, las cuales ascienden alrededor de quinientas horas, podríamos establecer un incremento del presupuesto obtenido en relación al precio por hora de la mano de obra empleada.

Finalmente, cabe destacar que los costes asociados al material necesario para realizar este proyecto, han sido asumidos por parte del Museo de Informática de la Universitat Politècnica de València.

Artículo	Precio Unitario	Unidades	Subtotal
Arduino Mega 2560 Rev.3	41,00 €	1	41,00 €
Communication Shield	15,00 €	1	15,00 €
RFID 125Khz Module	35,00 €	1	35,00 €
RFID Tag Combo	3,00 €	1	3,00 €
Raspberry Pi Model B	34,00 €	1	34,00 €
Arcade Joystick	15,00 €	1	15,00 €
Black Arcade Button	1,85 €	4	7,40 €
Jumper Wires M/F Pack of 100	20,00 €	1	20,00 €
MicroSD Memory Card 16GB	12,00 €	1	12,00 €
Adaptador USB Wireless	15,00 €	1	15,00 €
Total sin IVA			182,40 €
IVA (21%)			38,30 €
TOTAL			220,70 €

Figura 4.1: Presupuesto de los materiales necesarios

4.2 Computador central

Tal como hemos comentado anteriormente, vamos a hacer uso de una Raspberry Pi modelo B para que haga la función de computador central de nuestro sistema arcade. Entre las funciones asociadas a dicho computador central encontramos el almacenamiento de la base de datos que contendrá la información de los usuarios que estén autorizados a acceder al sistema, así como la interacción con el sistema de autenticación con el fin de realizar dicha autenticación. Además, el computador central almacenará el software emulador de los videojuegos arcade, y realizará la función de intermediario entre el emulador y los controles físicos que se añadirán al sistema para proporcionar una experiencia de juego más

entretenida y que recuerde al usuario la experiencia de jugar con una máquina arcade.

Otra de las funciones del computador central será almacenar y ofrecer un sistema de administración telemático, mediante el cual, los usuarios que sean administradores del sistema, podrán acceder al mismo de forma remota, con el fin de cambiar los videojuegos que ofrecerá el sistema para ser emulados, así como para realizar cambios en la base de datos de usuarios, lo que les permitirá añadir o quitar usuarios registrados para que estos tengan, o dejen de tener, los permisos necesarios para interactuar con el sistema. Dicha funcionalidad se explica con detenimiento en la sección 4.6.

4.2.1. Sistema operativo

Todas las funcionalidades descritas previamente precisan de la gestión y control de un SO¹, y como se ha indicado en secciones anteriores, la Raspberry Pi nos ofrece la posibilidad de integrar y trabajar con cualquier SO que requiera menos memoria de la integrada en la placa de la Raspberry Pi para poder funcionar. En este proyecto, hemos decidido trabajar una distribución de un SO llamada Raspbian.

Raspbian:

Raspbian, cuyo logo podemos observar en la Figura 4.2, es el nombre dado a una variante modificada de la popular distribución de Linux Debian. Debian es una de las más longevas distribuciones de Linux, y concentra una gran compatibilidad y un rendimiento excelente incluso con un hardware modesto, lo que lo convierte en una gran opción para la Raspberry Pi. Raspbian toma como base (o *distribución madre*) el SO Debian, y añade herramientas y software personalizado para conseguir trabajar con la Raspberry Pi de la forma más sencilla posible.

Para conseguir un consumo mínimo de recursos, la imagen de Raspbian para la Raspberry Pi (descargable desde la página web oficial de Raspbian²) incluye únicamente una pequeña parte del software que encontraríamos en una versión regular de un SO para escritorio. Se incluyen herramientas para navegar por Internet, programar en Python, o una interfaz gráfica para trabajar de forma más amena. Adicionalmente, se ofrece la posibilidad de instalar más software de forma rápida mediante el gestor de paquetes de la distribución, bien a través de un entorno de consola (comando *apt*), o a través del programa de interfaz gráfica *Raspberry Pi Store*.

¹Un sistema operativo (SO o, frecuentemente, OS —del inglés *Operating System*—) es un programa o conjunto de programas de un sistema informático que gestiona los recursos de hardware y provee servicios a los programas de aplicación, ejecutándose en modo privilegiado respecto de los restantes.

²Raspbian - descargas: <http://www.raspbian.org/RaspbianImages>

Raspbian incluye un entorno de escritorio conocido como el *Lightweight X11 Desktop Environment* (LXDE)³, el cual está diseñado para ofrecer una interfaz atractiva mediante un sistema de ventanas e interfaces similar al de otros sistemas operativos tales como Microsoft Windows u OSX.

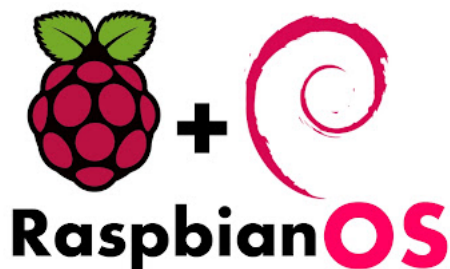


Figura 4.2: Logo de Raspbian

4.2.2. Mecanismo de alimentación

Con el fin de dotar a nuestra máquina arcade de un sistema de alimentación que nos permita encender y apagar el sistema mediante el uso de un botón de encendido/apagado, vamos a diseñar un mecanismo de alimentación que nos permita alimentar con corriente eléctrica al computador central mediante la acción de encender o apagar un interruptor; de tal forma que cuando dicho interruptor esté apagado, el computador central no recibirá energía mediante la cual funcionar él mismo o el conjunto de periféricos que controlará, mientras que cuando dicho interruptor esté activado, el computador central recibirá la corriente eléctrica necesaria para iniciar la funcionalidad del sistema.

Para montar tal mecanismo de alimentación, necesitaremos un interruptor del tipo *toggle switch*, y un cargador eléctrico del tipo *mini USB*, el cual es el que utiliza la Raspberry Pi para alimentarse de la corriente eléctrica alterna.

Una vez disponemos de los elementos necesarios, únicamente debemos cortar el cable del cargador *mini USB* y recablearlo tal y como muestra la Figura 4.3, de tal forma que al cortar el cable del cargador, nos encontramos con dos cables interiores, uno por el que pasa la corriente (*V-in*), y otro por el que se da acceso a *tomatierra* o *masa* (*GND*). En el recableado del cargador, debemos asegurarnos que los dos cables de *tomatierra* se conecten al mismo pin del *toggle switch* (el cual es mostrado en la figura), mientras que los otros dos cables de alimentación (*V-in* y *V-out*), deben ser conectados a los pines representados con el fin de que sea el propio interruptor *toggle switch* el que permita el paso de la corriente en el momento en que este sea activado.

³Más información en la página web de LXDE: <http://lxde.org/es>

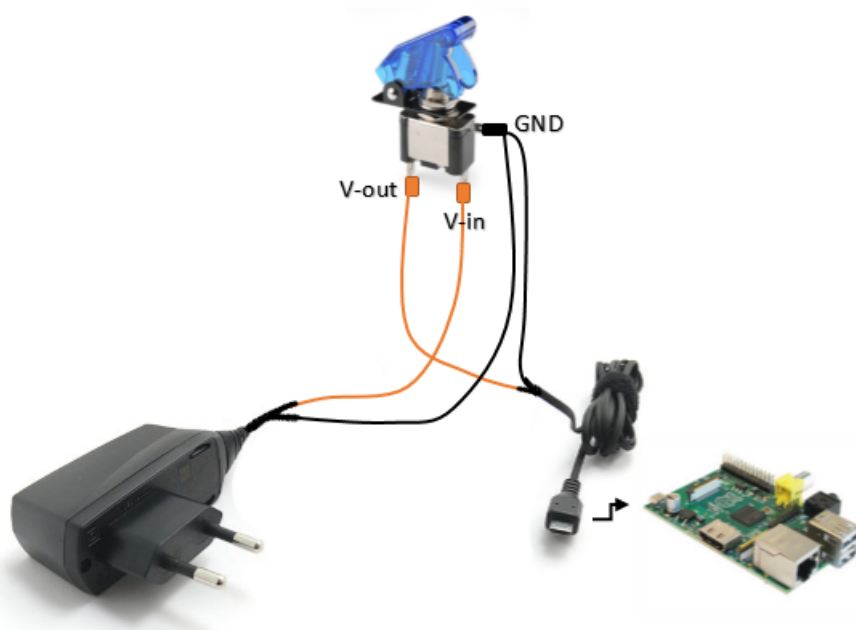


Figura 4.3: Mecanismo de alimentación de la Raspberry Pi

4.3 Emulador arcade

Un emulador es un *software* que permite ejecutar programas o, en este caso, videojuegos, en una plataforma (sea una arquitectura de hardware o un sistema operativo) diferente de aquella para la cual fueron escritos originalmente. A diferencia de un simulador, que solo trata de reproducir el comportamiento del programa, un emulador trata de modelar de forma precisa el dispositivo de manera que este funcione como si estuviese siendo usado en el aparato original.

Un uso popular de los emuladores es el de imitar la experiencia de los videojuegos de máquinas recreativas o videoconsolas en computadoras personales, o el poder ser jugados en otras videoconsolas. La emulación de videojuegos de sistemas antiguos en las modernas computadoras personales y videoconsolas de hoy día resulta generalmente más cómoda y práctica que en los dispositivos originales.

Los emuladores arrancan imágenes *ROM*, o sea, el contenido de los cartuchos, disquetes o cintas que se usaban con los sistemas antiguos. Físicamente en las PC las *ROMs* son archivos binarios que se pueden cargar en la memoria. Es decir, el emulador es un programa que hace las funciones de una consola, por ejemplo la *Game Boy Advance* o una *PDA*, y la *ROM* es un archivo que hace de cartucho, CD-ROM, o cinta, por ejemplo *Mario Bros*.

4.3.1. MAME

Para nuestro proyecto vamos a utilizar el emulador **MAME4ALL**, el cual es un emulador adaptado para la Raspberry Pi, basado en el emulador MAME 0.37b5. El *Multiple Arcade Machine Emulator* (emulador de múltiples máquinas recreativas), más conocido por sus siglas MAME⁴, cuyo logo observamos en la Figura 4.4, es un emulador de máquinas recreativas de los más completos que se pueden encontrar. Para hacer funcionar un juego, se requiere su correspondiente ROM (archivo con una imagen de la ROM de la máquina, que contiene el juego en sí). Mame es un programa de código abierto y gratuito si se utiliza sin ánimo de lucro.



Figura 4.4: Logo del emulador MAME

Con el fin de instalar dicho emulador en nuestra Raspberry Pi, nos dirigimos a la página web oficial del emulador MAME4ALL⁵, desde donde nos descargamos los ficheros necesarios para la instalación del emulador. Una vez descargados y descomprimidos los ficheros de instalación, únicamente debemos ejecutar el *script* `install.sh` que está presente entre los ficheros descargados.

Adicionalmente, para una posterior facilidad a la hora de invocar al emulador desde cualquier consola, añadimos al *PATH* del sistema la localización de nuestro emulador, por ejemplo, mediante la siguiente orden de consola: `PATH = //installationDirectory: $PATH ; export PATH`, donde la variable *installationDirectory* equivaldría al directorio en el que hemos instalado nuestro emulador.

Una vez realizadas dichas acciones, podríamos ejecutar el emulador a través de cualquier terminal de consola mediante la orden `./mame`; mientras que para añadir juegos al emulador únicamente debemos copiar las ROMs de los videojuegos al directorio llamado ROMS situado dentro del directorio de instalación del emulador.

4.3.2. Configuraciones del emulador

Para asegurar el correcto funcionamiento del emulador, de forma que se pueda jugar a los videojuegos de una forma fluida, vamos a realizar algunos ajustes en la configuración interna de la Raspberry Pi, tales como aumentar la frecuencia

⁴Más información en la página web oficial de MAME: <http://www.mame.net/>

⁵<http://code.google.com/p/mame4all-pi/>

de reloj de su *CPU*, ajustar la configuración del audio del emulador para lograr una calidad de sonido mejor, o modificar la interfaz gráfica del emulador.

Aumentar la frecuencia de reloj:

Es altamente recomendable aumentar la frecuencia de reloj de nuestra Raspberry Pi con el fin de obtener un máximo rendimiento, ya que MAME realiza un uso intensivo de la *CPU*, de tal forma que mejoraremos sustancialmente la fluidez de algunos videojuegos. Para ello, debemos modificar el fichero `config.txt` situado en la ruta `/boot/` de nuestro sistema de archivos, añadiendo los valores contenidos en las líneas indicadas en la Tabla 4.1, las cuales deben ser añadidas al final del documento. Además, es también recomendable asegurar un mínimo de 64 MB de *RAM GPU*⁶, lo cual es conseguido añadiendo, en el mismo fichero, a continuación de las líneas ya añadidas previamente, una nueva línea que modifique el valor de memoria asignado a dicho coprocesador (`gpu_mem = 64`).

```
arm_freq=900
core_freq=300
sdram_freq=500
```

Tabla 4.1: Líneas a añadir en el fichero de configuración `/boot/config.txt`

Aumentar la calidad del audio:

Para mejorar la calidad del audio, inicialmente debemos comprobar cuál es la salida de audio del emulador, esto lo averiguamos mediante la orden de consola `amixer controls`. Dicha orden nos devolverá un número (típicamente la interfaz que nos interesa es la etiquetada con el número 3). Una vez conocida la interfaz que nos interesa, únicamente debemos ejecutar desde una consola la orden `amixer cset numid=VALOR 90%`, donde el ítem *VALOR* equivale al valor de etiqueta de la interfaz de audio por la que se emiten los sonidos del emulador, obtenido mediante la orden de consola anterior. Dicha orden de consola aumentará el volumen de la salida de audio del emulador.

Editar la interfaz del emulador:

Para editar la interfaz gráfica del emulador debemos acceder al directorio `skins` situado dentro del directorio de instalación del emulador. Una vez dentro podemos encontrar algunos ficheros gráficos, entre los cuales distinguimos

⁶Unidad de procesamiento gráfico o GPU (*Graphics Processing Unit*) es un coprocesador dedicado al procesamiento de gráficos u operaciones de coma flotante, para aligerar la carga de trabajo del procesador central en aplicaciones como los videojuegos o aplicaciones 3D interactivas.

los ficheros *rpisplah16.bmp*, el cual es la pantalla de inicio del emulador, y *rpimenu16.bmp*, el cual es la interfaz de selección de juegos del emulador.

Para personalizar dichas interfaces, únicamente debemos editar los ficheros descritos, teniendo en cuenta que el resultado final debe ser un fichero de imagen posterizado a 16 bits de colores. Podemos ver las interfaces originales en la Figura 4.5, mientras que las interfaces modificadas que van a estar presentes en nuestro sistema emulador son las mostradas en la Figura 4.6.



Figura 4.5: Interfaces originales del emulador MAME

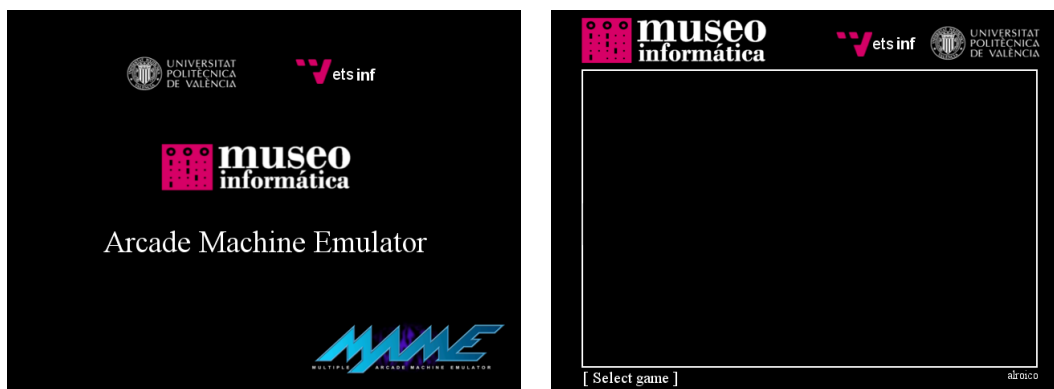


Figura 4.6: Interfaces editadas del emulador

4.4 Interconexión de controles

Uno de los elementos más representativos de las máquinas arcade son sus controles, es decir, las palancas (*joysticks*) y los botones pulsadores, los cuales son utilizados para interactuar con el sistema y poder jugar a los videojuegos con mayor facilidad permitiendo, además, simplificar las acciones requeridas para jugar, de tal forma que hace más entretenida y más amigable la experiencia de juego.

Nuestra máquina arcade se va a diseñar para jugar a videojuegos de un solo jugador, por lo que los elementos que vamos a necesitar para diseñar nuestro panel de controles van a ser un *joystick* de cuatro ejes y pulsador integrado (Figura 4.7), dos pulsadores de colores para realizar acciones dentro del juego, y tres pulsadores negros (Figura 4.8), los cuales controlarán algunas acciones del emulador tales como insertar una moneda, empezar el juego, o volver hacia atrás en el menú del juego.



Figura 4.7: Joystick arcade



Figura 4.8: Botones pulsadores arcade

Dichos componentes están formados todos ellos por un *microswitch* (micro-interruptor). Un *microswitch*, el cual podemos observar en la Figura 4.9, es un dispositivo cuya función es conmutar entre uno o varios circuitos. Son muy utilizados en los controles arcade, tanto en palancas como en botones. Tiene forma rectangular generalmente y en su exterior podemos visualizar tres terminales o contactos metálicos: dos patillas que salen de uno de sus lados más cortos y otro terminal saliendo en escuadra de uno de sus lados largos.

Los terminales del microswitch, están etiquetados como *NC*, *NO*, y *COM* por convención. Por lo tanto, el modo de cableado estándar es elegir como conexión a tierra (o masa) el terminal marcado como *COM*, es decir, el terminal que sale de uno de sus lados largos. Las otras dos patillas restantes son las que conmutan aunque de dos maneras diferentes: una es responsable de mantener el circuito abierto o desconectado hasta que el mecanismo de conmutación sea activado (lo que se denomina normalmente abierto), y la otra mantiene el circuito cerrado o conectado hasta que se activa el mecanismo conmutador (normalmente cerrado).

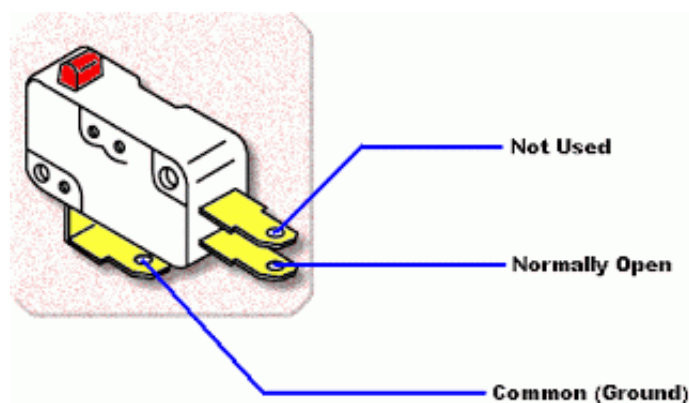


Figura 4.9: Pulsador *microswitch*

La patilla *NC* indica un contacto cerrado (no conectado) permanentemente cuando el microswitch está en estado de reposo. El circuito solo se interrumpirá cuando el pulsador del interruptor sea activado, en este caso, para nuestro control arcade, no nos interesa para nada. La patilla *NO*, como lo indica el esquema que se puede ver en el microswitch, nos indica que el circuito está abierto en estado de reposo, y solo se conectará cuando pulsemos el interruptor. Este es el terminal que nos interesa conectar.

Como hemos visto en apartados anteriores, la Raspberry Pi dispone de una serie de puertos físicos (pines *GPIO*, mostrados en la Figura 4.10) a los que podemos conectar cualquier tipo de dispositivo electrónico, y es en dichos pines, donde vamos a conectar los elementos integrantes de nuestro panel de controles arcade.

Así que para conectar los interruptores de los botones y palancas arcade a los puertos *GPIO* de la Raspberry Pi, tenemos que conectar el terminal de fase, es decir, la patilla *NO*, al terminal de fase o de señal del botón o dirección que corresponda, y también conectar el terminal de tierra o de masa (patilla *COM*), a la tierra del circuito. Así sucesivamente con todas las señales del circuito. De esta forma, podemos observar en la Figura 4.11 cuál sería el resultado de cablear todos y cada uno de los elementos integrantes de nuestro panel de controles.

