UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Diseño de máquina arcade con servidor Web e identificación NFC

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: Alejandro Grau López

Tutor: Jorge González Mollá

Curso 2020-2021

Resum

Mitjançant aquest projecte es pretén abordar una implementació completa d'una màquina arcade amb una Raspberry Pi a més d'una senzilla base de dades per a emmagatzemar la informació relativa dels nostres jocs. Amb un mòdul RFID/NFC integrat en la màquina, un usuari es podrà identificar per a emmagatzemar les seues puntuacions personals que posteriorment podrà consultar mitjançant una aplicació web. Durant el transcurs d'aquest projecte s'abordarà el disseny extern de la màquina, així com la implementació interna. S'analitzaran les possibles implementacions, així com els diferents tipus de disseny que el sistema pot arribar a tindre.

Paraules clau: arcade, raspberrypi, python, retropie

Resumen

Mediante este proyecto se pretende abordar una implementación completa de una máquina arcade con una Raspberry Pi además de una sencilla base de datos para almacenar la información relativa de nuestros juegos. Con un módulo RFID/NFC integrado en la máquina un usuario se podrá identificar para almacenar sus puntuaciones personales que posteriormente podrá consultar mediante una aplicación web. Durante el transcurso de este proyecto se abordará el diseño externo de la máquina, así como la implementación interna. Se analizarán las posibles implementaciones, así como los distintos tipos de diseño que el sistema puede llegar a tener.

Palabras clave: arcade, raspberrypi, python, retropie

Abstract

This project aims to address a complete implementation of an arcade machine with a Raspberry Pi as well as a simple database to store the relative information of our games. With an RFID / NFC module integrated in the machine, a user will be able to identify themselves to store their personal scores that can later be consulted through a web application. During the course of this project the external design of the machine will be addressed, as well as the internal implementation. The possible implementations will be analyzed, as well as the different types of design that the system may have.

Key words: arcade, raspberrypi, python, retropie

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1
Introducción

En los posteriores capítulos se pretende diseñar e implementar una máquina arcade de juegos como las de los años 1980-1990 con herramientas actuales. Además de la funcionalidad principal de emular juegos de las principales máquinas arcade de la época, tendrá la funcionalidad de almacenar las puntuaciones de los jugadores en una base de datos. Posteriormente, se podrían consultar con una aplicación móvil o web para visualizar las estadísticas de cada jugador.

1.1 Motivación

La principal razón que me llevó a la realización de este trabajo fue la idea de poder tener una máquina recreativa como las que en los años 1980-2000 poblaban la mayor parte de los bares y salas recreativas del territorio español. Por aquella época me llamó la atención aquellos videojuegos con tal aire retro que empezaba a perderse con la entrada de nuevas consolas en los hogares.

Otra de las razones por las que me lancé a escribir este proyecto es el auge que están teniendo las computadoras de placa reducida con altas prestaciones y bajo consumo/-precio esta última década. Sistemas como Raspberry Pi o Arduino, son un claro ejemplo de que el *hardware* y su producción está cada vez más al alcance de todos, no solo por la capacidad de producción a un bajo coste, sino también por su licencia libre de uso. Gracias a esto, una amplia comunidad de desarrolladores ha podido poner al alcance grandes conocimientos de electrónica e informática. Por consiguiente, esto supone una oportunidad ideal para la creación de una máquina que no antes podría haberse llevado a cabo tan fácilmente.

La última razón es el papel que las nuevas tecnologías pueden proporcionar a un sistema como este. Gracias a este abaratamiento del *hardware*, otras funciones pueden verse incluidas además de un sistema de emulación. Partiendo de la idea de sistemas como *Steam*[1] o juegos *online*, una plataforma de puntuaciones o un repositorio de archivos de juegos, pueden verse incluidos en estos sistemas de emulación de forma relativamente sencilla, si se compara con las posibilidades que había hace diez o quince años.

1.2 Objetivos

El objetivo general de este documento es la redacción de un manual para equipos basados en la tecnología de procesadores *ARM*, como lo es una Raspberry Pi 4, que aborde

[1] https://es.wikipedia.org/wiki/Steam

la instalación e integración de un sistema de emulación de juegos arcade como lo es *Retropie*, además de la instalación de una base de datos que actúe de *back-end* para nuestra máquina. En ella se creará una base de datos relacional para las diferentes clases que se usarán en nuestro sistema.