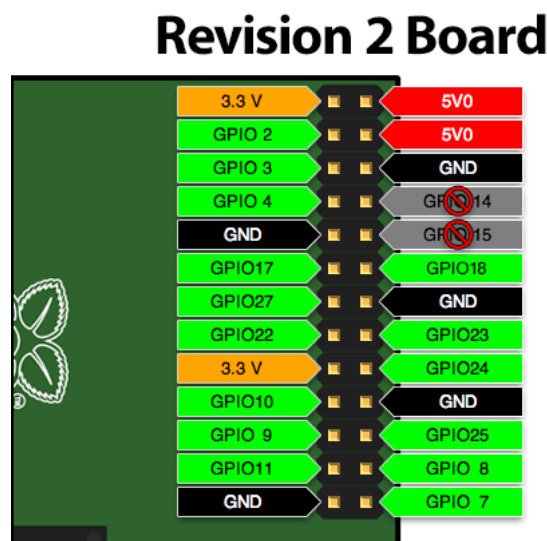


Figura 4.10: Puertos *GPIO* de la Raspberry Pi Modelo B

4.4.1. Emulación de controles

Una vez hemos conectado correctamente todas las conexiones de nuestros controles arcade con los pines *GPIO* de la Raspberry Pi, vamos a hacer uso de un software emulador de controles, el cual nos permitirá realizar la interacción del emulador con los controles, permitiendo que las señales emitidas por los interruptores sean reconocidas por el emulador como acciones dentro de los videojuegos emulados o dentro del propio emulador, acciones tales como moverse por los menús y por las interfaces de los videojuegos, saltar, disparar, etc.

El software que vamos a utilizar es un programa conocido como *Retrogame*, escrito en el lenguaje de programación *C*, cuyo desarrollador es el equipo de trabajo de la marca *Adafruit*. Los códigos necesarios para realizar la interacción entre el emulador y los controles son de código abierto, es decir, podemos descargarlos y modificarlos en base a nuestras necesidades⁷. Siguiendo las indicaciones incluidas en dichos códigos, así como las encontrados en las instrucciones de uso del propio desarrollador, podemos alterar los puntos claves del código de forma que mediante una simple modificación somos capaces de indicar qué acciones dentro del emulador estarán relacionadas con qué señales de los botones⁸.

La sección del código principal a modificar la podemos observar en la Tabla 4.2, la cual forma parte del fichero `retrogame.c` incluido en los códigos descargados, y en la que se relacionan cada una de las acciones dentro del emulador con uno de los pines *GPIO* de la Raspberry Pi. Podemos ver, por ejemplo, cómo el pin de la Raspberry Pi número 25 está asociado a la acción de moverse hacia

⁷Descargables desde <https://github.com/adafruit/Adafruit-Retrogame>

⁸Instrucciones de configuración accesibles en <https://learn.adafruit.com/retro-gaming-with-raspberry-pi/buttons#one-last-piece-dot-dot-dot>

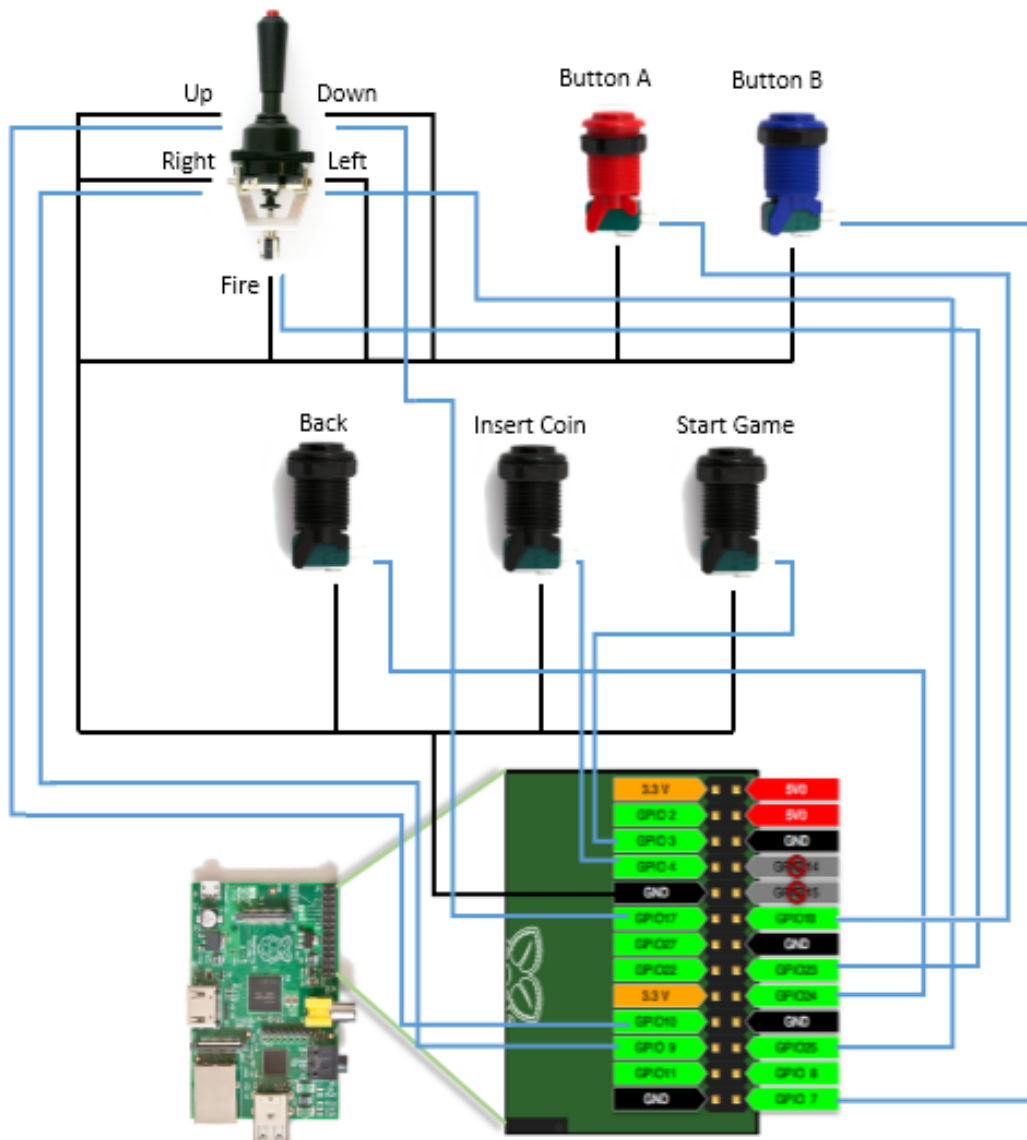


Figura 4.11: Cableado de los controles arcade

la izquierda, o cómo la acción de disparar es emulada por la tecla *leftctrl*, la cual está asociada a la señal emitida por el pin *GPIO* número 23.

```
ioStandard[] = {
// This pin/key table is used when the PiTFT isn't found
// (using HDMI or composite instead), as with our original
// retro gaming guide.
// Input  Output (from /usr/include/linux/input.h)
{ 25,    KEY_LEFT    }, // Joystick (4 pins)
{  9,    KEY_RIGHT   },
{ 10,    KEY_UP      },
{ 17,    KEY_DOWN    },
{ 23,    KEY_LEFTCTRL}, // A/Fire/jump/primary
{ 18,    KEY_LEFTCTRL}, // A/Fire/jump/primary
{  7,    KEY_LEFTALT }, // B/Bomb/secondary
{ 24,    KEY_ESC     }, //Back Key
{  4,    KEY_5       }, //Insert Coin
{  3,    KEY_1       }, //Start game
// For credit/start/etc., use USB keyboard or add more buttons.
{ -1,    -1          } }; // END OF LIST, DO NOT CHANGE
```

Tabla 4.2: Sección del código `retrogame.c`

4.5 Identificación de usuarios

Para realizar la autenticación de los usuarios en el sistema vamos a utilizar la tecnología de identificación mediante radiofrecuencia (RFID), integrándola en un sistema Arduino. De esta forma, los usuarios dispondrán de tarjetas y *tags* de radiofrecuencia mediante las cuales deberán identificarse a la hora de querer interactuar con el sistema. Consiguientemente, solo los usuarios cuyo identificador haya sido validado previamente en el sistema estarán autorizados a interactuar con el dispositivo, siendo rechazada dicha interacción en caso de una autenticación errónea o no validada.

4.5.1. Tecnología RFID

La identificación por radiofrecuencia (RFID, *Radio Frequency IDentification*), es un sistema de almacenamiento y recuperación de datos remoto que usa dispositivos denominados etiquetas, tarjetas, o *tags* RFID. El propósito fundamental de la tecnología RFID es transmitir la identidad de un objeto (similar a un número de serie único) mediante ondas de radio. Las tecnologías RFID se agrupan dentro de las denominadas Auto ID (*Automatic IDentification*, o identificación automática).

Tal y como afirman Paris Kitsos y Yan Zhang en el libro 'RFID Security'⁹, la identificación por radiofrecuencia es una de las formas más seguras de identificación electrónica.

Tal y como podemos observar en la Figura 4.12, los sistemas de identificación por radiofrecuencia básicos están formados por un lector y por un transpondedor. El lector es la unidad que actúa de base y proporciona al transpondedor la energía y las señales de comunicación necesarias para forzar al mismo a que ejecute la acción solicitada, que en este caso sería la emisión de su identificador. El control del lector puede realizarse tanto mediante la terminal de un computador, como mediante ejecuciones automáticas o *scripts*. En las instalaciones estáticas, los lectores son fijos y están conectados a una red eléctrica, mediante la que se alimenta al sistema, y a una red de comunicación, mientras que los identificadores son elementos móviles que no están conectados a fuentes de energía.

El transpondedor o *tag* es el dispositivo de identificación que se sitúa en el objeto que queremos identificar. La mayoría de *tags* de identificación no poseen una fuente de energía interna o propia, por lo que se les denomina transpondedores pasivos. La fuente de alimentación de los *tags* es proporcionada por los lectores, de tal forma que el *tag* genera su propio voltaje de alimentación mediante el voltaje inducido por las señales de radiofrecuencia que emite el lector. Además, también existen los transpondedores activos, los cuales funcionan igual que los transpondedores pasivos, pero disponen de una fuente de alimentación propia, mediante la cual alimentan un circuito que les permite enviar información al lector.

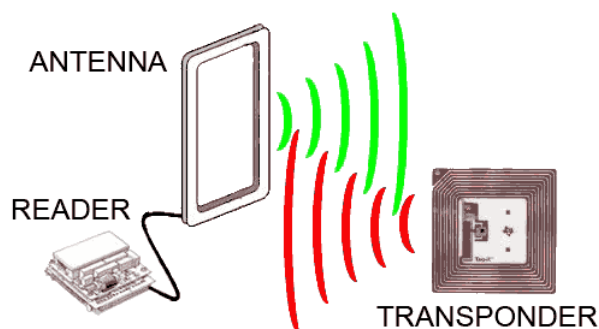


Figura 4.12: Lector y transpondedor RFID

4.5.2. RFID ISO Estándar

Desde los inicios de la tecnología RFID hasta prácticamente unos pocos años, la mayoría de las compañías relacionadas con estos sistemas desarrollaban sus propios diseños, tratando de mejorar el comportamiento de esta tecnología con el

⁹RFID Security: *Techniques, Protocols and System-On-Chip Design*. Springer Science, 2008, p.3

fin de cubrir necesidades específicas de sus usuarios o de sus dispositivos. Todas esas soluciones estaban basadas en conceptos similares, pero pequeñas diferencias, especialmente en la frecuencia bajo la que operaban los diferentes dispositivos y en la codificación de los datos, condujeron a problemas de interoperabilidad. Los usuarios y los fabricantes de la tecnología RFID reconocieron que esa situación no podía ayudar a entrar en una época en la que el gran volumen de aplicaciones precisaban de uniformidad, interoperabilidad y fuentes comunes como requisitos principales. La mejor forma de acabar con esa situación fue definir un estándar, lo que originó el establecimiento por parte de la Organización Internacional para la Estandarización (ISO, *International Organization for Standardization*) del estándar ISO-14443¹⁰.

Los estándares RFID más importantes que definen comunicaciones cubren las siguientes aplicaciones:

- Identificación de animales
- Gestión de objetos
- Logística
- Control de acceso
- Tarjetas inteligentes
- Pagos telemáticos
- Pasaportes electrónicos
- Gestión de residuos

Para permitir el uso de la comunicación por radiofrecuencia sin interferencias entre otros servicios, el espectro de frecuencia fue dividido en bandas de frecuencia y asignadas a determinados servicios, los cuales estarían autorizados a operar en dicho espectro de frecuencia. A estas frecuencias se les denominan servicios primarios, como la telefonía móvil, astronomía, o emisiones. Los espectros de frecuencia son gestionados por diferentes organizaciones tanto nacionales como internacionales, tales como ITU¹¹, ETSI¹², o CEPT¹³. Además de estas bandas de

¹⁰Para una mejor comprensión del estándar y de la forma de funcionamiento de este sistema, se puede consultar el artículo '*Requirements of ISO/IEC 14443 Type B Proximity Contactless Identification Cards*' de ATMEL en la dirección web <http://www.atmel.com/Images/doc2056.pdf>

¹¹La ITU (*International Telecommunication Union*) es el organismo especializado de las Naciones Unidas para las tecnologías de la información y la comunicación – TIC. <http://www.itu.int/>

¹²ETSI - *European Telecommunications Standards Institute*, www.etsi.org

¹³CEPT - *European Conference of Postal and Telecommunications Administrations*. <http://www.cept.org/>

servicios primarios, existen bandas de frecuencia definidas que pueden ser utilizadas por otros servicios, las cuales son las bandas industrial, científica, y médica (ISM). La tecnología RFID no está asignada como usuario primario de una banda de frecuencia concreta, está únicamente autorizada a operar bajo la banda de frecuencia ISM, la cual le otorga un rango de frecuencias sobre las que operar de 125 / 134 kHz, 13,56 MHz, 2,45 GHz, y 5,8 GHz. No hay una frecuencia de operación ideal que cubra las necesidades de toda aplicación, ya que cada frecuencia tiene sus pros y sus contras, que vienen dados por las propias propiedades de la frecuencia y por las restricciones de regulación (nivel y ancho de banda).

Algunos de los estándares asociados a las diferentes bandas de frecuencia son:

Estándar de Baja Frecuencia 134/125 kHz:

- ISO 11784 - Estructura de código de identificación de animales.
- ISO 11785 - Concepto técnico de identificación de animales.
- ISO 14223 - Transpondedores avanzados para animales.
- ISO 18000-2 - Gestión de objetos; Interfaz aérea y estructura del protocolo para baja frecuencia.

Estándar de Alta Frecuencia 13.56 MHz:

- ISO/IEC 14443 - Tarjetas de proximidad.
- ISO 18000-3 - Gestión de objetos; Interfaz aérea y estructura del protocolo para alta frecuencia.

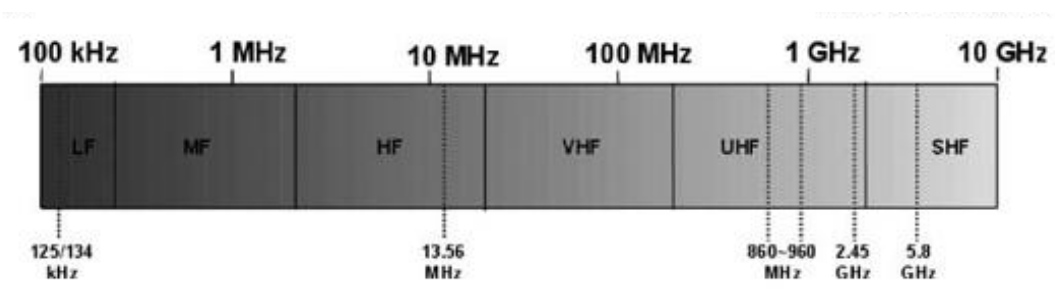


Figura 4.13: Espectro de frecuencias

4.5.3. Montaje del sistema mediante Arduino

Para establecer un sistema de identificación mediante radiofrecuencia con componentes de Arduino precisamos de elementos como una placa Arduino (sirve cualquiera de sus versiones, pero nosotros utilizaremos una placa Arduino Mega 2560 Rev.3), un lector de radiofrecuencia, un módulo de comunicación que nos permita establecer la conexión entre el Arduino y el lector, y una antena para conectar al lector de radiofrecuencia. Además, nuestro prototipo dispondrá de un zumbador (piezo, o *buzzer*¹⁴), de dos resistencias de 220 Ω , y de dos diodos LED, uno de cada color, con el fin de que cuando un *tag* RFID sea reconocido por nuestro lector, se produzca un sonido de alerta y se encienda un *led* u otro, dependiendo de si la verificación del usuario ha sido correcta o errónea.

Como hemos dicho anteriormente, la placa Arduino que vamos a utilizar es la Arduino Mega 2560 Rev.3 debido a que nos va a facilitar la conexión y comunicación del lector RFID mediante el uso de una placa de comunicación (*Communication Shield*). Además, otro motivo por el cual usamos la placa Mega 2560, es que dispone de varios puertos serie mediante los cuales establecer comunicación con otros dispositivos. El lector RFID elegido es un módulo de radiofrecuencia de 125 kHz, lo cual significa que vamos a trabajar en una banda de baja frecuencia. Hemos elegido trabajar en dicha frecuencia debido a que para este dispositivo ofrece algunas ventajas tales como que el coste económico del lector y de los transpondedores es inferior respecto a dispositivos de bandas de frecuencia superiores, la buena capacidad de emisión de la señal a través de materiales como plástico o madera, o la ventaja de que al ser una banda de frecuencia menos usada por dispositivos comunes, es menos probable obtener problemas o interferencias en la comunicación de nuestros dispositivos.

Para completar el uso operativo de nuestro sistema de identificación por radiofrecuencia debemos seguir una serie de pasos, los cuales son la identificación de los transpondedores, la instalación de una base de datos en la Raspberry Pi con el fin de establecer un conjunto de identificadores autorizados a utilizar la máquina arcade, y la comunicación de nuestro sistema Arduino con la Raspberry Pi.

4.5.4. Identificación de transpondedores

Para proceder a la identificación de los transpondedores que vamos a autorizar para usar nuestro sistema arcade, debemos realizar el montaje de los componentes Arduino, mediante los que leeremos los identificadores de nuestros *tags*. Como hemos dicho anteriormente, cada *tag* o transpondedor, dispone de un nú-

¹⁴Zumbador, *buzzer* en inglés, es un transductor electroacústico que produce un sonido o zumbido continuo o intermitente de un mismo tono. Sirve como mecanismo de señalización o aviso, y son utilizados en múltiples sistemas como en automóviles o en electrodomésticos, incluidos los despertadores.

mero de identificación propio y único que le diferencia de otros *tags* o transpondedores. Estos identificadores son los que queremos obtener para establecer un listado de identificadores admitidos, validados, y por lo tanto autorizados, a utilizar nuestro sistema.

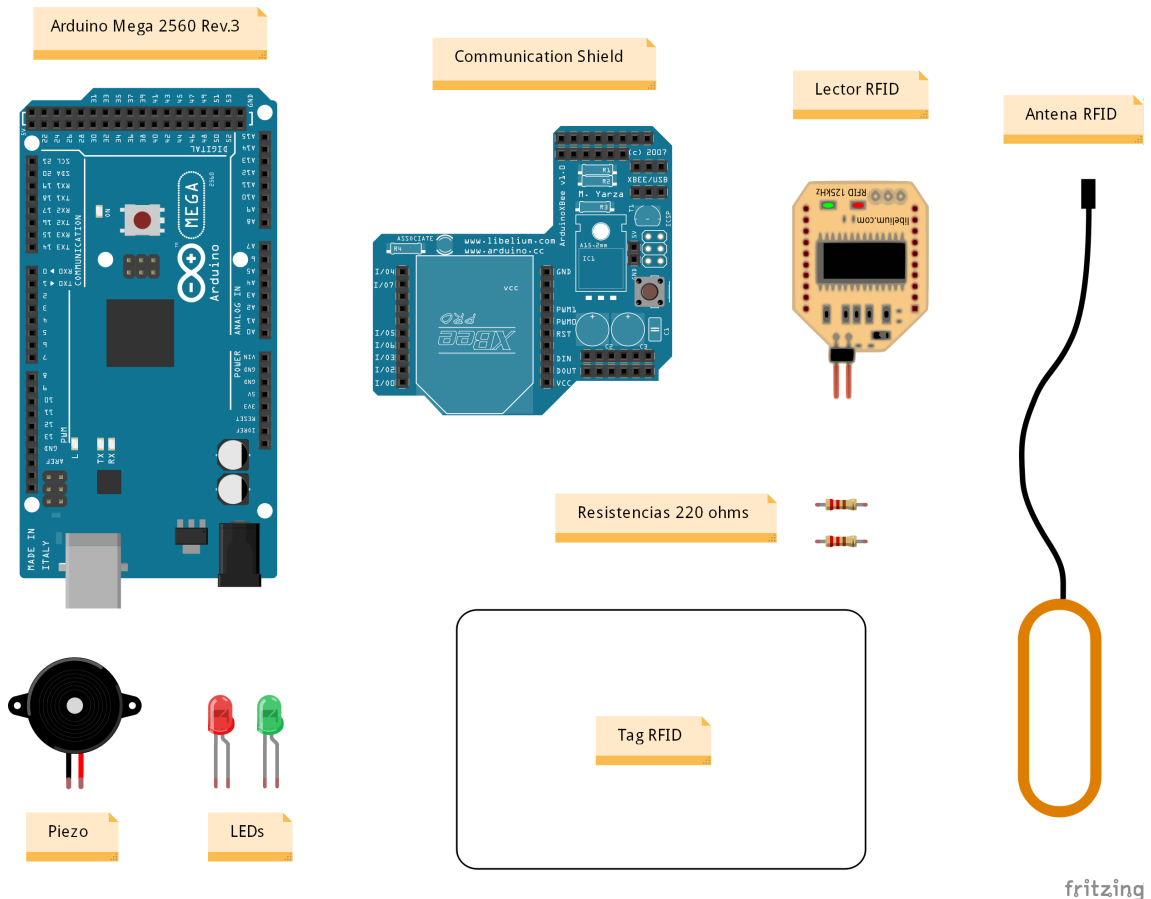


Figura 4.14: Componentes necesarios para el montaje RFID

Inicialmente, procedemos al montaje de los componentes hardware del lector de radiofrecuencia. De esta forma, el primer paso es acoplar la placa de comunicación (*Communication Shield*) a la placa microcontroladora Arduino Mega 2560 Rev3. Un detalle a tener en cuenta, es que los pines de comunicación de la *Communication Shield* (Tx y Rx) debemos conectarlos a los pines 14 y 15 del Arduino Mega, ya que de esta forma utilizamos un puerto serie adicional de los que dispone la placa, y dejamos libre el puerto serie correspondiente al conector USB (pines 0 y 1), el cual será utilizado para la comunicación del Arduino con la Raspberry Pi. Una vez hecho esto, el siguiente paso es acoplar el lector de radiofrecuencia en el *slot* dedicado de la placa de comunicación, el cual está especialmente diseñado para facilitar la comunicación entre módulos lectores de radiofrecuencia o módulos de comunicación inalámbrica y las placas Arduino. Tras haber acoplado el lector de radiofrecuencia, únicamente queda conectar la

antena del lector RFID, y establecer el montaje del piezo y de los LEDs con sus correspondientes resistencias tal y como muestra el esquema de la Figura 4.15.