Por otra parte, nos encontramos con los siguientes objetivos específicos:

- Conocer las máquinas recreativas más famosas de los años 1980-1990.
- Conocer cómo funcionan los sistemas basados en *Linux* mediante una Raspberry Pi.
- Implementar un sistema de emulación basado en *Linux* para dicha máquina.
- Implementar un servidor *web* para el manejo de puntuaciones y estadísticas.

El principal destinatario de este documento será todo aquel que tenga conocimientos de sistemas *Linux* así como, desarrolladores de *software* o ingenieros informáticos que estén interesados en la instalación de un sistema de emulación arcade.

1.3 Estructura de la memoria

Las posteriores secciones se organizan en tres grandes bloques. El primer bloque consta de los capítulos 1, 2 y 3, y sirve de introducción para los temas tratados. El siguiente bloque se compone de los capítulos 4, 5 y 6, y habla del diseño y desarrollo de la máquina tanto la parte *hardware* (física) como el *software* desarrollado para alcanzar los objetivos anteriormente mencionados. El tercer bloque, constituido por los capítulos 7 y 8 sirve a modo conclusión donde se ven los resultados obtenidos y las ideas fundamentales de la memoria.

En cada capítulo se aborda de forma más extensa los siguientes temas:

- **Capítulo 1: Introducción.** En este capítulo se habla de forma genérica el tema a abordar además de los objetivos que se pretenden conseguir y el impacto generado después de este desarrollo.
- **Capítulo 2: Estado del arte.** Aquí se pretende hablar de la situación actual de trabajos relacionados con nuestro tema. Se hablará de las máquinas recreativas de los años 80. Se pondrá en situación el problema a desarrollar y se comparará con trabajos anteriores, resaltando las diferencias y similitudes con estos.
- **Capítulo 3: Análisis del problema.** En este capítulo se abordarán las posibles soluciones, con las que se puede realizar nuestro proyecto. Se hablará de las tecnologías que se van a usar y qué enfoque tener en la elaboración de nuestro código.
- **Capítulo 4: Diseño de la solución.** En este capítulo, se diseñará de forma específica cada componente de nuestro trabajo, explicando detalladamente la solución escogida, desde la parte física de la máquina, hasta el código necesario que usará nuestra Raspberry Pi.
- **Capítulo 5: Desarrollo de la solución propuesta.** Se verán las fases por las que hemos pasado hasta llegar a la solución final, incidiendo en los problemas encontrados y la forma de resolverlos.

- **Capítulo 6: Fase de implantación.** Este apartado sirve a modo de guía donde se aborda la implementación de nuestra solución en nuestra máquina y la puesta en marcha del servidor que albergará la base de datos.
- **Capítulo 7: Resultados.** Aquí veremos los resultados obtenidos y una pequeña demostración de nuestro sistema.
- **Capítulo 8: Conclusiones.** Daremos un repaso a todo lo que hemos visto en este proyecto relacionándolo con los estudios cursados en el grado. Además, propondremos algunas ideas que se pueden implementar posteriormente partiendo de este trabajo.

1.4 Impacto esperado

El periodo de auge o «Era Dorada»[2] de estas máquinas, como así lo llaman algunos autores, abarca la década de los 70 hasta principios de los años 80. Pese a ser un periodo de gran lucidez, podría haber sido aún más lúcido si cabe si se hubiese producido hoy en día. Imaginemos por un momento si todas estas máquinas y todos aquellas salas recreativas en aquella época hubiesen tenido una herramienta esencial hoy en día: Internet.

Pese a que las bases del internet tal y como lo conocemos hoy en día surgieron a principios de los años 60[3], no fue hasta finales del siglo XX cuando dió la luz globalmente a un internet más parecido al que tenemos hoy en día, periodo en el que las máquinas recreativas se estaban viendo sustituidas poco a poco por otro tipo de consolas más familiares como lo son la *NES*, *SNES*, *PlayStation 1* o similares.