Tras haber procedido al montaje del hardware de nuestro sistema de identificación mediante radiofrecuencia, ha llegado el momento de programarlo con el fin de que el sistema lea los identificadores de los transpondedores. Para ello, hacemos uso del entorno de desarrollo y programación (IDE) de Arduino, que podemos descargar de forma gratuita desde la página web oficial de Arduino¹⁵.

Una vez descargado e instalado el IDE de Arduino, procedemos a diseñar el código necesario para que el lector RFID lea los identificadores de los tags RFID que detecte. Dicho código puede ser consultado en el **Apéndice B.1: 'Obtención de identificadores'**.

El código mediante el cual vamos a obtener los identificadores de los transpondedores RFID, se encarga básicamente de establecer una comunicación entre el Arduino y el lector de radiofrecuencia, de forma que configuramos el lector en el modo de lectura automática (Modo de decodificación EM4102 - Sin contraseña) y esperamos a que sea detectado algún transpondedor. Una vez detectado un *tag*, leemos mediante una comunicación serie lector-*tag* los valores del identificador, y procedemos a imprimirlo por pantalla y a emitir un sonido mediante el piezo, con el fin de que el usuario sea consciente de que el *tag* ha sido detectado.

Tras averiguar los valores de los identificadores, únicamente debemos anotarlos para poder añadirlos, tal y como veremos en los siguientes pasos, en una base de datos que instalaremos en la Raspberry Pi, mediante la cual nos comunicaremos para averiguar si los identificadores leídos están presentes en dicha base de datos, y por lo tanto, autorizados por nuestra parte a utilizar nuestro sistema arcade.

4.5.5. Instalación de la base de datos

Una vez tenemos montados los componentes hardware Arduino, y hemos obtenido y anotado los identificadores de los transpondedores de radiofrecuencia que queremos autorizar a usar el sistema arcade, es el momento de proceder a la instalación de la base de datos que desde la Raspberry Pi va a gestionar la información de los transpondedores.

La base de datos elegida que vamos a utilizar en nuestro proyecto es SQLite. SQLite es un proyecto de dominio público, su creador es D. Richard Hipp, el cual implementa una pequeña librería de aproximadamente 500 kb, programado en el lenguaje C, totalmente libre y que tiene como función establecer un sistema de bases de datos relacional. Una de las primeras diferencias entre los motores de bases de datos convencionales es su arquitectura cliente/servidor, pues SQLite es independiente, simplemente se realizan llamadas a subrutinas o funciones de las propias librerías de SQLite, lo cual reduce ampliamente la latencia en cuanto

¹⁵[<http://arduino.cc/en/Main/Software>]

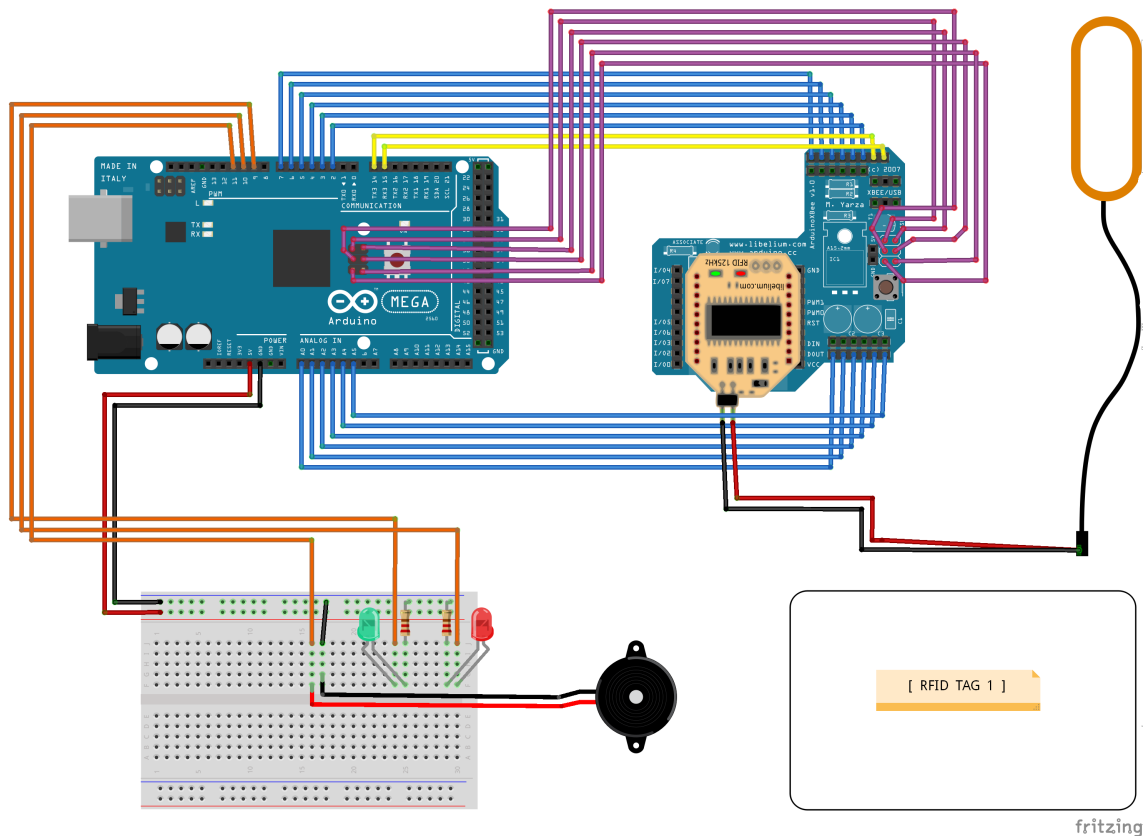


Figura 4.15: Montaje Arduino RFID

al acceso a las bases de datos. Con lo cual podemos decir que las bases de datos compuestas por la definición de las tablas, índices y los propios datos son guardados por un solo fichero estándar y en un solo ordenador.



Figura 4.16: Logo de SQLite

Veamos algunas razones para escoger SQLite como una herramienta de desarrollo:

- **Tamaño:** SQLite tiene una pequeña memoria, y una única biblioteca es necesaria para acceder a bases de datos, lo que lo hace ideal para aplicaciones de bases de datos incorporadas.

- **Rendimiento de base de datos:** SQLite realiza operaciones de manera eficiente y es más rápido que MySQL y PostgreSQL.
- **Portabilidad:** se ejecuta en muchas plataformas y sus bases de datos pueden ser fácilmente portadas sin ninguna configuración o administración.
- **Estabilidad:** SQLite es compatible con ACID, reunión de los cuatro criterios de Atomicidad, Consistencia, Aislamiento y Durabilidad
- **SQL:** implementa un gran subconjunto de la ANSI – 92 SQL estándar, incluyendo sub-consultas, generación de usuarios, vistas y triggers.
- **Interfaces:** cuenta con diferentes interfaces del API, las cuales permiten trabajar con C++, PHP, Perl, Python, Ruby, Tcl, groovy, etc.
- **Coste:** SQLite es de dominio público, y por tanto, es libre de utilizar para cualquier propósito sin costo y se puede redistribuir libremente.

Tras haber decidido el sistema de base de datos que vamos a utilizar, solo resta instalarla en nuestra Raspberry Pi, acción que realizamos ejecutando las órdenes de consola desde un terminal de la Raspberry Pi reflejadas en la Tabla 4.3.

```
$ sudo apt-get update
$ apt-get upgrade
$ sudo apt-get install sqlite3
```

Tabla 4.3: Órdenes para la instalación de SQLite

Dichas órdenes se encargan de actualizar los repositorios, actualizar los paquetes del sistema operativo, e instalar la última versión (v.3) de SQLite en la Raspberry Pi, respectivamente.

Una vez instalado SQLite en la Raspberry Pi, podemos manipular la base de datos de dos formas diferentes, desde una interfaz de consola o desde un programa escrito en alguno de los lenguajes de programación que son compatibles con la API de SQLite. Inicialmente vamos a usar el entorno de consola para crear la tabla en la que almacenaremos la información correspondiente a los transpondedores de radiofrecuencia, para ello abrimos un terminal de la Raspberry Pi, y tecleamos la orden `sqlite3 arcadeDB.db` para acceder al sistema de gestión de SQLite. Dicha orden abre mediante SQLite el archivo `arcadeDB.db` (el cual contiene nuestra base de datos), o lo crea nuevo en caso de que este no existiera. Tras acceder al sistema de gestión de SQLite, tecleamos los siguientes comandos para crear la tabla donde almacenaremos la información de los transpondedores (Véase la Tabla 4.4).

```
sqlite> CREATE TABLE identifications(  
...> serialNumber char(10) PRIMARY KEY,  
...> transpondedor char(20) NOT NULL,  
...> id char(10) UNIQUE NOT NULL,  
...> user char(50) NOT NULL,  
...> cargo char(50) NOT NULL  
...> );
```

Tabla 4.4: Órdenes para la creación de la tabla *identifications*

Interpretando las instrucciones mediante las cuales creamos la tabla *identifications*, podemos determinar que está formada por cinco valores, el `serialNumber`, el cual debe ser una cadena de 10 caracteres que equivalga al número de serie del transpondedor, que es la clave primaria, y por lo tanto debe ser un valor único y no nulo; el `transpondedor`, el cual puede ser una cadena de hasta 20 caracteres, que está destinada a describir el tipo de transpondedor registrado (`tagVerde`, `tagRojo`, `Card1`, `Card2`, etc.), y que no puede ser un valor nulo; el `id`, el cual es un valor de 10 caracteres en el que guardaremos el identificador del transpondedor asociado al número de serie introducido, valor que es definido como único y que no puede ser nulo, por lo que restringimos la inserción de transpondedores de forma que no podemos insertar *tags* cuyo ID o número de serie ya hayan sido insertados en la tabla; el `user`, que identificará el transpondedor con un nombre de usuario con un máximo de 50 caracteres, y que no puede ser nulo; y el `cargo`, que describe las funciones del usuario que estamos validando en el sistema (técnico, administrador, visitante, etc.), también con un máximo de 50 caracteres y sin la posibilidad de que sea un valor nulo.

Tras haber creado la tabla, procedemos a la creación de un programa en el lenguaje de programación Python que nos servirá para modificar los elementos de la base de datos de forma más amena, de tal forma que podamos insertar, borrar, y visualizar los elementos de la tabla *identifications* presente en nuestra base de datos. Dicho programa en Python puede ser encontrado en el Apéndice B.2.

En el código podemos ver cómo se definen funciones que mediante el uso de la API de SQLite para Python, nos permiten hacer consultas SQL sobre la base de datos que hemos creado previamente. De forma que podemos mostrar todo el contenido de la tabla *identifications*, así como insertar o eliminar un elemento de la misma. De esta forma, haciendo uso de las funciones implementadas, insertamos la información referente a los transpondedores de radiofrecuencia que queremos autorizar a usar el sistema arcade. De forma similar a los ejemplos de uso mostrados en el código, escribimos las líneas de código mostradas en la Tabla 4.5 en el programa, las cuales, tras haber sido ejecutado, habrán insertado en la tabla *identifications* los valores de los transpondedores autorizados por nuestra parte a usar el sistema arcade.


```
insertItem(conn, '0001994104', 'tagAzul', '0E001E6D78', 'Paco Martinez', 'Tecnico')
insertItem(conn, '0002158299', 'tagRojo', '100020EEDB', 'Juan Pablo', 'Administrador')
insertItem(conn, '0003199987', 'tagAmarillo', '0F0030D3F3', 'Maria Gomez', 'Visitante')
insertItem(conn, '0002787712', 'card1', '0D002A8980', 'Anna Marin', 'Visitante')
```

Tabla 4.5: Instrucciones para la inserción de elementos en la tabla *identifications*

identifications				
serialNumber	transpondedor	id	user	cargo

↓

identifications				
serialNumber	transpondedor	id	user	cargo
0001994104	tagAzul	0E001E6D78	Paco Martinez	Tecnico
0002158299	tagRojo	100020EEDB	Juan Pablo	Administrador
0003199987	tagAmarillo	0F0030D3F3	Maria Gomez	Visitante
0002787712	card1	0D002A8980	Anna Marin	Visitante

Figura 4.17: Estado de la tabla *identifications*

4.5.6. Comunicación con RaspberryPi

Una vez hemos instalado la base de datos, e introducido la información relativa a los *tags* de radiofrecuencia, es el momento de diseñar el sistema mediante el cual el Arduino se va a comunicar con la Raspberry Pi con el fin de verificar si un *tag* leído por el lector RFID está autorizado, o no, a usar el sistema arcade. Este sistema, el cual formará parte del diseño final de la máquina arcade, consta de dos secciones, el apartado referente al Arduino, y el apartado referente a la Raspberry Pi.

En el apartado correspondiente al Arduino, crearemos un nuevo programa que se encargue de que el lector de radiofrecuencia detecte los TAG RFID, y que envíe su identificador mediante una comunicación serie establecida con el Raspberry Pi a través del cable USB, de forma que además de alimentar la placa Arduino, también usaremos el cable USB para la comunicación y el envío de datos entre los dos dispositivos. Dicho código puede encontrarse en el Apéndice B.3.

En este programa, el Arduino establece dos puertos de comunicación serie en paralelo, de forma que mediante un puerto escucha al lector de radiofrecuencia, y por el otro se comunica con la Raspberry Pi, de tal manera, que al detectar un TAG RFID, el piezo emite un sonido para indicar al usuario que el *tag* ha sido reconocido, el lector lee su identificador, y este es transmitido a través del puerto serie del cable USB. Una vez hecho esto, el programa queda a la espera de recibir una respuesta por parte de la Raspberry Pi, la cual debe indicarle si el identificador leído está, o no, en la base de datos. Finalmente, tras haber recibido la respuesta por parte de la Raspberry Pi, en caso de que el identificador sí estuviera en la base de datos, haríamos parpadear la luz verde, mientras que si el identificador no estuviera presente en la base de datos, procederíamos a hacer parpadear la luz roja.

En el apartado correspondiente a la Raspberry Pi, crearemos un nuevo programa en Python que se encargará de establecer una comunicación serie con el Arduino, de tal forma que el programa quedará a la espera de recibir un identificador para ser validado. Este programa puede encontrarse en el Apéndice B.4.

Como hemos dicho anteriormente, este programa establece una comunicación serie con el Arduino y queda a la espera de recibir un identificador. Una vez recibido dicho identificador, se realiza una consulta SQL en la base de datos SQLite mediante el uso de su API, la cual nos indica si el identificador recibido a través del Arduino está presente, o no, en la base de datos. En caso de que no estuviera presente, enviaríamos al Arduino mediante la comunicación serie, el valor 'N', lo que el Arduino, tras haber sido correctamente programado, identificaría como que el identificador que ha transmitido no está autorizado, por lo que haría parpadear el led rojo. Por otro lado, si el identificador sí que está en la base de datos, enviaríamos el valor 'S', que de idéntica forma al procedimiento anterior, haría que el Arduino interpretara que el identificador transmitido sí está autorizado, y encendería el led verde. En la Figura 4.18 podemos observar un esquema del sistema de autenticación completamente terminado.

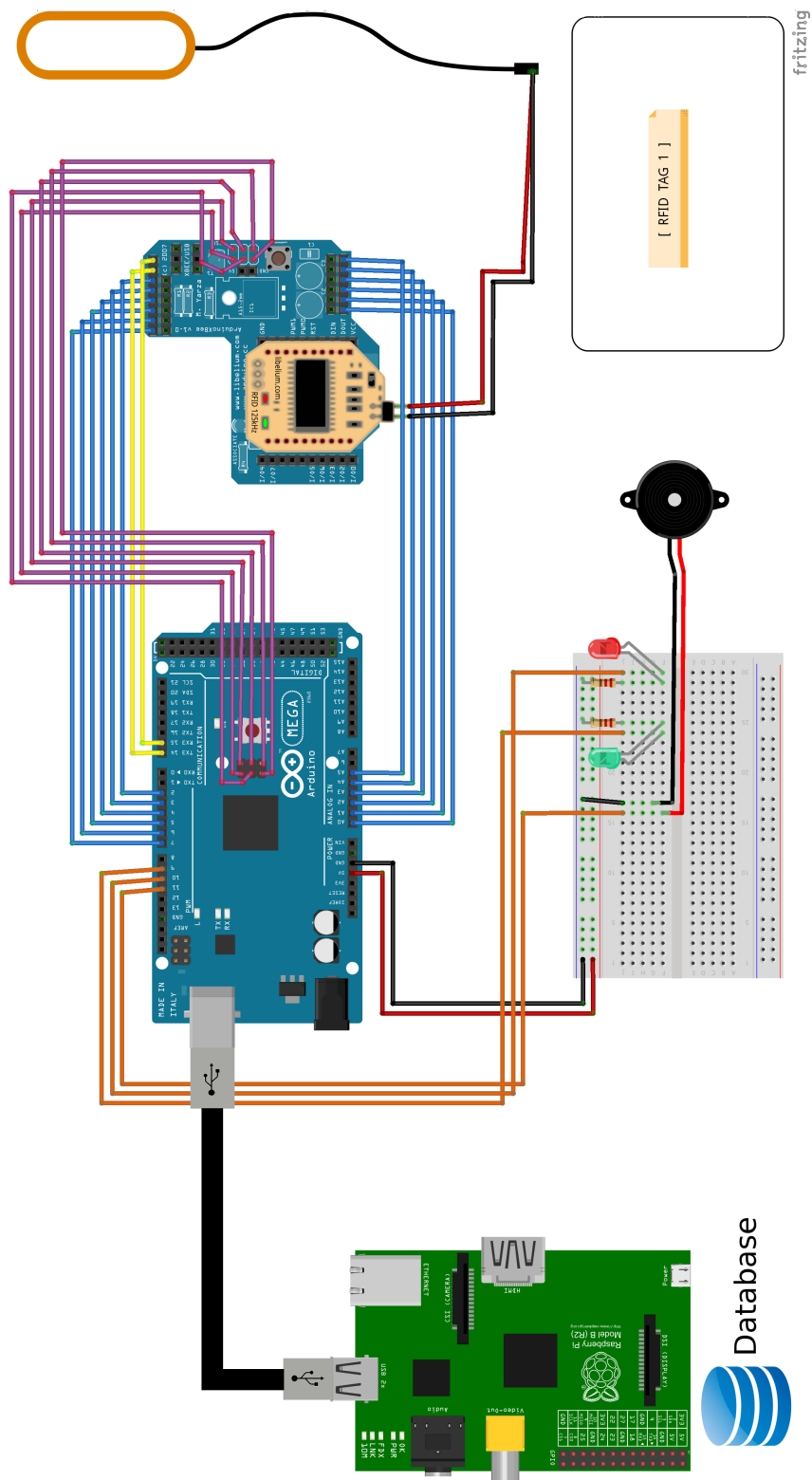


Figura 4.18: Sistema de autenticación

4.6 Gestión remota

Uno de los objetivos principales del proyecto es administrar el sistema de forma remota, de tal forma que una persona autorizada pueda acceder al sistema de gestión a través de cualquier ordenador con conexión a Internet, y tras autenticarse con sus credenciales de acceso, sea capaz de realizar gestiones relativas a la administración de la máquina arcade, tareas tales como editar la base de datos de los usuarios a los que se permite utilizar el sistema, añadir o eliminar videojuegos de la lista de juegos disponibles en el sistema, o realizar copias de seguridad de las bases de datos.