Pues bien, lo que se propone en este trabajo es hacer uso de herramientas contemporáneas que hoy en día son accesibles por todo el mundo, integrándolas en sistemas con un factor de forma similar al de las máquinas recreativas de los años 80, pudiendo así crear aplicaciones que hace unas décadas no podrían haber dado luz sin el uso del internet. En este documento se deja planteada una de ellas, como es una aplicación móvil para ver puntuaciones de juegos arcade, que se podría desarrollar más adelante.

[2] https://disney.fandom.com/es/wiki/Era_Dorada_de_los_videojuegos_arcade
[3] https://informeglobal.com/quien-invento-el-internet-evolucion-historia

CAPÍTULO 2
Estado del arte

En este capítulo se pretende dar una visión general sobre las máquinas recreativas o máquinas arcade, hablando sobre la historia de las mismas, el impacto que las nuevas tecnologías ha tenido sobre ellas y el nuevo uso que estas pueden darle, rejuveneciendo así la etapa en la que los primeros videojuegos llegaron al mercado.

2.1 Concepto de máquina arcade

Las máquinas arcade, son un tipo de computadores que contenían un tipo de videojuego como mucho, y que surgieron a principios de los años 70, décadas antes de la aparición del primer computador personal. Se componen de un mueble de tamaño medio entre uno y dos metros de altura, con una pantalla y controles debajo de ella.

El término *arcade*, proveniente del Inglés, se usaba para designar las galerías(o arcadas) en las que se colocaban las primeras máquinas de este tipo. Más tarde se usó para denominar a los recreativos donde se colocaban éstas, hasta que finalmente hoy en día se usa para denominar a la máquina en sí.[1]

2.2 Evolución de las máquinas arcade

Los orígenes de las *arcade machines* o máquinas arcade son un poco difusos puesto que fueron evolucionando con la llegada de los primeros ordenadores de sobremesa y los videojuegos.

Para tener una visión más completa de lo que supone la llegada de las máquinas arcade, deberemos remontarnos a mediados del siglo XIX, cuando surgía por primera vez un concepto nuevo de juego. Ya por aquel entonces, en las ferias se vislumbraba lo que con la llegada de la tecnología digital, llegaría a ser lo que se conoce ahora como máquina arcade. Por supuesto, todos los aparatos eran mecánicos y no digitales como ahora. Los *bagatelles* (precursor del *pinball*)(figura 2.1a); las máquinas de fuerza, con las que debías golpear una campana con un martillo; el futbolín, patente Española de finales del siglo XIX, por Alejandro Finisterre (figura 2.1b); las escopetas de perdigones, etc. Todas estas máquinas recreativas, daban el pistoletazo de salida a lo que sería, años más tarde con la llegada de los microprocesadores, un movimiento que hoy en día está más en auge que nunca si cabe[2].

[1] What did the word *"arcade"* mean before video games?
[2] http://indicelatino.com/juegos/historia/recreativas/

(a) Bagatelle

(b) Jugando al futbolín en California

Figura 2.1: Distintas máquinas recreativas de principios del siglo XX

Según la página *web*, RetroInformática[3], el primer videojuego como tal, se desarrolló en 1958 por William Higginbotham y es el mundialmente conocido *Tennis for Two* (figura 2.2), que consistía en un osciloscopio que simulaba una red de tenis con una "pelota"que se pasaba de un lado para otro con dos controles. Fue el primero de la historia, en permitir el juego entre dos jugadores humanos. Un mueble de unos dos metros de altura, con una pantalla a la altura del pecho y uno o varios controles debajo de ella.