4.6.1. Diseño del portal de administración remoto

Para diseñar un portal de administración web seguro hemos optado por hacer uso de los lenguajes de programación HTML y PHP, de tal forma que mediante una combinación de ambos, obtendremos los recursos y la funcionalidad necesaria para diseñar un portal a medida que satisfaga nuestras necesidades. Además, en cuanto a la sección de programación HTML, haremos uso del *framework* de edición web Bootstrap¹⁶, el cual nos permite combinar las tecnologías HTML, CSS, y Javascript, de una forma sencilla.

El punto de partida para la administración telemática, y la primera sección de nuestro portal de administración, será una página de *login*, es decir, una página en la que el usuario podrá indicar su nombre de usuario y contraseña con el fin de obtener el acceso autorizado al portal, desde el cual podrá realizar los trámites pertinentes. Cabe destacar que la base de datos que contendrá los datos de acceso de los usuarios autorizados a realizar trámites en el portal será una base de datos diferente a la utilizada para almacenar las credenciales de los usuarios autorizados a usar la máquina arcade. Siendo esta una medida de seguridad adicional, la cual nos permitirá, además, ampliar el abanico de tecnologías utilizadas en el proyecto, ya que dicha base de datos será implementada en MYSQL debido a la facilidad de interacción de este sistema de base de datos con los lenguajes de programación HTML y PHP, que van a ser los lenguajes mediante los cuales diseñaremos el portal de administración. De esta forma, para permitir al usuario autenticarse, crearemos una base de datos en MYSQL, y definiremos una tabla formada por dos campos, un nombre de usuario, y una contraseña asociada a él.

Una vez definidos los datos de acceso para el usuario autorizado a realizar trámites en el portal online, nos centramos en el diseño de la página de *login*, la cual contendrá, tal como se observa en la Figura 4.19, dos secciones identificadas: un formulario para que el usuario pueda introducir su nombre de usuario y contraseña con el fin de ser verificado en la base de datos, y una sección de información en la que situaremos un pequeño párrafo informativo acerca de las

¹⁶<http://getbootstrap.com/>

características del portal. Podemos ver el código completo de la página de acceso a nuestro portal en el Apéndice C.1.



Los datos de usuario para acceder a la sección de gestión remota serán expedidos por el Museo de Informática de la Universidad Politécnica de Valencia, de forma personal, a los encargados de realizar dichos trámites.

Acceso Usuarios

Usuario

Contraseña

Entrar

Figura 4.19: Página de acceso al portal de trámites telemáticos

Tras haber sido autenticado de forma satisfactoria en el sistema, el usuario accederá a la página de inicio del portal, la cual podemos observar en la Figura 4.20, y cuyo código podemos encontrar en el Apéndice C.2. Centrándonos en el diseño de la página de inicio del portal, encontramos un menú lateral con las diferentes acciones que puede realizar el usuario (Figura 4.21), un menú superior desde el cual el usuario podrá cerrar la sesión o volver en cualquier momento a la página de inicio (Figura 4.22), y una sección central en la que se nos muestra información personal del usuario.

Diferenciando entre el conjunto de acciones que nos ofrece el menú lateral del portal, podemos encontrar acciones referidas a la gestión de identificadores de radiofrecuencia, la gestión del conjunto de ROMs de las que dispone el sistema, o la generación de copias de seguridad de las bases de datos.

En cuanto a las acciones referidas a la gestión de identificadores de radiofrecuencia, tal y como podemos observar en la Figura 4.23, disponemos de las opciones de listar los identificadores existentes, añadir identificadores nuevos, modificar la información referida a algún identificador, o eliminar un identificador dado existente en el sistema.

Al seleccionar la opción Listado de Identificadores, tal y como podemos observar en la Figura 4.24, nos aparecerá un listado del conjunto de identificadores de radiofrecuencia que hay introducidos en el sistema, así como toda la información asociada a cada uno de los identificadores existentes. Estos identificadores serán los que permitirán el acceso autenticado de los usuarios a hacer uso de la máquina arcade.

A su vez, con el fin de añadir un nuevo identificador en el sistema, seleccionaremos la opción Alta Nuevo Identificador, la cual nos conducirá a una página

Arcade Machine Emulator - Sistema de gestión remota

Usuario: [Roig Coves, Alvaro]

Perfil de Usuario

museu
informàtica

Identificadores <

Gestión de ROMs <

Datos de usuario

Usuario:	arco
Nombre:	Alvaro
Apellidos:	Roig Coves

Figura 4.20: Página de inicio del portal de trámites telemáticos



Figura 4.21: Menú lateral del portal de trámites telemáticos

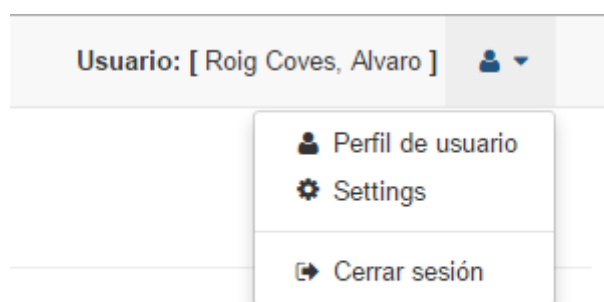


Figura 4.22: Menú superior del portal de trámites telemáticos

con un formulario en la que rellenar todos los campos necesarios para la inserción de un nuevo identificador en la base de datos. Podemos ver algunos de estos campos a rellenar en la Figura 4.25.

Por otra parte, el portal también nos da la posibilidad de modificar la información asociada a un identificador de radiofrecuencia dado y existente en el sistema; para ello accederemos a la sección *Modificar Identificador*, la cual nos mostrará un listado de los identificadores presentes en el sistema, de los cuales deberemos elegir uno, tal como muestra la Figura 4.26. Una vez elegido el identificador del cual queremos modificar sus datos asociados, se nos mostrará un formulario como el presente en la Figura 4.27, el cual contendrá los datos del identificador, y se nos permitirá modificar aquellos que sean posibles. Además de poder modificar un identificador, también tenemos la posibilidad de eliminarlo del sistema, acción que realizaremos mediante la opción *Eliminar identificador* del menú lateral, en la que de forma análoga a modificar el identificador, deberemos selec-



Figura 4.23: Acciones asociadas a la gestión de identificadores de radiofrecuencia

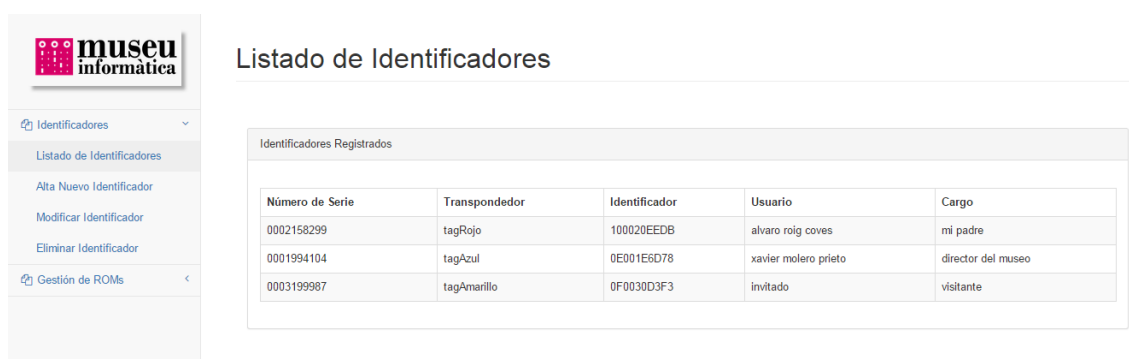
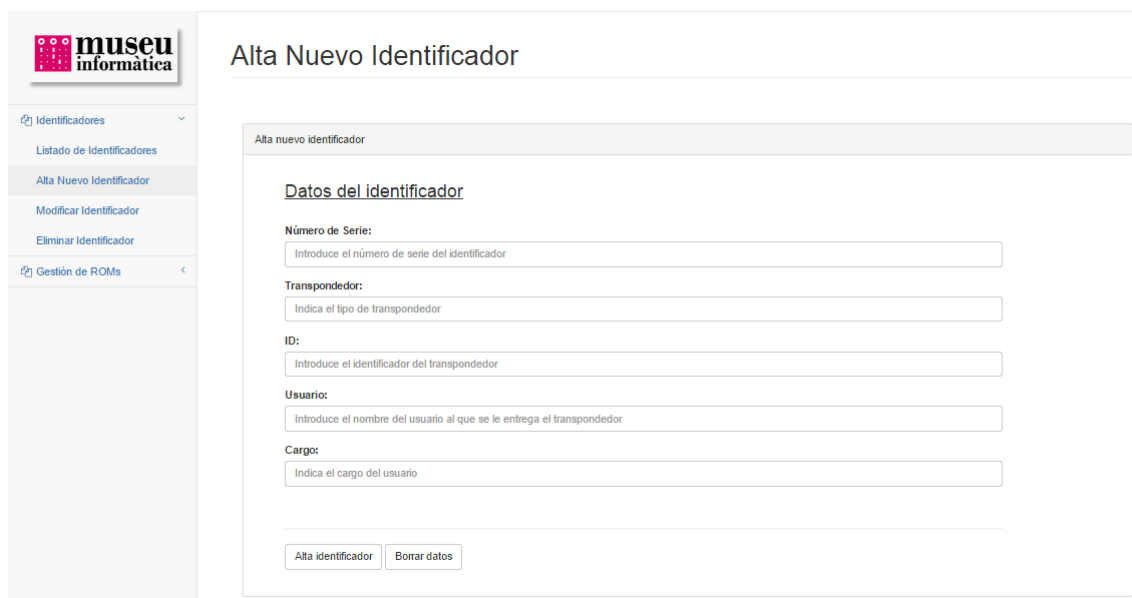


Figura 4.24: Listado de identificadores de radiofrecuencia

cionar uno de los que ya están presentes en el sistema, para que este sea eliminado posteriormente.

Además de realizar acciones referidas a la gestión de identificadores de radiofrecuencia, tal y como podemos observar en la Figura 4.28, también disponemos de acciones que nos permiten gestionar las ROMs existentes en el sistema; entre ellas, diferenciamos las de listar el conjunto de ROMs existentes, añadir una nueva ROM, o eliminar una de las que ya disponemos en el sistema.

Para ver el conjunto de ROMs disponibles en el sistema nos dirigiremos a la sección Listado de ROMs del menú lateral, en la que tal y como muestra la Figura 4.29, encontraremos una lista de todos los videojuegos a los que los usuarios podrán jugar una vez identificados correctamente en el sistema. Además, haciendo clic en cualquiera de los elementos de la lista, podremos descargarlos desde el servidor, lo que nos permite poder guardarlos y llevarlos con nosotros.

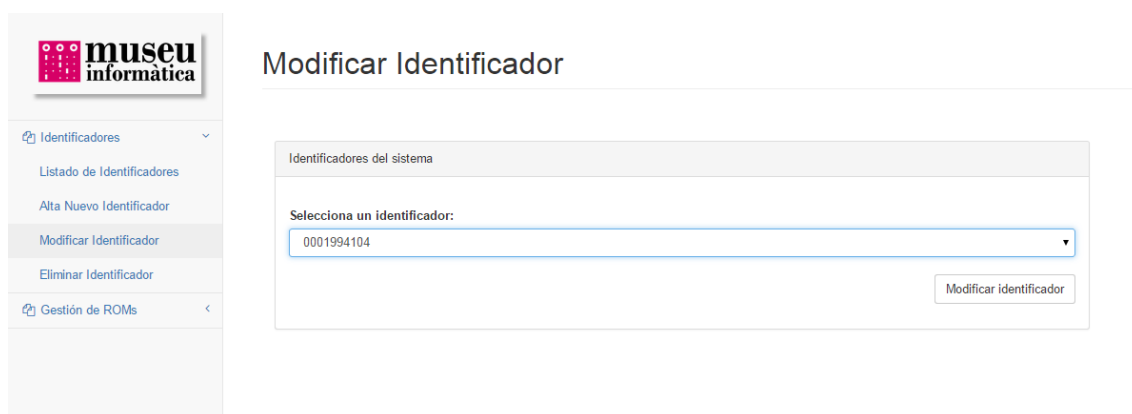


The screenshot shows the 'Alta Nuevo Identificador' page. On the left is a sidebar with the 'museu informática' logo and a menu with options: 'Identificadores', 'Listado de Identificadores', 'Alta Nuevo Identificador' (highlighted), 'Modificar Identificador', 'Eliminar Identificador', and 'Gestión de ROMs'. The main content area is titled 'Alta Nuevo Identificador' and contains a form with the following fields:

- Datos del identificador**
- Número de Serie:** Introduce el número de serie del identificador
- Transpondedor:** Indica el tipo de transpondedor
- ID:** Introduce el identificador del transpondedor
- Usuario:** Introduce el nombre del usuario al que se le entrega el transpondedor
- Cargo:** Indica el cargo del usuario

At the bottom of the form are two buttons: 'Alta identificador' and 'Borrar datos'.

Figura 4.25: Dar de alta un nuevo identificador de radiofrecuencia



The screenshot shows the 'Modificar Identificador' page. The sidebar is identical to the previous figure. The main content area is titled 'Modificar Identificador' and contains a form with the following elements:

- Identificadores del sistema**
- Selecciona un identificador:** A dropdown menu showing '0001994104'.
- Modificar identificador** button.

Figura 4.26: Seleccionar un identificador para ser modificado

Otra de las acciones que nos permite el portal referente a la gestión de las ROMs del sistema es la de añadir una nueva ROM. Esta acción la realizaremos desde la sección Añadir ROM del menú lateral; en dicha sección encontraremos un formulario como el presente en la Figura 4.30, mediante el cual podremos seleccionar una ROM y subirla al sistema mediante una conexión FTP; esto requerirá por nuestra parte la instalación y configuración de un servicio de FTP para realizar dichas transferencias a través del portal. De esta forma, el conjunto de instalaciones para disponer de los servicios necesarios los expondremos en la Sección 4.6.2. Tras haber seleccionado la ROM y haber iniciado la subida del fichero al servidor, el portal nos mostrará, tal y como podemos observar en la Figura 4.31, un reporte de la información asociada a la subida del fichero, en la que se nos informará de las operaciones realizadas y del estado final del intento de subida del fichero.

Modificar Identificador

Modificar datos de identificador

Datos del identificador

Número de Serie:
0001994104

Transpondedor:
tagAzul

ID:
0E001E6D78

Usuario:
xavier molero prieto

Cargo:
director del museo

Guardar Identificador

Figura 4.27: Modificar la información asociada a un identificador

Para finalizar con la sección de gestión de ROMs, cabe destacar la posibilidad por parte del administrador del portal, de eliminar cualquiera de las ROMs presentes en el servidor. Para realizar dicha acción, nos dirigiremos a la sección **Eliminar ROM** del menú lateral, en la que se nos conducirá a una página en la que como muestra la Figura 4.32, deberemos seleccionar una de las ROMs presentes en el sistema con el fin de que sea eliminada.

Finalmente, cabe destacar el uso de dos ficheros PHP adicionales, los cuales son `conexion.php` (presente en el Anexo C.3) y `logout.php` (presente en el Anexo C.4). Dichos códigos se incluyen y asocian tanto en la página de acceso como en la página de inicio del portal, con el fin de gestionar el acceso a las bases de datos en el momento en que iniciamos una sesión en el portal, y el cierre de sesión seguro en el momento en que el usuario decide cerrar la sesión.

4.6.2. Instalación de servidores en la RaspberryPi

Con tal de ofrecer a los administradores de la máquina arcade un sistema de gestión fiable y sencillo desde el cual realizar el conjunto de operaciones necesarias para garantizar su correcto funcionamiento, hemos desarrollado un portal de administración desde el que realizar dichos trámites a través de Internet. De esta forma, debemos garantizar el acceso a dicho portal por parte de los usuarios. Consecuentemente, en esta sección nos disponemos a instalar todos los servidores necesarios para mantener operativo y accesible nuestro portal administrativo.

Debido a las necesidades del portal de gestión, precisamos hacer uso de un servidor web, un servidor FTP, y una base de datos que ofrezca una interacción sencilla con dichos servidores. Dadas las necesidades anteriormente nombradas,



Figura 4.28: Acciones asociadas a la gestión de ROMs



Figura 4.29: Listado de las ROMs presentes en el sistema

vamos a hacer uso de un servidor web *Apache*, un servidor ftp *VSFTPD*, y una base de datos *MySql*.

Instalación del servidor Web

Como hemos comentado anteriormente, el servidor que vamos a instalar en la Raspberry Pi es un servidor web *Apache*. El primer paso para la instalación del mismo es crear (en caso de que no exista) y dar permisos al grupo de trabajo que usa *Apache* por defecto. Realizamos dichas acciones ejecutando las órdenes mostradas en la Tabla 4.6.

Una vez hecho esto, nos disponemos a actualizar los repositorios, y a instalar *Apache*, así como las extensiones *PHP* y *SQLite* que utilizará el servidor web para

Figura 4.30: Formulario para añadir una ROM en el sistema

Añadir ROMs

Figura 4.31: Reporte de la subida de una ROM en el sistema

gestionar los recursos de la máquina arcade. Podemos encontrar las órdenes de consola necesarias para este propósito en la Tabla 4.7.

Tras haber instalado los servicios anteriores, reiniciamos el servidor web mediante la orden `sudo /etc/init.d/apache2 restart`.

Instalación de la base de datos

Una vez hemos instalado el servidor web, nos disponemos a instalar una base de datos que nos garantice el acceso autenticado al portal. La base de datos que vamos a instalar es MySQL, y además, instalaremos también el software de gestión de bases de datos *PHPMyAdmin*, desde el cual poder gestionar todo lo referente a las bases de datos *MySQL*. Para ello, ejecutaremos las órdenes mostradas en la Figura 4.8.

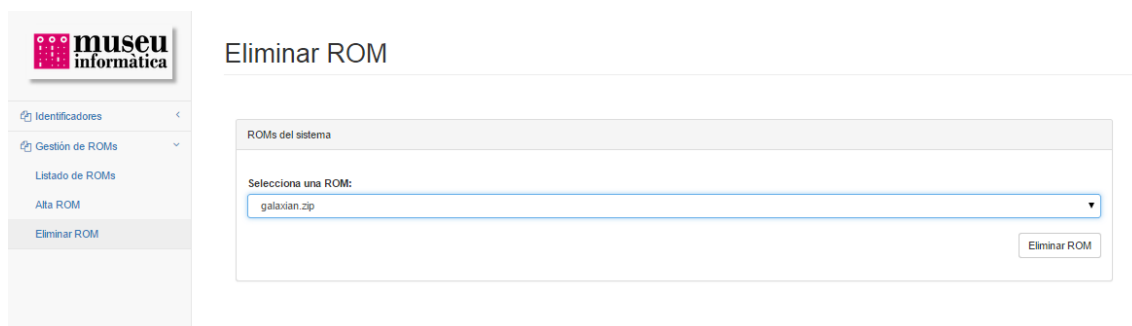


Figura 4.32: Eliminar ROM del sistema

```
$ sudo addgroup www-data
$ sudo usermod -a -G www-data www-data
```

Tabla 4.6: Órdenes para crear y dar permisos al grupo de trabajo que utiliza *Apache*

Tras haber instalado los servidores anteriores, debemos modificar el fichero `php.ini` con el fin de indicar al servidor web que haremos uso de los servicios de *MySQL* y *SQLite*. Dichas acciones las realizamos insertando las líneas `extension=mysql.so` y `extension=sqlite.so` después de la cabecera [Dynamics Extensions] del fichero situado en la ruta `/etc/php5/apache2/php.ini`.

Además, deberemos indicar en el fichero de configuración de *Apache* que hemos habilitado el gestor de bases de datos *PHPMyAdmin*. Dicha acción la realizamos añadiendo la línea `Include /etc/phpmyadmin/apache.conf` al final del fichero situado en la ruta `/etc/apache2/apache2.conf`, y reiniciando el servidor mediante la orden `sudo /etc/init.d/apache2 restart`.

Una vez hecho esto, ejecutamos las órdenes de consola situadas en la Tabla 4.9 para crear los ficheros de configuración necesarios para la asociación de todos los servidores que hemos instalado.

```
$ sudo ln -s /etc/phpmyadmin/apache.conf /etc/apache2/conf.d/phpmyadmin.conf
$ sudo /etc/init.d/apache2 reload
```

Tabla 4.9: Órdenes para crear ficheros de configuración

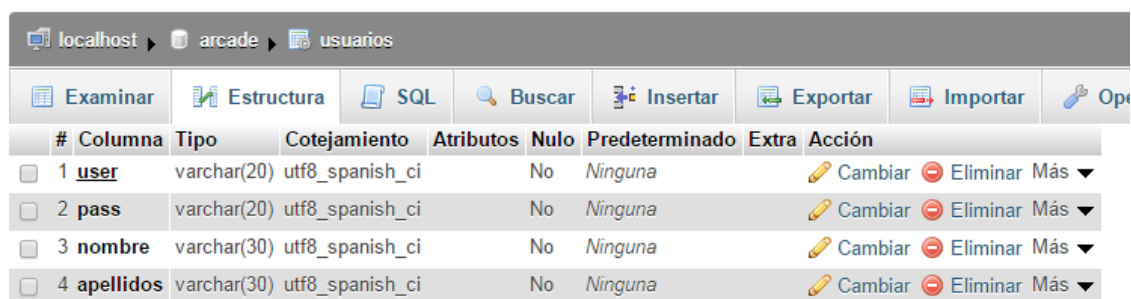
Finalmente, situamos todos los códigos pertenecientes al portal de gestión en la ruta `/var/www/`, la cual es donde se sitúan los archivos públicos del servidor web; y mediante el uso de la herramienta de gestión de bases de datos *PHPMyAdmin*, creamos una base de datos semejante a la de la Figura 4.33 para guardar las credenciales de acceso de los administradores del portal.

```
$ sudo apt-get update
$ sudo apt-get install apache2 php5 libapache2-mod-php5
$ sudo apt-get install php5-sqlite
```

Tabla 4.7: Órdenes para actualizar los repositorios e instalar *Apache*

```
$ sudo ifup lo
$ sudo apt-get install mysql-server mysql-client php5-mysql phpmyadmin
```

Tabla 4.8: Órdenes para instalar *MySQL* y *PHPMyAdmin*



#	Columna	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra	Acción
<input type="checkbox"/>	1	user	varchar(20)	utf8_spanish_ci	No	Ninguna		Cambiar Eliminar Más ▼
<input type="checkbox"/>	2	pass	varchar(20)	utf8_spanish_ci	No	Ninguna		Cambiar Eliminar Más ▼
<input type="checkbox"/>	3	nombre	varchar(30)	utf8_spanish_ci	No	Ninguna		Cambiar Eliminar Más ▼
<input type="checkbox"/>	4	apellidos	varchar(30)	utf8_spanish_ci	No	Ninguna		Cambiar Eliminar Más ▼

Figura 4.33: Tabla de *MySQL* para guardar las credenciales de acceso de los administradores del portal

Instalación del servidor FTP

El portal de gestión telemático nos ofrece, como uno de los principales elementos funcionales, la posibilidad de gestionar el conjunto de ROMs disponibles en la máquina arcade. Para realizar las transacciones de dichas ROMs desde el equipo local del administrador hacia la máquina arcade haremos uso de un servidor FTP, el cual nos permitirá de manera sencilla, copiar los archivos que contienen las ROMs, a los directorios de la máquina arcade destinados a contener dichos ficheros.

Consecuentemente, hemos elegido hacer uso del servidor FTP *VSFTPD*. Para su instalación ejecutaremos la orden de consola `sudo apt-get install vsftpd`, y tras la misma, deberemos modificar el fichero `/etc/vsftpd.conf` descomentando las líneas `local-enable=YES` y `write-enable=YES`; lo que permitirá la escritura de archivos en la Raspberry Pi por parte de los usuarios que administren la máquina arcade.

Finalmente, para la correcta puesta en marcha del servidor FTP, ejecutaremos la orden `sudo service vsftpd restart`.

4.7 Inicio automático del sistema

Con el fin de iniciar la funcionalidad completa de nuestra máquina arcade, vamos a crear un *script* de inicio, el cual será ejecutado automáticamente al conectar la Raspberry Pi a la corriente eléctrica.

Dicho *script*, presente en el Apéndice A.1, inicia una interfaz de consola como la mostrada en la Figura 4.34, en la que se nos indican las instrucciones para autenticarnos en el sistema, y pone a la Raspberry Pi a la espera de que el sistema Arduino detecte un identificador de radiofrecuencia; una vez detectado, el identificador del transpondedor de radiofrecuencia es comunicado a la Raspberry Pi para ser verificado en la base de datos.

En caso de autenticación negativa, la interfaz nos indica que la tarjeta o *tag* de radiofrecuencia introducida no está autorizada, y vuelve a quedar a la espera de un nuevo identificador. En el caso de que dicho identificador sea válido, se muestra un mensaje de autenticación correcta a través de la interfaz, se activan los controles arcade, y se inicia el emulador. Una vez iniciado el emulador arcade, podemos navegar a través de él mediante los controles arcade, y cuando hayamos acabado de jugar, debemos apagar el interruptor situado en el mueble de la máquina a fin de apagar el sistema.

Finalmente, cabe destacar que en el momento en el que encendemos la Raspberry Pi, además de iniciarse el *script* de inicio, también se ponen en ejecución los servidores encargados de ofrecer el sistema de gestión telemático, por lo tanto, los usuarios administradores de la máquina arcade podrán gestionar la máquina de forma remota una vez haya sido encendida.

CAPÍTULO 5

Construcción de la máquina arcade

En este capítulo incluimos una serie de imágenes mediante las cuales mostramos cada uno de los montajes hardware que hemos realizado durante el transcurso del trabajo; los cuales, en su conjunto, integran nuestra máquina arcade.

5.1 Laboratorio de trabajo

Para comenzar, cabe destacar que la mayor parte del trabajo ha sido realizado en el laboratorio de TFG del edificio 1G de la UPV, el cual podemos observar en la Figura 5.1.

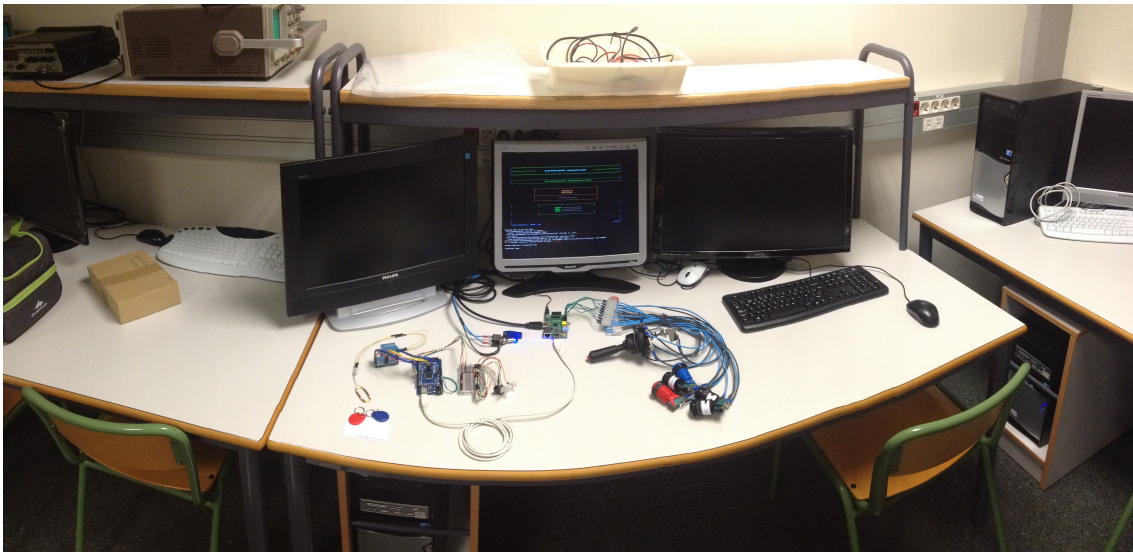


Figura 5.1: Laboratorio de trabajo

5.2 Detalles de los componentes

En cuanto al sistema de autenticación diseñado mediante Arduino, mostrado en la Figura 5.2, podemos destacar el uso de un módulo de comunicación y lectura de transpondedores de radiofrecuencia (Figura 5.3), y el uso de tarjetas y llaveros RFID (Figura 5.4).

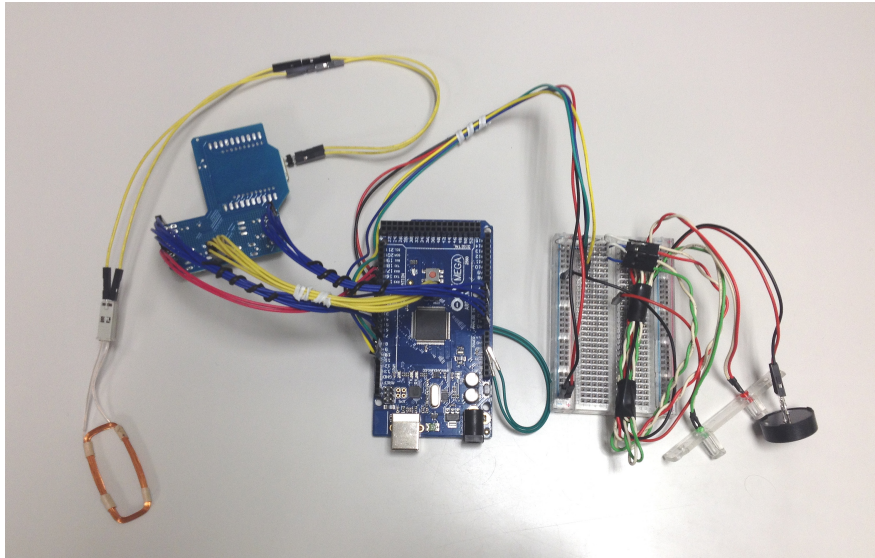


Figura 5.2: Sistema de autenticación Arduino

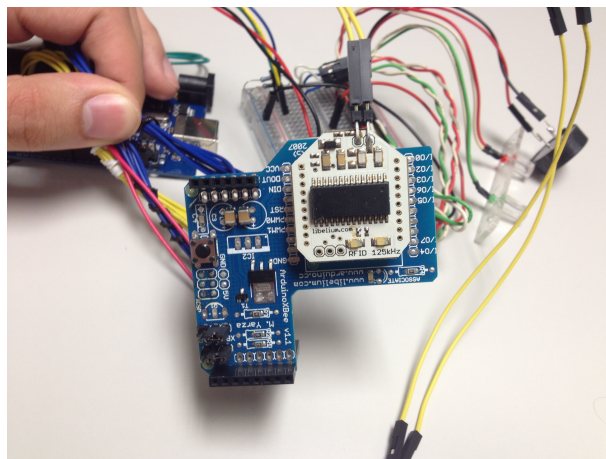


Figura 5.3: Módulo RFID Arduino

Como hemos comentado anteriormente, el lugar donde se procesa la mayoría de la información de nuestra máquina arcade es una Raspberry Pi. En este trabajo, tal como observamos en la Figura 5.5, hacemos uso de una Raspberry Pi modelo B.



Figura 5.4: Tarjetas de radiofrecuencia utilizadas



Figura 5.5: Raspberry Pi utilizada

Por otra parte, uno de los cableados más laboriosos es el correspondiente al de los controles físicos que nos permiten interactuar con el emulador arcade, dicho cableado, mostrado en la Figura 5.6, comunica los botones y palancas arcade con la Raspberry Pi.

Otro montaje realizado es el sistema de alimentación para conectar la Raspberry Pi a la corriente eléctrica, podemos observar dicho montaje en la Figura 5.7.

Finalmente, tras unir todos los componentes que hemos desarrollado, tenemos una visión de conjunto de nuestra máquina arcade, la cual observamos en la Figura 5.8.

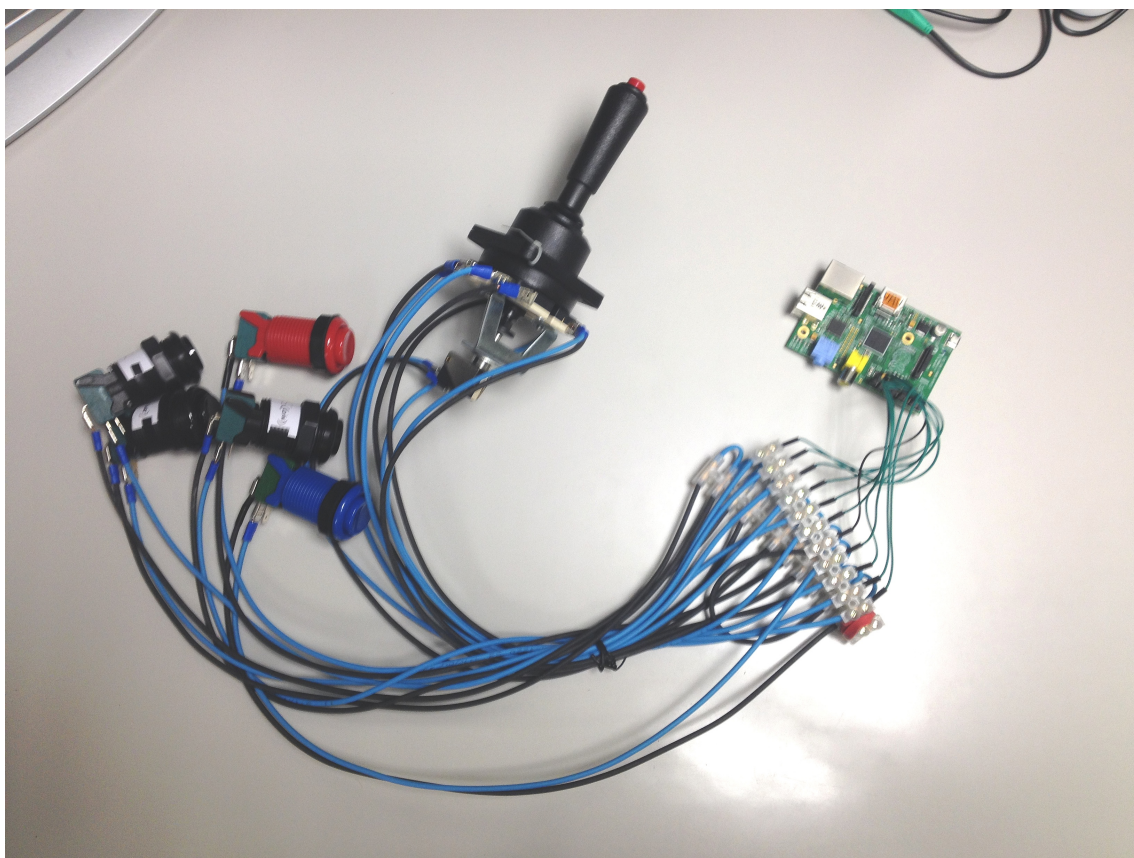


Figura 5.6: Sistema de control arcade

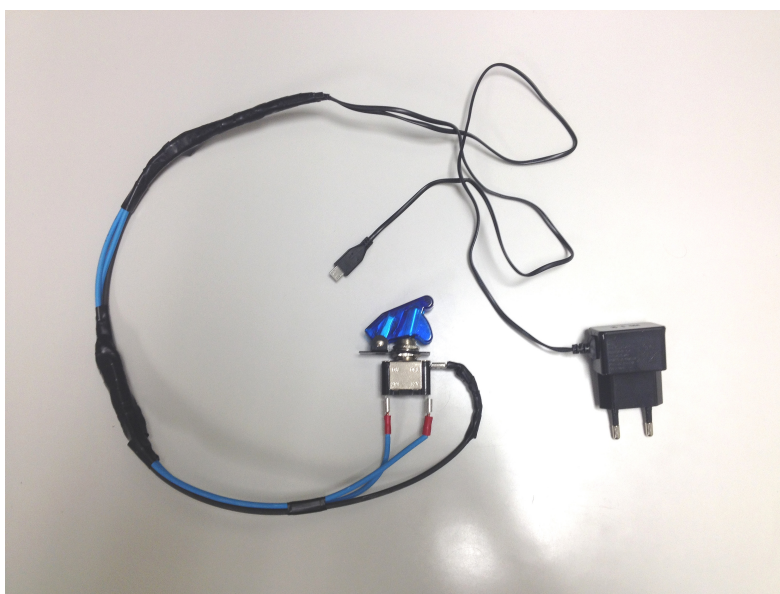


Figura 5.7: Sistema de alimentación de la Raspberry Pi

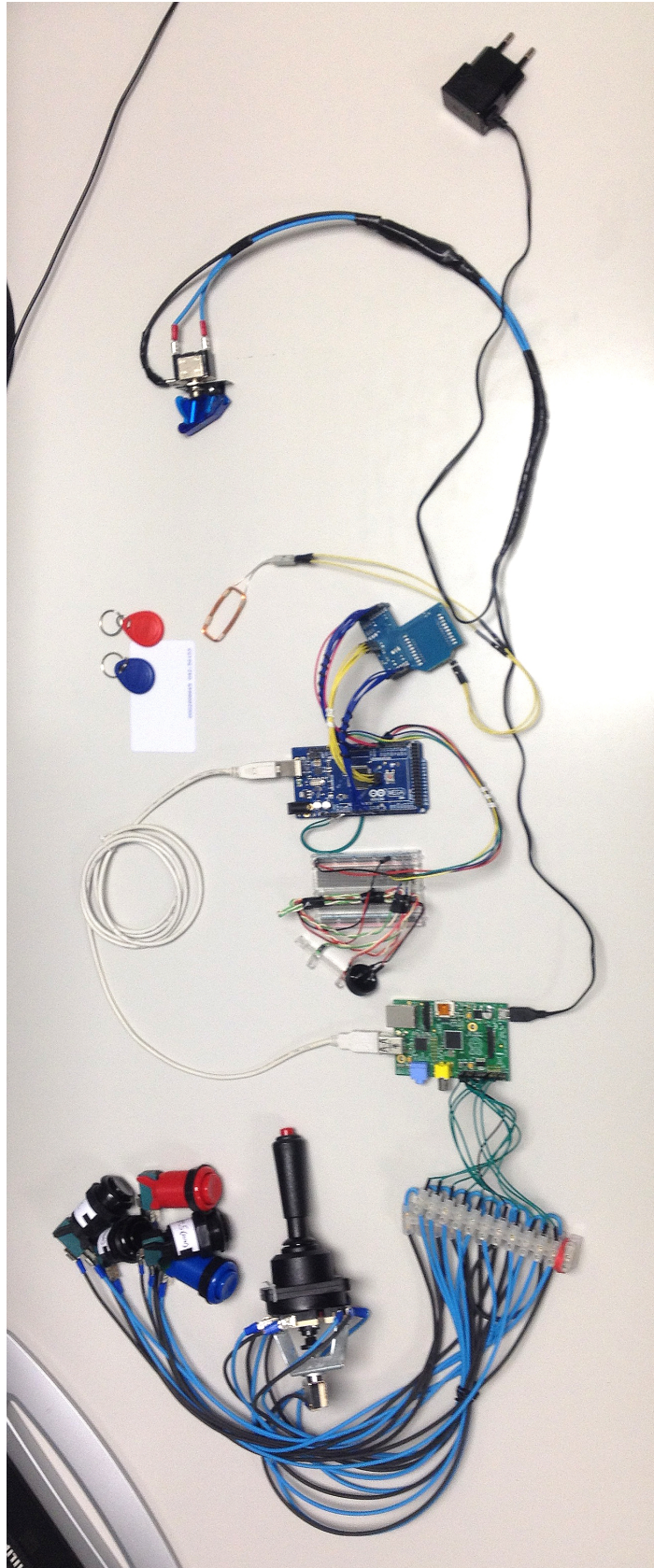


Figura 5.8: Montaje final del hardware

CAPÍTULO 6

Conclusiones

En este último capítulo incluimos una sección en la que describimos los diferentes problemas y contratiempos que hemos sufrido durante el transcurso del trabajo. Además, se plantean las consideraciones finales obtenidas a partir del trabajo desarrollado, realizando un resumen final, y analizando el conjunto de funcionalidades implementadas

6.1 Contratiempos y problemas sufridos

A lo largo del desarrollo del trabajo hemos sufrido contratiempos y problemas tanto a la hora de realizar implementaciones del software, como a la hora de llevar a cabo los montajes físicos necesarios. Exponemos a continuación un breve resumen de cada uno de ellos, así como la solución que le encontramos.

- **Comunicación serie Arduino-RaspberryPi:** Uno de los puntos más conflictivos del desarrollo del trabajo ha sido establecer una comunicación serie eficaz entre el Arduino y la Raspberry Pi. Dicha comunicación la realizamos mediante un cable USB, y el problema que encontramos fue que no conseguíamos enviar correctamente los identificadores de radiofrecuencia desde el Arduino hacia la Raspberry Pi. El problema ligado a la representación de la información se solucionó de forma sencilla al darnos cuenta de que tras cada identificador enviado desde el Arduino, se incluía un retorno de carro de forma automática; de tal forma que únicamente debíamos eliminar dicho retorno de carro.
- **Diseño del mueble arcade:** Con el fin de exponer la máquina arcade en el Museo de Informática nos dispusimos a fabricar un mueble especial en el que establecer y cablear todo el sistema; para ello, contactamos con diversas organizaciones y constructoras expertas en el diseño y fabricación de muebles arcade, pero tras varios acercamientos, no conseguimos encauzar este

propósito, por lo que la fabricación del mueble necesario para la exposición de la máquina arcade la hemos pospuesto para llevarla a cabo más adelante.