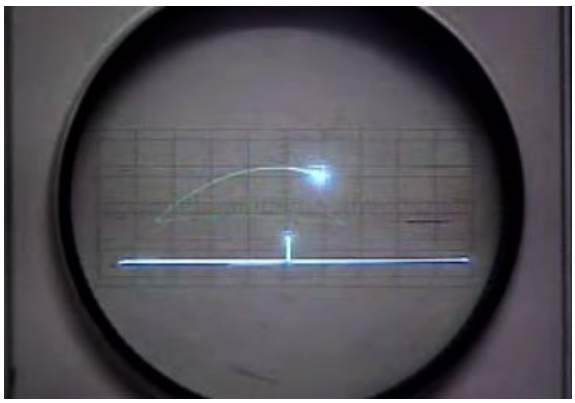


Figura 2.2: Tennis for two

En 1971, Bill Pitts y Hugh Tuck, creaban un juego que para muchos también fue un pilar importante de la historia de los videojuegos. *Galaxy Game*, una versión de su antecesor *Space War*, el cual estuvo en la Universidad de Stanford(California) entre 1971 y 1979 como parte de un experimento, y que llegó a generar colas de más de una hora de espera para jugar al mismo (figura 3.2). El mismo año, Nolan Bushnell también lanzaba al mercado *Computer Space* (figura 2.3b), aunque su repercusión no fue muy grande a nivel comercial, sí lo tuvo en los campus universitarios.

Un año más tarde, en 1972, la compañía estadounidense Atari fundada por Nolan Bushnell, adaptó la idea de *Tennis for Two* a una forma más familiar, dando lugar al *Pong* (figura2.3a), que se integró en un sistema que se asemeja algo más a lo que se conoce hoy

[3] https://www.fib.upc.edu/retro-informatica/historia/videojocs.html

por hoy como máquina arcade[4]. Una idea simple a la par que adictiva, hizo que cuando la máquina se lanzó por primera vez al público en una gasolinera local, dejase de funcionar a los pocos días por la cantidad de monedas que la gente había introducido en ella.

En los años venideros, cantidad de juegos fueron lanzados para máquinas arcade. Por poner algunos ejemplos, *Pacman*[5], *Donkey Kong*[6], *Gunfight*[7], *Space Invaders*[8], *Street Fighter*[9], *MetalSlug*[10] fueron algunos de los juegos más populares que llenaban las salas recreativas años atrás y de los que hoy en día se siguen haciendo versiones para consolas modernas.

(a) Pong

(b) Computer Space

Figura 2.3: Comparación entre el diseño original de la máquina de *Pong* y *Computer Space*

Como hemos mencionado, el concepto de máquina arcade no se consolidó hasta principio de la década de 1970 con la comercialización de los juegos para estas máquinas. Con la entrada al mercado de *Pong* y *Computer Space* (2.3a 2.3b), se daba entrada a una época que iba a estar marcada por estas máquinas, hasta su decadencia que coincidía con la llegada de otras consolas más familiares y potentes. La capacidad de cómputo de estas consolas, a la par que la llegada a los hogares, hizo que compañías como Nintendo[11] apostaran finalmente por un formato de forma más portable. *Donkey Kong*(1981), desarrollado originalmente para una máquina arcade, se vio trasladado más tarde al resto de consolas portables de la compañía.

[4] http://www.pong-story.com/
[5] https://pacman.com/en/
[6] https://es.wikipedia.org/wiki/Donkey_Kong_(videojuego)
[7] https://kotaku.com/in-search-of-the-first-video-game-gun-5626466
[8] https://spaceinvaders.square-enix-games.com/
[9] https://streetfighter.com/
[10] http://www.neo-geo.com/reviews/neo-reviews/metalslug/metalslug1.html
[11] https://www.nintendo.es/

2.2.1. Situación actual

Con la comercialización de los computadores de sobremesa y la llegada de las consolas con videojuegos más potentes a finales del siglo pasado, el mercado de las consolas arcade se vio afectado drásticamente. Como Daniel Valerón destaca[12], solo en algunos barrios de Japón continúa el fanatismo por esta época, llegando a tener edificios enteros dedicados a estos aparatos.