- **Conectividad inalámbrica:** Otro de los problemas que nos hemos encontrado en el trabajo, y que ha supuesto un gasto adicional en el presupuesto, ha sido la necesidad de conectar la máquina arcade a Internet de forma inalámbrica, por lo que hemos tenido que adquirir un adaptador WiFi inalámbrico compatible con nuestra Raspberry Pi.

6.2 Consideraciones finales

Durante el transcurso del proyecto, tras haber realizado un breve trabajo de investigación acerca de las máquinas arcade, en el que hemos indagado sobre su impacto en la sociedad, así como qué máquinas y videojuegos eran los más populares y extendidos, nos hemos centrado en diseñar una máquina arcade mediante el uso combinado de un microcontrolador Arduino y de un dispositivo empotrado de la familia Raspberry Pi.

Dichos dispositivos combinan sus funcionalidades con el fin de establecer un sistema de juego compacto, mediante el cual el usuario final pueda disfrutar de una experiencia de juego similar a la que se podría tener haciendo uso de una máquina arcade de la época del apogeo de las máquinas recreativas, pero con ventajas evidentes como la integración de más de un videojuego en un mismo dispositivo, el acceso autenticado necesario para hacer uso de la misma, o incluso la posibilidad por parte de los encargados del funcionamiento del sistema de realizar configuraciones de la máquina de forma remota a través de Internet.

Entre las diferentes etapas del diseño e implementación de la máquina arcade podemos destacar tanto el trabajo realizado con el microcontrolador Arduino, mediante el cual permitimos al usuario final autenticarse haciendo uso de tarjetas de radiofrecuencia, como el trabajo realizado con el dispositivo empotrado Raspberry Pi, mediante el cual llevamos a cabo acciones tan variopintas como comprobar si el usuario final está autorizado a usar el sistema, la emulación y control de los videojuegos arcade, o la puesta en marcha de los servidores necesarios para ofrecer un portal telemático desde el cual realizar configuraciones remotas de la máquina arcade.

Por todo lo citado anteriormente, tras haber diseñado, implementado, y configurado cada uno de los elementos que conforman nuestra máquina arcade, hemos conseguido construir un sistema de emulación de una máquina recreativa fiel a las máquinas recreativas originales, añadiendo, de forma adicional, elementos funcionales que la dotan de sistemas de seguridad y de administración propios, los cuales son habituales y necesarios en los nuevos dispositivos electrónicos de nuestra época.

De esta forma, mediante la exposición y uso de nuestra máquina arcade por parte del Museo de Informática de la Universitat Politècnica de València, podemos considerar el dispositivo que hemos diseñado y construido como un elemento que contribuye a difundir el patrimonio cultural informático de los años ochenta y noventa del siglo XX, siendo este un punto de unión entre el pasado y el presente de la informática.

Bibliografía

- [1] ATMEL. Artículo sobre el funcionamiento de la tecnología RFID y el estándar ISO-14443 redactado por la empresa ATMEL. *Requirements of ISO/IEC 14443 Type B Proximity Contactless Identification Cards*. Consultado en <http://www.atmel.com/Images/doc2056.pdf>.
- [2] Massimo Banzi. *Getting Started with Arduino*. Maker Media. Inc, 2011.
- [3] Ángel Cobo. *PHP y MySQL. Tecnología para el desarrollo de aplicaciones web*. Ediciones Díaz de Santos, 2005.
- [4] Rafael Enríquez Herrador. *Guía de Usuario de Arduino. I.T.I. Sistemas*, Universidad de Córdoba. Noviembre, 2009.
- [5] Alejandro Esquivá Rodríguez. *Geeky Theory. Tutorial Raspberry Pi - 9. Servidor FTP*. Consultado en <https://geekytheory.com/tutorial-raspberry-pi-9-servidor-ftp>. Mayo, 2013.
- [6] Alejandro Esquivá Rodríguez. *Geeky Theory. Tutorial Raspberry Pi - 15. Instalación de Apache + MySql + PHP*. Consultado en <https://geekytheory.com/tutorial-raspberry-pi-15-instalacion-de-apache-mysql-php>. Agosto, 2013.
- [7] Carlos Forcada. Regreso al Pasado: Recreativas españolas. *MeriStation Magazine*. Consultado en <http://www.meristation.com/pc/reportaje/regreso-al-pasado-recreativas-espanolas/1975205>. Mayo, 2014.
- [8] Bill Glover y Himanshu Bhatt. *RFID Essentials. Theory in Practice*. O'Reilly Media, 2006.
- [9] Olivier Heurtel. *PHP 5.5. Desarrollar un sitio web dinámico e interactivo*. Ediciones ENI, 2014.
- [10] Paris Kitsos y Yan Zhang. *RFID Security. Techniques, Protocols and System-On-Chip Design*. Springer Science, 2008.
- [11] Daniel Martín Maldonado. SQLite, el motor de base de datos ágil y robusto. *Aplicaciones para empresas*. Consultado en

- <http://www.empresayeconomia.es/aplicaciones-para-empresas/sqlite-el-motor-de-base-de-datos-agil-y-robusto.html>. Julio, 2008.
- [12] Michael Margolis. *Arduino Cookbook*. O'Reilly Media, 2014.
- [13] Guías Nunoarcade. Conexión de un microswitch arcade. Cómo funcionan. *Nunoarcade.com*. Consultado en <http://nunoarcade.blogspot.com.es/2013/03/conexion-de-un-microswtich-arcade-como.html>. Marzo, 2013.
- [14] David Quesada Sydaroa. *Cómo convertir un PC en una máquina multijuegos. La guía de estrategia oficial*. AndxDD, 2012.
- [15] Matt Richardson y Shawn Wallace. *Getting Started with Raspberry Pi. Make: projects. O'Reilly and Associate Series*. O'Reilly Media, 2012.
- [16] Eben Upton y Gareth Halfacree. *Raspberry Pi User Guide*. John Wiley and Sons, 2014.
- [17] Roy Want. *RFID Explained. A Primer on Radio Frequency Identification Technologies*. Morgan&Claypool, 2006.
- [18] Hiram Zúñiga Romero. Combinados. Integrando Raspberry Pi con Arduino. *Linux Magazine - Especial Hardware*. Abril, 2013.

APÉNDICE A

Código inicio del sistema

A.1 Script de inicio

Este script será ejecutado por el computador central en el momento en el que se inicie el sistema, y será el encargado de mostrar una interfaz de consola mediante la cual el usuario podrá identificarse y acceder al sistema de emulación arcade.

```
1
2 #!/usr/bin/python
3
4 # _____
5 # AUTHENTICATION SYSTEM
6 # _____
7 #
8 # _____
9 # Este codigo establece comunicacion serial a traves del cable
10 # usb con el Arduino, espera a recibir un identificador del
11 # lector, y una vez obtenido, verifica en la base de datos si
12 # el id esta registrado o no, comunicandose posteriormente
13 # al Arduino. Ademas, implementa una interfaz de consola para
14 # ver la interaccion del programa.
15 # _____
16
17 # Librerias importadas
18 import sqlite3 as lite
19 import sys
20 import serial
21 import time
22 import os
23 from termcolor import colored
24 # _____
25
26 # Variables
27 con = None
28 path = "/usr/local/bin/indiecity/InstalledApps/mame4all_pi/Full"
```

```

29 controls = "/home/pi/Desktop/Adafruit-Retrogame-master/"
30 #-----
31
32 # Funcion para acceder a un directorio pasado como parametro
33 def openDir(path):
34     os.chdir(path)
35 #-----
36
37 # Funcion para ejecutar mame
38 def execMame():
39     os.system("./mame")
40 #-----
41
42 # Funcion para ejecutar mame
43 def execControls():
44     os.system("./retrogame &")
45 #-----
46
47 #Funcion para imprimir la cabecera
48 def header():
49     #Limpiamos la pantalla
50     os.system('clear')
51     header = '''\t=====
52     ====='''
53     title1 = '''\t=====\t\t'''
54     title2 = '''Arcade Machine Emulator – Authentication System'''
55     title3 = '''\t\t====='''
56     print '\n'
57     print colored(header, 'green')
58     print colored('\t|                               |', 'blue', attrs=['bold'])
59     print colored(title1, 'green')+colored(title2, 'green', attrs=['bold', 'underline'])+colored(title3, 'green')
60     print colored('\t|                               |', 'blue', attrs=['bold'])
61     print colored(header, 'green')
62     print '\n'
63 #-----
64
65 #Funcion para imprimir el modo de uso
66 def info():
67     print colored('\t|', 'blue', attrs=['bold'])+colored(''''
68     |-----|''' , 'green')+colored('|', 'blue',
69     attrs=['bold'])
70     print colored('\t|\t\t', 'blue', attrs=['bold'])+colored('Para
71     autenticarse pase su identificador por el lector.\t\t', 'green')+
72     colored('|', 'blue', attrs=['bold'])
73     print colored('\t|', 'blue', attrs=['bold'])+colored(''''
74     |-----|''' , 'green')+colored('|', 'blue', attrs
75     =['bold'])
76     print '\n'
77 #-----
78
79 #Funcion para mostrar la lectura y comprobacion del identificador

```

```

76 def rfid(value):
77     print colored('\t|-----|', 'yellow')
78     print colored('\t|                                     |', 'yellow')
79     print colored('\t|             ESPERANDO UN          |', 'yellow')
80     print colored('\t|             IDENTIFICADOR ...         |', 'yellow')
81     print colored('\t|-----|', 'yellow')
82     print colored('\t|                                     |', 'yellow')
83     #Si el parametro pasado = 1, usuario valido. Color azul.
84     if value == 1:
85         print colored('\t|                                     |', 'yellow') + colored('- ACCESO
            AUTORIZADO -', 'white', 'on_blue') + colored(' |', 'yellow'
            )
86     #Si el parametro pasado = 0, usuario no valido. Color rojo.
87     elif value == 0:
88         print colored('\t|                                     |', 'yellow') + colored('- ACCESO DENEGADO
            -', 'white', 'on_red') + colored(' |', 'yellow')
89     #Valor por defecto. Color negro.
90     else:
91         print colored('\t|                                     |', 'yellow') + colored('
            -----', 'white', 'on_grey') + colored(' |
            ', 'yellow')
92     print colored('\t|-----|', 'yellow')
93     print '\n'
94     #-----
95
96     #Funcion para imprimir la informacion del usuario
97     def user(usuario, cargo, acceso):
98         print colored('\t|', 'blue', attrs=['bold']) + colored(' '\t
            |
            -----\t ', 'green') + colored(' |', 'blue', attrs=['bold'
            ])
99         print colored('\t|', 'blue', attrs=['bold']) + colored('\t|
            -----\t ', 'green') + colored(' |', 'blue', attrs=['bold'
            ])
100        print colored('\t\t          USUARIO: ', 'green') + colored(usuario.
            upper(), 'white')
101        print colored('\t\t          CARGO: ', 'green') + colored(cargo.
            upper(), 'white')
102        print colored('\t\t          ACCESO: ', 'green') + colored(acceso.
            upper(), 'white')
103        print colored('\t|', 'blue', attrs=['bold']) + colored('\t|
            -----\t ', 'green') + colored(' |', 'blue', attrs=['bold'
            ])
104        print colored('\t|', 'blue', attrs=['bold']) + colored(' '\t
            |
            -----\t ', 'green') + colored(' |', 'blue', attrs=['bold'
            ])
105        print '\n'
106        #-----
107
108     #Funcion para iniciar el emulador
109     def emulationStart(value):
110         if value == 1:
111             if value == 1:
112

```



```
160     res = res[0]
161
162     # Acciones dependiendo del resultado de la consulta
163     if (res == 0):
164         # Tarjeta degenegada
165         # Comunicamos al arduino que el identificador NO esta
166             autorizado
167         arduino.write('N')
168         # Cerramos la base de datos
169         con.close()
170         # Mostramos el acceso denegado por la interfaz
171         header()
172         info()
173         rfid(0)
174         user('no registrado', 'desconocido', 'denegado')
175         emulationStart(0)
176         # Reiniciamos la gui
177         time.sleep(3)
178         header()
179         info()
180         rfid(-1)
181         user('_____','_____','')
182             _____')
183         emulationStart(0)
184
185     if (res == 1):
186         # Tarjeta aceptada
187         # Recuperamos los datos de usuario
188         consultaUsr = "SELECT user FROM identifications WHERE id = '%
189             s';" % aux
190         cur.execute(consultaUsr)
191         usuario = cur.fetchone()
192         usuario = usuario[0]
193         consultaCargo = "SELECT cargo FROM identifications WHERE id =
194             '%s';" % aux
195         cur.execute(consultaCargo)
196         cargo = cur.fetchone()
197         cargo = cargo[0]
198         # Comunicamos al arduino que el identificador SI esta
199             autorizado
200         arduino.write('S')
201         # Cerramos la base de datos
202         con.close()
203         # Mostramos el acceso aceptado por la interfaz
204         header()
205         info()
206         rfid(1)
207         user(usuario, cargo, 'aceptado')
208         emulationStart(1)
209         # Activamos los controles
210         opendir(controls)
211         execControls()
212         # Iniciamos el emulador
213         opendir(path)
```

```
209     execMame()
210     # Esperamos hasta que se cierre el emulador
211     # Reiniciamos la gui
212     time.sleep(3)
213     header()
214     info()
215     rfid(-1)
216     user('_____' , '_____' ,
217         '_____')
218     emulationStart(0)
219 # En caso de fallo imprimir el error y salir del programa
220 except lite.Error, e:
221     print "Error: %" %e.args[0]
222     sys.exit(1)
223
224
225 # _____
226 # _____
```

APÉNDICE B

Código gestión de usuarios

B.1 Obtención de identificadores

Mediante este código de Arduino, somos capaces de leer los identificadores de las tarjetas y *tags* de radiofrecuencia que identifican a los usuarios autorizados a acceder a nuestro sistema arcade.

```
1
2 //-----//
3 //----- RFID READER -----//
4 //-----//
5
6 // Este codigo establece una comunicacion serial a 19200 bps entre
7 // el Arduino y el modulo lector de radiofrecuencia , el cual
8 // detecta los tags RFID ceranos y lee su identificador .
9 // Una vez detectado un transpondedor , el piezo emite un sonido ,
10 // y se muestra el identificador por pantalla .
11
12 byte dato[5]; //Array donde almacenaremos el ID de la tarjeta leida
13 int val = 0; //Buffer para los datos leidos por el puerto serie
14 int LED_OK = 9; //Pin donde conectamos el led verde
15 int LED_NO = 10; //Pin donde conectamos el led rojo
16 int speaker_pin = 11; //Pin donde conectamos el piezo
17
18 void setup ()
19 {
20 // Iniciamos el puerto serie 3 (rfid) a 19200 bps
21 Serial3.begin(19200);
22 // Iniciamos el puerto serie 0 (USB) a 9600 bps
23 Serial.begin(9600);
24 //Establecemos los pines de los LEDs y del piezo en modo salida
25 pinMode(LED_OK, OUTPUT);
26 pinMode(LED_NO, OUTPUT);
27 pinMode(speaker_pin, OUTPUT);
28 delay(300);
29
```

```
30 // Configuramos el modo lectura automatica – Modo de
31 // decodificacion EM4102 – Sin contraseña
32 // Comando segun Datasheet: FF 01 09 87 01 03 02 00 10 20 30 40 37
33 Serial3.write(0xFF); // Header
34 Serial3.write(0x01); // Reservado
35 Serial3.write(0x09); // Longitud = Comando(1) + Datos(8)
36 Serial3.write(0x87); // Comando a enviar (0x87 selecciona modo lectura
    automatica)
37 Serial3.write(0x01); // Dato 1: Activamos lectura automatica
38 Serial3.write(0x03); // Dato 2: Mode – Parity decoded – Manchester RF
    /64
39 Serial3.write(0x02); // Dato 3: Numero total de bloques a leer (2)
40 Serial3.write((byte)0x00); // Dato 4: Sin contraseña especificada
41 Serial3.write(0x10); // Dato 5: contraseña byte 1
42 Serial3.write(0x20); // Dato 6: contraseña byte 2
43 Serial3.write(0x30); // Dato 7: contraseña byte 3
44 Serial3.write(0x40); // Dato 8: contraseña byte 4
45 Serial3.write(0x37); // Checksum (Suma de verificacion)
46
47 delay(300);
48 while(Serial3.available() > 0)
49 {
50 Serial3.read();
51 }
52 Serial.println();
53
54 // Imprimimos por pantalla un mensaje para indicar el inicio del
    sistema
55 Serial.println("El modulo RFID esta en modo de lectura automatica –
    Esperando etiqueta...");
56
57 } // Fin setup()
58
59 void loop()
60 {
61 // Capturamos lo que hay en el puerto serie en la variable val
62 val = Serial3.read();
63
64 // Si no es 0xFF, volvemos a leer, esto nos indica el inicio de los
    datos procedentes de la etiqueta
65 while (val != 0xff)
66 {
67 val = Serial3.read();
68 delay(1000);
69 }
70
71 // Desechamos los tres primeros bytes recibidos para llegar al byte que
    nos interesa, la ID del TAG
72 Serial3.read(); // Reservado
73 Serial3.read(); // Longitud
74 Serial3.read(); // Comando (indica los datos de la etiqueta)
75 dato[0] = Serial3.read(); // Leemos el dato 1
76 dato[1] = Serial3.read(); // Leemos el dato 2
77 dato[2] = Serial3.read(); // Leemos el dato 3
```

```

78 dato[3] = Serial3.read();// Leemos el dato 4
79 dato[4] = Serial3.read();// Leemos el dato 5
80
81 // Identificamos la etiqueta RFID y la imprimimos por pantalla
82 Serial.print("Etiqueta detectada - ID: ");
83 for (int i=0; i<5; i++)
84 {
85 if (dato[i] < 16) Serial.print("0");
86 Serial.print(dato[i], HEX);
87 }
88
89 Serial.println();
90 //Emitimos un sonido mediante el piezo
91 //tone(Pin,Frecuencia,Tiempo);
92 tone(speaker_pin,800,250);
93 delay(300);
94
95 } // Fin loop()

```

B.2 Edición de objetos en la base de datos

Este script nos permite realizar funciones básicas con la base de datos, tales como listar los elementos almacenados, y añadir o eliminar ítems de la misma.

```

1
2 #!/usr/bin/python
3
4 # -----
5 # Este código implementa una serie de funciones para facilitar
6 # las tareas de insertar, borrar, y mostrar elementos de la
7 # tabla identifications de nuestra base de datos, así como
8 # algunos ejemplos de uso de dichas funciones.
9 # -----
10
11 # Import modules
12 import sqlite3 as lite
13 import sys
14
15 # Variable de la conexión
16 con = None
17
18 # Función para imprimir la BD por pantalla
19 def showTable(cur):
20     cur.execute("select * from identifications;")
21     print "\n"
22     print "%-15s %-15s %-15s %-30s %-30s" % ("SerialNumber", "TagType", "ID", "
23         User", "Cargo")
24     print "%-15s %-15s %-15s %-30s %-30s" % ("_____", "_____", "
25         _____", "_____", "_____")
26     while True:

```

```

25     row = cur.fetchone()
26     if row == None:
27         break
28     print "%-15s %-15s %-15s %-30s %-30s" %(row[0], row[1], row[2], row
29         [3], row[4])
30     print "\n"
31     # -----
32     # Funcion para insertar un elemento en la BD
33     def insertItem(cur, serial, trans, ident, usr, cargo):
34         cur.execute("insert into identifications values ('%s','%s','%s','%s
35            ','%s');" %(serial, trans, ident, usr, cargo))
36         print "\n"
37         print "Tag with serial ",serial, " saved successfully\n"
38         showTable(cur)
39         # -----
40     # Funcion para eliminar un elemento en la BD
41     def dropItem(cur, serial):
42         cur.execute("delete from identifications where serialNumber = '%s ';"
43             "% serial")
44         print "\n"
45         print "Tag with serial ",serial, " deleted successfully\n"
46         showTable(cur)
47         # -----
48
49     try:
50
51         # Abrimos la BD
52         con = lite.connect('arcadeDB.db')
53         print "DataBase opened OK"
54         cur = con.cursor()
55
56
57         # Ejemplos de uso
58
59         showTable(cur)
60         #insertItem(cur,"0002158299","tagRojo","100020EEDB","alvaro roig
61             coves","mi padre")
62         #insertItem(cur,"0001994104","tagAzul","0E001E6D78","xavier molero
63             prieto","director del museo")
64         #insertItem(cur,"0003199987","tagAmarillo","0F0030D3F3","invitado","
65             visitante")
66         #dropItem(cur,"0002158299")
67         #dropItem(cur,"0001994104")
68         #dropItem(cur,"0003199987")
69
70         # -----
71
72         # Hacemos commit en la BD
73         con.commit()

```

```

73 # Si hay algun error hacemos rollback en la BD e imprimimos el mensaje
    de error
74 except lite.Error, e:
75     if con:
76         con.rollback()
77     print "Error: %" %e.args[0]
78     sys.exit(1)
79
80 # Cerramos la BD
81 finally:
82     if con:
83         con.close()
84     print "DataBase closed OK"

```

B.3 Comunicación con la base de datos

Este código de Arduino obtiene el identificador de una tarjeta o *tag* de radiofrecuencia, y lo envía al computador central para ser validado en la base de datos.

```

1
2 //-----//
3 //      RFID AUTH      ---//
4 //-----//
5
6 // Este codigo establece una comunicacion serial a 19200 bps entre
7 // el Arduino y el modulo lector de radiofrecuencia, el cual
8 // detecta los tags RFID cercanos y lee su identificador.
9 // Ademas, se establece otra comunicacion serial a 9600 bps a
10 // traves del USB para comunicarse con la Raspberry Pi.
11 // Una vez detectado un transpondedor, el piezo emite un sonido,
12 // y se envia el id del tag a traves del USB para ser verificado
13 // en la base de datos.
14
15
16 byte dato[5]; // Variable donde almacenaremos el ID de la tarjeta
    leida
17 int val = 0; // Buffer para los datos leidos por el puerto serie
18 int LED_OK = 9; // Pin del led verde
19 int LED_NO = 10; // Pin del led rojo
20 int speaker_pin = 11; // Pin del piezo
21
22
23 void setup()
24 {
25 // Iniciamos el puerto serie usb (pin0-rx/pin1-tx) a 9600 bps
26 Serial.begin(9600);
27 // Iniciamos el puerto serie rfid (pin15-rx/pin14-tx) a 19200 bps
28 Serial3.begin(19200);
29 // Establecemos los LEDs verde y rojo y el piezo como salidas
30 // digitales
31 pinMode(LED_OK, OUTPUT);

```

```

32 pinMode(LED_NO, OUTPUT);
33 pinMode(speaker_pin, OUTPUT);
34 delay(300);
35
36 // Configuramos el modo lectura automatica del lector rfid- Modo
37 // de decodificacion EM4102 - Sin contraseña
38 // Comando segun Datasheet: FF 01 09 87 01 03 02 00 10 20 30 40 37
39 Serial3.write(0xFF); //Header
40 Serial3.write(0x01); //Reservado
41 Serial3.write(0x09); //Longitud = Comando(1) + Datos(8)
42 Serial3.write(0x87); //Comando a enviar (0x87 selecciona modo
43 // lectura automatica)
44 Serial3.write(0x01); //Dato 1: Activamos lectura automatica
45 Serial3.write(0x03); //Dato 2: Mode - Parity decoded - Manchester
46 // RF/64
47 Serial3.write(0x02); //Dato 3: Numero total de bloques a leer (2)
48 Serial3.write((byte)0x00); //Dato 4: Sin contraseña especificada
49 Serial3.write(0x10); //Dato 5: contraseña byte 1
50 Serial3.write(0x20); //Dato 6: contraseña byte 2
51 Serial3.write(0x30); //Dato 7: contraseña byte 3
52 Serial3.write(0x40); //Dato 8: contraseña byte 4
53 Serial3.write(0x37); //Checksum (Suma de verificacion)
54
55 delay(300);
56 while (Serial3.available() > 0)
57 {
58   Serial3.read();
59 }
60 Serial3.println();
61
62 } //Fin setup()
63
64 void loop()
65 {
66   // Capturamos lo que hay en el puerto serie rfid en la variable val
67   val = Serial3.read();
68
69   // Si no es 0xFF, volvemos a leer, esto nos indica el inicio de los
70   // datos procedentes de la etiqueta
71   while (val != 0xff)
72   {
73     val = Serial3.read();
74     delay(300);
75   }
76
77   // Desechamos los tres primeros byte recibidos para llegar al byte que
78   // nos interesa, la ID del TAG
79   Serial3.read(); // Reservado
80   Serial3.read(); // Longitud
81   Serial3.read(); // Comando (indica los datos de la
82   // etiqueta)
83   dato[0] = Serial3.read(); // Leemos el dato 1
84   dato[1] = Serial3.read(); // Leemos el dato 2
85   dato[2] = Serial3.read(); // Leemos el dato 3

```



```
83 dato[3] = Serial3.read(); // Leemos el dato 4
84 dato[4] = Serial3.read(); // Leemos el dato 5
85
86 // Emitimos un sonido para confirmar la lectura
87 tone(speaker_pin,800,250); //pin,frec,dur
88 delay(300);
89
90 // Transferimos el id mediante el puerto serie usb
91 Serial.println();
92 for (int i=0; i<5; i++)
93 {
94   if (dato[i] < 16)
95   {
96     Serial.print("0");
97   }
98   Serial.print(dato[i], HEX);
99 }
100 Serial.println();
101 delay(300);
102
103 // Leemos la respuesta emitida por la RaspberryPi
104 char recived;
105 recived = Serial.read();
106
107 // Dependiendo de la respuesta de la BD encendemos un led u otro
108 if(recived == 'N')
109 {
110   digitalWrite(LED_NO, HIGH);
111   delay(500);
112   digitalWrite(LED_NO, LOW);
113   delay(500);
114   digitalWrite(LED_NO, HIGH);
115   delay(500);
116   digitalWrite(LED_NO, LOW);
117 }
118 else if(recived == 'S')
119 {
120   digitalWrite(LED_OK, HIGH);
121   delay(500);
122   digitalWrite(LED_OK, LOW);
123   delay(500);
124   digitalWrite(LED_OK, HIGH);
125   delay(500);
126   digitalWrite(LED_OK, LOW);
127 }
128
129 }//Fin loop()
```

B.4 Comunicación con Arduino

Este script, que es ejecutado en el computador central, se comunica con el Arduino y verifica los identificadores de los usuarios en la base de datos, además, envía al Arduino la respuesta de si el identificador es válido a no.

```
1
2 #!/usr/bin/python
3
4 # -----
5 # Este codigo establece comunicacion serial a traves del cable
6 # usb con el Arduino, espera a recibir un identificador del
7 # lector , y una vez obtenido , verifica en la base de datos si
8 # el id esta registrado o no, comunicandose posteriormente
9 # al Arduino.
10 # -----
11
12 # Import modules
13 import sqlite3 as lite
14 import sys
15 import serial
16 import time
17
18 # Variable de la conexion
19 con = None
20
21 try:
22
23     # Abrimos comunicacion serial con el arduino
24     arduino = serial.Serial('/dev/ttyACM0', 9600)
25     time.sleep(0.5)
26     print "\nComunicacion serial establecida."
27
28     # Variable donde guardar el id recibido
29     ident = ""
30
31     with arduino:
32         # Esperamos hasta recibir un identificador
33         print "El modulo RFID esta en modo de lectura automatica -
34             Esperando etiqueta...\n"
35         for line in arduino:
36             if (len(line)==12):
37                 print "Etiqueta detectada - ID:"
38                 ident = line.strip("\n")
39                 print ident
40                 print "Verificando en BD..."
41                 # Abrimos la BD
42                 con = lite.connect('databaseArcade.db')
43                 print "DataBase opened OK"
44                 cur = con.cursor()
45                 # Realizamos la consulta
46                 print "Requesting..."
```

```
46     # Debido a que el identificador obtenido tiene un retorno de
47     # carro al final,
48     # creamos una variable auxiliar, y volcamos el contenido del
49     # identificador
50     # (menos el retorno de carro) en dicha variable.
51     aux = ""
52     for i in range(0, len(ident)-1):
53         aux += ident[i]
54     consulta = "SELECT count(*) FROM identifications WHERE id = '%s
55     ';' " % aux
56     cur.execute(consulta)
57     # Recogemos el resultado de la consulta
58     res = cur.fetchone()
59     # Como el resultado es una tupla, cogemos el primer elemento
60     res = res[0]
61     # Acciones dependiendo del resultado de la consulta
62     if (res == 0):
63         print " -> Tarjeta degenegada"
64         # Comunicamos al arduino que el identificador NO esta
65         # autorizado
66         arduino.write('N')
67     if (res == 1):
68         print " -> Tarjeta aceptada"
69         # Comunicamos al arduino que el identificador SI esta
70         # autorizado
71         arduino.write('S')
72     # Cerramos la base de datos
73     con.close()
74     print "DataBase closed OK\n"
75
76 # En caso de fallo imprimir el error y salir del programa
77 except lite.Error, e:
78     print "Error: %" % e.args[0]
79     sys.exit(1)
```

APÉNDICE C

Código del portal de administración telemática

C.1 Página de acceso

El siguiente código pertenece a la página de acceso al portal de administración telemática de nuestra máquina arcade, desde la cual el usuario puede introducir su nombre de usuario y contraseña con el fin de verificar sus credenciales en la base de datos del portal de administración.

```
1 <?php
2 session_start();
3 include_once "conexion.php";
4
5
6 function verificar_login($user,$password,&$result) {
7     $sql = "SELECT * FROM 'usuarios' WHERE user='$user' AND pass='
8         $password'";
9     $rec = mysql_query($sql);
10    $count = 0;
11
12    while($row = mysql_fetch_object($rec))
13    {
14        $count++;
15        $result = $row;
16    }
17    if($count == 1)
18    {
19        return 1;
20    }
21
22    else
23    {
24        return 0;
25    }
26 }
```

```
25 }
26
27 $_SESSION[ 'firmado' ] = 0;
28 ?>
29
30 <html>
31   <head>
32     <meta charset="utf-8">
33     <meta http-equiv="X-UA-Compatible" content="IE=edge">
34     <meta name="viewport" content="width=device-width, initial-scale=1"
35       >
36     <meta name="description" content="">
37     <meta name="author" content="">
38
39     <title>Arcade Machine Emulator – Gesti&oacute; remota</title>
40
41     <!-- Bootstrap Core CSS -->
42     <link href="bower_components/bootstrap/dist/css/bootstrap.min.css"
43       rel="stylesheet">
44
45     <!-- MetisMenu CSS -->
46     <link href="bower_components/metisMenu/dist/metisMenu.min.css" rel=
47       "stylesheet">
48
49     <!-- Custom CSS -->
50     <link href="dist/css/sb-admin-2.css" rel="stylesheet">
51
52     <!-- Custom Fonts -->
53     <link href="bower_components/font-awesome/css/font-awesome.min.css"
54       rel="stylesheet" type="text/css">
55
56     <!-- HTML5 Shim and Respond.js IE8 support of HTML5 elements and
57       media queries -->
58     <!-- WARNING: Respond.js doesnt work if you view the page via file :
59       // -->
60     <!--[if lt IE 9]>
61     <script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.
62       js"></script>
63     <script src="https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.
64       min.js"></script>
65     <![endif]-->
66
67     <link rel="shortcut icon" href="images/museo.ico"/>
68   </head>
69
70   <body>
71     <br><br><br>
72     <div class="container">
73
74       <div class="col-lg-12" align="center">
75
76         <div class="panel panel-default" style="box-shadow: 10px 10px
77           10px #aaa;">
78         <div class="panel-heading">
```

```

70     <div class="row">
71     <div class="col-md-4" align="center">
72     <a href="../index.html"></a>
73     </div>
74     <div class="col-md-6" align="left">
75     <p><strong><h2>Arcade Machine Emulator</h2></strong><h3>
       Sistema de gesti&oacuten remota</h3></p>
76     </div>
77     <div class="col-md-4"></div>
78     </div>
79 </div>
80 <!-- /.panel-heading -->
81
82 <div class="panel-body">
83     <div id="morris-line-chart">
84     <hr>
85     <div class="col-lg-6" align="center" style="margin-top:30px;"
86     >
87     <p>Los datos de usuario para acceder a la secci&oacuten de
       gesti&oacuten remota ser&aacuten expedidos por el Museo
       de Inform&aacuten tica de la Universidad Polit&eaacuten
       cnica de Valencia, de forma personal, a los encargados de
       realizar dichos tr&aacuten mites.</p>
88     <hr>
89 <?php
90     if (!isset($_SESSION[ 'userid ']))
91     {
92         if (isset($_POST[ 'login ']))
93         {
94             if (verificar_login($_POST[ 'user ' ],$_POST[ 'pass ' ],$result) == 1)
95             {
96                 $_SESSION[ 'userid ' ] = $_POST[ 'user ' ];
97                 $_SESSION[ 'ses ' ] = False;
98                 $_SESSION[ 'firmado ' ] = 1;
99                 header( 'Location: accesoAdmin.php?action=home' );
100            }
101            else
102            {
103                echo ' <br><div align="center" style="color:red;"><strong>Sus
                  datos de acceso son incorrectos , int&eaacuten ntelo
                  nuevamente.</strong></div> ' ;
104                $_SESSION[ 'firmado ' ] = 0;
105            }
106        }
107    }
108 <?>
109
110     </div>
111     <div class="col-lg-6" align="center">
112     <div class="login-panel panel panel-default" style="margin-
       top:10px; margin-left:30px; margin-right:30px;">
113     <div class="panel-heading">
114     <h3 class="panel-title">Acceso Usuarios</h3>
115     </div>
116     <div class="panel-body">

```

```

114     <form action="" method="post" class="login">
115     <fieldset>
116     <div class="form-group">
117     <input class="form-control" placeholder="Usuario" name="user"
118         type="text" autofocus>
119     </div>
120     <div class="form-group">
121     <input class="form-control" placeholder="Contraseña" name="
122         pass" type="password">
123     </div>
124     <!-- Change this to a button or input when using this as a
125         form -->
126     <div>
127     <input name="login" type="submit" value="Entrar" class="btn
128         btn-lg btn-success btn-block">
129     </div>
130     </fieldset>
131     </form>
132     </div>
133     <!-- /.panel-body -->
134 </div>
135 <!-- /.panel -->
136 </div>
137 </div>
138
139 <!-- jQuery -->
140 <script src="bower_components/jquery/dist/jquery.min.js"></script
141     >
142
143 <!-- Bootstrap Core JavaScript -->
144 <script src="bower_components/bootstrap/dist/js/bootstrap.min.js"
145     ></script>
146
147 <!-- Metis Menu Plugin JavaScript -->
148 <script src="bower_components/metisMenu/dist/metisMenu.min.js"></
149     script>
150
151 <!-- Custom Theme JavaScript -->
152 <script src="dist/js/sb-admin-2.js"></script>
153
154 </body>
155 </html>
156
157 <?php
158     } else {
159         echo 'Usuario autenticado correctamente. ';
160         echo '<a href="logout.php">Cerrar Sesión.</a>';
161     }
162 ?>

```


C.2 Página de inicio

El siguiente código pertenece a la página de inicio del portal de administración telemática de nuestra máquina arcade, desde la cual el usuario puede realizar gestiones tales como la edición de identificadores de radiofrecuencia, la edición de las ROMs existentes en el sistema, o la generación de copias de seguridad de las bases de datos.

```
1 <?php
2
3
4 //Incluimos el fichero auxiliar para la conexion a la BD
5 include_once "conexion.php";
6 //Iniciamos la sesion de usuario
7 session_start();
8
9 //Control para asegurar que el usuario inicia sesion
10 $firmado = $_SESSION['firmado'];
11 if ($firmado==0){
12     header('Location: signin.php');
13 }
14
15 //Definicion de la BD Sqlite3
16 try {
17 $bd = new SQLite3('/home/pi/Desktop/arcadeDB.db');
18 } catch (Exception $e) {
19     echo 'Excepcion capturada: ', $e->getMessage(), "\n";
20 }
21
22 //Damos forma al contenido HHTML
23 echo '
24 <!DOCTYPE html>
25 <html lang="es">
26
27 <head>
28
29     <meta charset="utf-8">
30     <meta http-equiv="X-UA-Compatible" content="IE=edge">
31     <meta name="viewport" content="width=device-width, initial-scale
32     =1">
33     <meta name="description" content="">
34     <meta name="author" content="">
35
36     <title>Arcade Machine Emulator</title>
37
38     <!-- Bootstrap Core CSS -->
39     <link href="bower_components/bootstrap/dist/css/bootstrap.min.css"
40         rel="stylesheet">
41
42     <!-- MetisMenu CSS -->
43     <link href="bower_components/metisMenu/dist/metisMenu.min.css" rel
44         ="stylesheet">
```

```

42 <!-- Timeline CSS -->
43 <link href="dist/css/timeline.css" rel="stylesheet">
44
45 <!-- Custom CSS -->
46 <link href="dist/css/sb-admin-2.css" rel="stylesheet">
47
48 <!-- Morris Charts CSS -->
49 <link href="bower_components/morrisjs/morris.css" rel="stylesheet">
50
51 <!-- Custom Fonts -->
52 <link href="bower_components/font-awesome/css/font-awesome.min.css"
53       rel="stylesheet" type="text/css">
54
55 <!-- HTML5 Shim and Respond.js IE8 support of HTML5 elements and
56       media queries -->
57 <!-- WARNING: Respond.js doesnt work if you view the page via file
58       :// -->
59 <!--[if lt IE 9]>
60   <script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/
61       html5shiv.js"></script>
62   <script src="https://oss.maxcdn.com/libs/respond.js/1.4.2/
63       respond.min.js"></script>
64 <![endif]-->
65 <link rel="shortcut icon" href="images/museo.ico"/>
66 </head>
67 <body>
68   <div id="wrapper">
69     <!-- Navigation -->
70     <nav class="navbar navbar-default navbar-static-top" role="
71       navigation" style="margin-bottom: 0">
72       <div class="navbar-header" style="margin-left:10px;">
73         <button type="button" class="navbar-toggle" data-toggle
74           ="collapse" data-target=".navbar-collapse">
75           <span class="sr-only">Toggle navigation</span>
76           <span class="icon-bar"></span>
77           <span class="icon-bar"></span>
78           <span class="icon-bar"></span>
79         </button>
80         <a class="navbar-brand" href="accesoAdmin.php?action=
81           home"><strong>Arcade Machine Emulator – Sistema de
82           gesti&oacute;n remota</strong></a>
83       </div>
84     <!-- /.navbar-header -->
85
86     <ul class="nav navbar-top-links navbar-right" align="center
87       " style="margin-right:50px;">';
88
89     //Averiguamos e imprimimos los datos del usuario
90     $usr = $_SESSION['userid'];

```

```

86 $sql = "SELECT nombre, apellidos FROM 'usuarios' WHERE user='$usr'";
87 $query = mysql_query($sql);
88 $row = mysql_fetch_row($query);
89 $nombre = $row[0];
90 $apellidos = $row[1];
91
92 echo '<strong>Usuario: [ </strong>';
93 echo $apellidos; echo ", "; echo $nombre;
94 echo '<strong> ]</strong>';
95
96     echo '
97         <li class="dropdown">
98             <a class="dropdown-toggle" data-toggle="dropdown"
99                 href="#">
100                 <i class="fa fa-user fa-fw"></i> <i class="fa
101                 fa-caret-down"></i>
102             </a>
103             <ul class="dropdown-menu dropdown-user">
104                 <li><a href="accesoAdmin.php?action=home"><i
105                 class="fa fa-user fa-fw"></i> Perfil de
106                 usuario</a>
107                 </li>
108                 <li><a href="#"><i class="fa fa-gear fa-fw"></i>
109                 > Settings</a>
110                 </li>
111                 <li class="divider"></li>
112                 <li><a href="logout.php"><i class="fa fa-sign-
113                 out fa-fw"></i> Cerrar sesi&oacute;n</a>
114                 </li>
115             </ul>
116             <!-- /.dropdown-user -->
117         </li>
118     <!-- /.dropdown -->
119 </ul>
120 <!-- /.navbar-top-links -->
121
122     <div class="navbar-default sidebar" role="navigation">
123         <div class="sidebar-nav navbar-collapse">
124             <ul class="nav" id="side-menu">
125                 <li align="center">
126                     <a href="accesoAdmin.php?action=home"></a>
130                 </li>
131                 <li>
132                     <a href="#"><i class="fa fa-files-o fa-fw
133                     "></i> Identificadores<span class="fa
134                     arrow"></span></a>
135                     <ul class="nav nav-second-level">
136                         <li>';
137                         $_SESSION['ses'] = True;

```

```

127         echo ' <a href="accesoAdmin.php?
           action=listaIdentificador">
           Listado de Identificadores </a>
128     </li>
129     <li>';
130         $_SESSION['ses'] = True;
131         echo ' <a href="accesoAdmin.php?
           action=altaIdentificador">Alta
           Nuevo Identificador </a>
132     </li>
133     <li>';
134         $_SESSION['ses'] = True;
135         echo ' <a href="accesoAdmin.php?
           action=modificarIdentificador
           "></i>Modificar Identificador </
           a>
136     </li>
137     <li>';
138         $_SESSION['ses'] = True;
139         echo ' <a href="accesoAdmin.php?
           action=eliminarIdentificador
           "></i>Eliminar Identificador </a
           >
140     </li>
141 </ul>
142 <!-- /.nav-second-level -->
143 </li>
144 <li>
145     <a href="#"><i class="fa fa-files-o fa-fw
           "></i> Gesti&oacute;n de ROMs<span
           class="fa arrow"></span></a>
146     <ul class="nav nav-second-level">
147         <li>';
148             $_SESSION['ses'] = True;
149             echo ' <a href="accesoAdmin.php?
           action=listaRoms">Listado de
           ROMs</a>
150         </li>
151         <li>';
152             $_SESSION['ses'] = True;
153             echo ' <a href="accesoAdmin.php?
           action=altaRom">Alta ROM</a>
154         </li>
155         <li>';
156             $_SESSION['ses'] = True;
157             echo ' <a href="accesoAdmin.php?
           action=eliminarRom">Eliminar
           ROM</a>
158         </li>
159     </ul>
160 <!-- /.nav-second-level -->
161 </li>
162 </ul>
163 </div>

```

```

164         <!-- /.sidebar-collapse -->
165     </div>
166     <!-- /.navbar-static-side -->
167 </nav>
168
169 <!-- Page Content -->
170 <div id="page-wrapper">
171     <div class="container-fluid">
172         <div class="row">
173             <div class="col-lg-12">';
174
175 $ses = $_SESSION[ 'ses '];
176
177 if ($ses!=False){
178     $action=$_GET[ 'action '];
179
180 //Diferenciamos el contenido a mostrar dependiendo de la seleccion
181 //del usuario
182 if($action == 'home'){
183
184     $sql = "SELECT user, nombre, apellidos FROM 'usuarios' WHERE
185           user='$usr' ";
186     $query = mysql_query($sql);
187     $row = mysql_fetch_row($query);
188     $user = $row[0];
189     $nombre = $row[1];
190     $apellidos = $row[2];
191
192     echo '<h1 class="page-header">Perfil de Usuario</h1>';
193     echo ' <br><br>
194         <div class="col-lg-8">
195             <div class="panel panel-default">
196                 <div class="panel-heading">
197                     Datos de usuario
198                 </div>
199                 <!-- /.panel-heading -->
200                 <div class="panel-body">
201                     <div class="table-responsive">
202                         <table class="table table-striped">
203                             <tbody>
204                                 <tr>
205                                     <td><strong>Usuario:</strong></td>
206                                     <td>';echo $user;
207                                     echo '
208                                     </td>
209                                 </tr>
210                                 <tr>
211                                     <td><strong>Nombre:</strong></td>
212                                     <td>';echo $nombre;
213                                     echo '
214                                     </td>

```



```
262         echo '<td class="center">'. $row[ 'id' ]. '  
263             </td>';  
264         echo '<td class="center">'. $row[ 'user '  
265             ]. '</td>';  
266         echo '<td class="center">'. $row[ 'cargo '  
267             ]. '</td>';  
268         echo "</tr>";  
269     }  
270     echo '</tbody>  
271     </table>  
272 </div>  
273 <!-- /.table-responsive -->  
274 </div>  
275 <!-- /.panel-body -->  
276 </div>  
277 <!-- /.panel -->  
278 <br>  
279 </div>  
280  
281 else if ($action == 'altaIdentificador') {  
282     echo '<h1 class="page-header">Alta Nuevo Identificador</h1>';  
283     echo ' <br><br>  
284         <div class="col-lg-10">';  
285  
286  
287     if (isset($_POST[ 'serialNumber' ]) && isset($_POST[ 'transpondedor '  
288         ]) && isset($_POST[ 'id' ])  
289         && isset($_POST[ 'user' ]) && isset($_POST[ 'cargo' ])) {  
290         $serialNumber = $_POST[ 'serialNumber' ]; $transpondedor =  
291             $_POST[ 'transpondedor' ]; $id = $_POST[ 'id' ];  
292         $user = $_POST[ 'user' ]; $cargo = $_POST[ 'cargo' ];  
293         try {  
294             $sql = "INSERT INTO identifications (serialNumber,  
295                 transpondedor, id, user, cargo)  
296                 VALUES ('$serialNumber', '$transpondedor', '$id', '  
297                     '$user', '$cargo')";  
298             $bd->exec($sql);  
299             echo '  
300             <div class="alert alert-success alert-dismissable"  
301                 align="center">  
302             <button type="button" class="close" data-dismiss="alert"  
303                 aria-hidden="true">&times;</button>  
304             Identificador dado de alta correctamente.  
305             </div>  
306             ';  
307         }  
308         catch (Exception $e) {  
309             echo '  
310             <div class="alert alert-danger alert-dismissable" align  
311                 ="center">
```

```
305 <button type="button" class="close" data-dismiss="alert
      " aria-hidden="true">&times;</button>
306 Error en la base de datos. Algun valor introducido en
      los datos del identificador no es correcto: '. $e->
      getMessage();
307 echo '</div>
308 ';
309 }
310 }
311 echo '
312 <div class="panel panel-default">
313 <div class="panel-heading">
314 Alta nuevo identificador
315 </div>
316 <div class="panel-body">
317 <div class="row">
318 <div class="col-lg-10" style="margin-
      left:40px;">
319 <form role="form" action="
      accesoAdmin.php?action=
      altaIdentificador" method="post
      ">
320 <p><h3><u>Datos del
      identificador </u></h3></p>
321 <br>
322 <div class="form-group">
323 <label>Número de
      Serie:</label>
324 <input class="form-control"
      type="text"
      placeholder="Introduce
      el número de
      serie del identificador"
      name="serialNumber"
      required>
325 </div>
326 <div class="form-group">
327 <label>Transpondedor:</
      label>
328 <input class="form-control"
      type="text"
      placeholder="Indica el
      tipo de transpondedor"
      name="transpondedor"
      required>
329 </div>
330 <div class="form-group">
331 <label>ID:</label>
332 <input class="form-control"
      type="text"
      placeholder="Introduce
      el identificador del
      transpondedor" name="id"
      required>
```



```

333         </div>
334         <div class="form-group">
335             <label>Usuario:</label>
336             <input class="form-control"
337                 type="text"
338                 placeholder="Introduce
339                 el nombre del usuario
340                 al que se le entrega el
341                 transpondedor" name="
342                 user" required>
343         </div>
344         <div class="form-group">
345             <label>Cargo:</label>
346             <input class="form-control"
347                 type="text"
348                 placeholder="Indica el
349                 cargo del usuario" name="
350                 cargo" required>
351         </div>
352         <br>
353         <hr>
354         <input type="submit" class="btn
355             btn-default" value="Alta
356             identificador">
357         <button type="reset" class="btn
358             btn-default">Borrar datos
359         </button>
360         <br><br>
361     </form>
362 </div>
363 <!-- /.col-lg-10 (nested) -->
364 <br><br>
365 </div>
366 <!-- /.row (nested) -->
367 </div>
368 <!-- /.panel-body -->
369 </div>
370 <!-- /.panel -->
371 <hr>
372 </div>
373 ';
374 }
375
376 else if($action == 'modificarIdentificador'){
377     echo '<h1 class="page-header">Modificar Identificador</h1>';
378
379     if(isset($_POST['selectedSerialNum'])){
380
381         if(isset($_POST['transpondedor'])&&isset($_POST['id'])
382             &&isset($_POST['user'])&&isset($_POST['cargo'])){
383
384             $serialNumber = $_POST['selectedSerialNum'];
385             $transpondedor = $_POST['transpondedor'];

```

```

371     $id = $_POST['id']; $user = $_POST['user']; $carga =
372         $_POST['carga'];
373
374     try{
375         $sql = "UPDATE identificaciones SET transpondedor='
376             $transpondedor', user='$user', carga='$carga'
377             WHERE serialNumber='$serialNumber'";
378         $bd->exec($sql);
379         echo '
380         <div class="alert alert-success alert-dismissable"
381             align="center">
382         <button type="button" class="close" data-dismiss="
383             alert" aria-hidden="true">&times;</button>
384         Identificador modificado correctamente.
385         </div>
386         '
387     }
388     catch(Exception $e){
389         echo '
390         <div class="alert alert-danger alert-dismissable"
391             align="center">
392         <button type="button" class="close" data-dismiss="
393             alert" aria-hidden="true">&times;</button>
394         Error en la base de datos. Algun valor introducido
395         no es correcto: '. $e->getMessage();
396         echo '</div>
397         '
398     }
399 }
400 else{
401     $serialNumber = $_POST['selectedSerialNum'];
402     $sql = "SELECT * FROM identificaciones WHERE
403         serialNumber='$serialNumber'";
404     $results = $bd->query($sql);
405     $row = $results->fetchArray();
406     echo '
407     <br><br>
408     <div class="col-lg-10">
409         <div class="panel panel-default">
410             <div class="panel-heading">
411                 Modificar datos de identificador

```

```
412         <label>Número de
413         Serie:</label>
414         <input class="form-control"
415         value="'. $serialNumber
416         .' " type="text" name="
417         serialNumber" disabled>
418         ' '
419     </div>
420     <input type="hidden" name="
421     selectedSerialNum" value="'.
422     $serialNumber. "'>
423     <div class="form-group">
424     <label>Transpondedor:</
425     label>
426     <input class="form-control"
427     value="'. $row [1]. "'
428     type="text" name="
429     transpondedor" required
430     ">' '
431     </div>
432     <div class="form-group">
433     <label>ID:</label>
434     <input class="form-control"
435     value="'. $row [2]. "'
436     type="text" name="id"
437     disabled>' '
438     </div>
439     <input type="hidden" name="id"
440     value="'. $row [2]. "'>
441     <div class="form-group">
442     <label>Usuario:</label>
443     <input class="form-control"
444     value="'. $row [3]. "'
445     type="text" name="user"
446     required">' '
447     </div>
448     <div class="form-group">
449     <label>Cargo:</label>
450     <input class="form-control"
451     value="'. $row [4]. "'
452     type="text" name="cargo"
453     required">' '
454     </div>
455     <br>
456     <hr>
457     <div align="right">
458     <input type="submit" class
459     ="btn btn-default"
460     value="Guardar
461     identificador">
462     </div>
463     <br>
464     </form>
465     </div>
```

```

441         </div>
442     </div>
443 </div>
444 <!-- /.panel-body -->
445 </div>
446 <!-- /.panel -->
447 </div>
448     ';
449 }
450
451 }
452 else if (!isset($_POST[ 'selectedSerialNum ' ]) || $_POST[ '
453 selectedSerialNum ' ]==" default" ) {
454     echo ' <br><br>
455     <div class="col-lg-10">
456         <div class="panel panel-default">
457             <div class="panel-heading">
458                 Identificadores del sistema
459             </div>
460             <!-- /.panel-heading -->
461             <div class="panel-body">
462                 <br>
463                 <div class="form-group">
464                     <label>Selecciona un identificador:</
465                     label>
466                     <form action="accesoAdmin.php?action=
467                         modificarIdentificador" method="
468                         post">
469                         <select class="form-control" name="
470                             selectedSerialNum">
471                             <option value="default" selected="
472                                 selected"></option>';
473                             $results = $bd->query('SELECT
474                                 serialNumber FROM identifications')
475                             ;
476                             while ($row = $results->fetchArray()) {
477                                 echo ' <option value="' . $row[0] . '">
478                                     ' . $row[0] . ' </option></a>';
479                             }
480                             echo '
481                             </select>
482                             <div align="right">
483                                 <br>
484                                 <button type="submit" class="btn
485                                     btn-default">Modificar
486                                     identificador </button>
487
488                             </div>
489                             </form>
490                         </div>
491                     </div>
492                 </div>
493             <!-- /.table-responsive -->
494         </div>
495     <!-- /.panel-body -->
496 </div>

```



```

527     }
528     else {
529         $serialNumber = $_POST[ 'selectedSerialNum ' ];
530         try {
531             $sql = "DELETE FROM identifications WHERE serialNumber
                    = '$serialNumber ' ";
532             $bd->exec($sql);
533             echo '
534             <div class="alert alert-success alert-dismissable"
                    align="center">
535             <button type="button" class="close" data-dismiss="alert
                    " aria-hidden="true">&times;</button>
536             Identificador eliminado correctamente.
537             </div>
538             ' ;
539         }
540         catch (Exception $e) {
541             echo '
542             <div class="alert alert-danger alert-dismissable" align
                    ="center">
543             <button type="button" class="close" data-dismiss="alert
                    " aria-hidden="true">&times;</button>
544             Error en la base de datos. Algun valor introducido no
                    es correcto: '. $e->getMessage();
545             echo '</div>
546             ' ;
547         }
548     }
549 }
550
551 else if ($action == "listaRoms") {
552     echo '<h1 class="page-header">Listado de ROMs</h1>';
553     echo ' <br><br>
554         <div class="col-lg-8">
555             <div class="panel panel-default">
556                 <div class="panel-heading">
557                     ROMs del sistema
558                 </div>
559                 <!-- /.panel-heading -->
560                 <div class="panel-body">
561                     <br>
562                     <div class="form-group">';
563                     $directorio = opendir("/usr/local/bin/
                    indiecity/InstalledApps/mame4all_pi
                    /Full/roms/"); //ruta directorio
564                     while ($archivo = readdir($directorio))
565                         {
566                         //obtenemos un archivo y luego otro
567                         sucesivamente
568                         if (!is_dir($archivo)) {
569                             echo '
                    <li class="dropdown">
                    <a href="roms/' .
                    $archivo.'" target

```

```

570                                     = "_blank">'.
571                                     $archivo.'</a>
572                                     </li>
573                                     ';
574                                     }
575                                     }
576                                     echo '
577                                     </div>
578                                     </div>
579                                     <!-- /.table-responsive -->
580                                     </div>
581                                     <!-- /.panel-body -->
582                                     </div>
583                                     <!-- /.panel -->
584                                     </div>
585                                     ';
586     }
587     else if($action == "altaRom"){
588     echo '<h1 class="page-header">Alta ROMs</h1>';
589     echo ' <br><br>
590         <div class="col-lg-8">
591         <div class="panel panel-default">
592         <div class="panel-heading">
593         Alta una nueva ROM
594         </div>
595         <!-- /.panel-heading -->
596         <div class="panel-body">
597         <br>
598         </div>
599         </div>
600         <!-- /.panel-body -->
601         </div>
602         <!-- /.table-responsive -->
603         </div>
604         <!-- /.panel -->
605         </div>
606         </div>
607         </div>
608         </div>
609         </div>
610         </div>';
611     }
612     else{
613     // Primero creamos un ID de conexion a nuestro servidor
614     $cid = ftp_connect("localhost");
615     // Luego creamos un login al mismo con nuestro usuario y
616     // contraseña
617     $resultado = ftp_login($cid, "pi", "arco93upv");
618     // Comprobamos que se crea el Id de conexion y se puede
619     // hacer el login

```

```

618     if ((!$cid) || (!$resultado)) {
619         echo "Fallo en la conexion"; die;
620     } else {
621         echo " -> Conectado.";
622     }
623     // Cambiamos a modo pasivo, esto es importante porque, de
        // esta manera le decimos al servidor que seremos nosotros
        // quienes comenzaremos la transmision de datos.
624     ftp_pasv ($cid, true) ;
625     echo "<br> -> Cambio a modo pasivo<br />";
626     // Cogemos el nombre del archivo a transmitir.
627     $local = $_FILES["rom"]["name"];
628     // Este es el nombre temporal del archivo mientras dura la
        // transmision
629     $remoto = $_FILES["rom"]["tmp_name"];
630     // El tamaño del archivo
631     $tama = $_FILES["rom"]["size"];
632     echo "<br/> -> Fichero: $local<br />";
633     echo " -> Directorio temporal: $remoto<br />";
634     echo " -> Subiendo el archivo...<br />";
635     // Juntamos la ruta del servidor con el nombre real del
        // archivo
636     $ruta = "/usr/local/bin/indiecity/InstalledApps/mame4all_pi
        /Full/roms/" . $local;
637     echo " -> Ruta del fichero: " . $ruta;
638     // Verificamos si ya se subio el archivo temporal
639     if (is_uploaded_file($remoto)){
640         // copiamos el archivo temporal, del directorio de
        // temporales de nuestro servidor a la ruta que
        // creamos
641         copy($remoto, $ruta);
642         echo "<br> -> [Archivo subido correctamente]";
643     }
644     // Si no se pudo subir el temporal
645     else {
646         echo "<br> -> [No se pudo subir el archivo]";
647     }
648     //}
649     //cerramos la conexion FTP
650     ftp_close($cid);
651 }
652     echo '
653     </div>
654     <!-- /.table-responsive -->
655     </div>
656     <!-- /.panel-body -->
657     </div>
658     <!-- /.panel -->
659     </div>
660     ';
661 }
662
663 else if($action == 'eliminarRom'){
664     echo '<h1 class="page-header">Eliminar ROM</h1>';

```



```

665     if (!isset($_POST['selectedROM']) || $_POST['selectedROM'] == "
        default"){
666     echo ' <br><br>
667         <div class="col-lg-10">
668             <div class="panel panel-default">
669                 <div class="panel-heading">
670                     ROMs del sistema
671                 </div>
672                 <!-- /.panel-heading -->
673                 <div class="panel-body">
674                     <br>
675                     <div class="form-group">
676                         <label>Selecciona una ROM:</label>
677                         <form action="accesoAdmin.php?action=
678                             eliminarRom" method="post">
679                             <select class="form-control" name="
680                                 selectedROM">
681                                 <option value="default" selected="
682                                     selected"></option>';
683                                 $directorio = opendir("/usr/local/bin/
684                                     indiecity/InstalledApps/mame4all_pi
685                                     /Full/roms/"); //ruta directorio
686                                 while ($archivo = readdir($directorio))
687                                 {
688                                     //obtenemos un archivo y luego otro
689                                     sucesivamente
690                                     if (!is_dir($archivo)){
691                                         echo '<option value="'. $archivo
692                                             . '">' . $archivo . '</option
693                                             ></a>';
694                                     }
695                                 }
696                                 echo '
697                                 </select>
698                                 <div align="right">
699                                     <br>
700                                     <button type="submit" class="btn
701                                         btn-default">Eliminar ROM</
702                                         button>
703                                 </div>
704                                 </form>
705                                 </div>
706                                 </div>
707                                 <!-- /.table-responsive -->
708                             </div>
709                             <!-- /.panel-body -->
710                         </div>
711                         <!-- /.panel -->
712                     </div>
713                 ';
714             }
715         else {
716             $rom = $_POST['selectedROM'];
717             try {

```

```

707         unlink("/usr/local/bin/indiecity/InstalledApps/
708             mame4all_pi/Full/roms/" . $rom);
709         echo '
710         <div class="alert alert-success alert-dismissable"
711             align="center">
712         <button type="button" class="close" data-dismiss="alert
713             " aria-hidden="true">&times;</button>
714         ROM eliminada correctamente.
715         </div>
716         ' ;
717     }
718     catch(Exception $e){
719         echo '
720         <div class="alert alert-danger alert-dismissable" align
721             ="center">
722         <button type="button" class="close" data-dismiss="alert
723             " aria-hidden="true">&times;</button>
724         Error al eliminar la ROM: ' . $e->getMessage();
725         echo '</div>
726         ' ;
727     }
728 }
729 //Fin de las acciones de usuario
730 }
731
732     echo '
733         </div>
734         <!-- /.col-lg-12 -->
735     </div>
736     <!-- /.row -->
737 </div>
738 <!-- /.container-fluid -->
739 </div>
740 <!-- /#page-wrapper -->
741
742 <!-- jQuery -->
743 <script src="bower_components/jquery/dist/jquery.min.js"></script>
744
745 <!-- Bootstrap Core JavaScript -->
746 <script src="bower_components/bootstrap/dist/js/bootstrap.min.js
747     "></script>
748
749 <!-- Metis Menu Plugin JavaScript -->
750 <script src="bower_components/metisMenu/dist/metisMenu.min.js"></
751     script>
752
753 <!-- Custom Theme JavaScript -->
754 <script src="dist/js/sb-admin-2.js"></script>

```

```
754 </body>
755
756 </html>';
757
758 ?>
```

C.3 Conexión a la base de datos

Este código forma parte del fichero `conexion.php`, el cual se incluye tanto en la página de acceso, como en la página de inicio del portal, y es el encargo de gestionar la conexión con la base de datos en el momento en que el usuario inicia una sesión nueva en el portal.

```
1 <?php
2 //datos para la conexion a mysql
3 define( 'DB_SERVER', 'localhost' );
4 define( 'DB_NAME', 'arcade' );
5 define( 'DB_USER', 'pi' );
6 define( 'DB_PASS', 'WB5GSQ5bbBuZZ4KQ' );
7 $con = mysql_connect(DB_SERVER,DB_USER,DB_PASS);
8 mysql_select_db(DB_NAME,$con);
9 ?>
```

C.4 Cerrar sesión de usuario

Este código forma parte del fichero `logout.php`, el cual se incluye en la página de inicio del portal, y es el encargo de cerrar la sesión de trabajo del usuario de una forma segura en el momento en el que el usuario decide cerrar la sesión.

```
1 <?php
2 session_start();
3 session_destroy();
4
5 header( 'location: signin.php' );
6 ?>
```