La llegada al mercado de los microcomputadores, ha dado un vuelco total a esta situación, retomando nuevamente este sentimiento *arcade* en videojuegos, ya casi extinto. Muchos son los proyectos que se pueden hacer con estas computadoras, y uno de ellos que triunfa entre la gente que se está iniciando es este mismo: Una máquina arcade con Raspberry Pi. Sin ir más lejos, la propia página oficial de la compañía anglosajona, tiene un mes entero de la revista oficial de Raspberry Pi (Noviembre, 2019), dedicado exclusivamente a este tipo de consola[13]. Algunos son los factores que podrían haber contribuido a este vuelco respecto a años atrás, poniéndose incluso en paralelo con entregas actuales *AAA*[14], no en cantidad de ventas o jugabilidad si no en repercusión mediática. Esto puede deberse entre otras cosas al precio del computador (alrededor de 50$), la buena relación entre el precio y la capacidad de cómputo, una rápida curva de aprendizaje en todo tipo de proyectos, el tamaño del dispositivo o una gran comunidad detrás que puede resolver cualquier tipo de problema con el que se encuentre. Como es lógico, los videojuegos que requieran de altas prestaciones siempre estarán a la cabeza del mercado, pero cierto es que además de estar retomando videojuegos arcade para incluirlos en emuladores, también hay una cantidad pequeña de desarrolladores que apuestan por la creación de videojuegos con un estilo retro para sus entregas. Sin entrar en más detalle algunos ejemplos que alcanzaron el *top* de ventas con este estilo son *Undertale*[15], *Super Meat Boy*[16], *Crypt of the NecroDancer*[17], *Nidhogg*, *Terraria*[18], entre otros.

Figura 2.4: Máquina arcade con una Raspberry Pi

[12] https://powerups.es/maquinas-recreativas-auge-y-decadencia-de-un-sector-casi-extinto/
[13] https://magpi.raspberrypi.org/books
[14] https://es.wikipedia.org/wiki/AAA_(industria_del_videojuego)
[15] https://undertale.com/
[16] http://www.supermeatboy.com/
[17] https://braceyourselfgames.com/crypt-of-the-necrodancer/
[18] https://terraria.org/

Pero no solo eso, sino que además siguen existiendo zonas recreativas con máquinas arcade como las de antes, que están empezando a resurgir poco a poco. Ya sea porque aquellos niños que vivieron de pleno la época dorada de los videojuegos[19], ahora son mayores y buscan de alguna forma reencontrarse con su niño interior o porque el abaratamiento del *hardware* ha propiciado que se puedan restaurar de forma relativamente sencilla y con ello el sentimiento arcade ya casi perdido, vuelva a estar vigente. Claros ejemplos de que a la cultura arcade aún le queda salas que llenar son:

- **Arcade Vintage:** Esta asociación cuenta varios locales en la provincia de Alicante. En primer lugar, en la localidad de Ibi, la capital del juguete, se halla uno de los museos más importantes de la Comunidad Valenciana en cuanto a máquinas arcade. Cuenta con una colección de unos 70 títulos que va rotando semanalmente. Además del museo, otra localidad alicantina, en Petrer, cuenta con un local del que uno se puede hacer socio y con ello disfrutar de algunos de los títulos que se muestran en el museo.[20]

- **Arcade Planet:** Este salón se encuentra en Sevilla y presume de ser uno de los más grandes de España. Similar al de Petrel pero del que se puede percibir un aire aún más retro que el anterior. Cuenta con una colección de no menos de 150 máquinas arcade, con títulos desde los más antiguos a los más modernos.[21]

- **RetroArcadia:** Situada en Valencia, dedicada a la restauración de componentes arcade.

- **Arcade CAT:** Situada en Hospitalet de Llobregat, Barcelona. Cuenta con una colección de unas 120 máquinas arcade. Además, como las anteriores, es una asociación sin ánimo de lucro, en la que todos los fondos recaudados van exclusivamente al mantenimiento de la asociación y los eventos relacionados. [22]

- **Retroinformática:** Por último, nuestra escuela cuenta con el Museo de Informática[23], un taller dedicado al culto de la década de los 80. En él se llevan a cabo cada año numerosos eventos como Retrópolis, que pretende divulgar la cultura del videojuego y la informática clásica.[24]

Además, existen talleres donde artesanos, siguen creando máquinas arcade desde cero. Existe por ejemplo el caso Enrique Ortiz, un gerente de ventas al por menor que se pasó al mundo de la artesanía de estas máquinas, después de pasar un virus que le afectó al corazón y que le hizo replantearse su trabajo. Este lleva a cabo el taller de Universo Arcade, en Madrid, en el que se encarga del diseño y montaje de estas máquinas. Como muestra de su popularidad, personajes públicos como el ex-jugador del Real Madrid, Iván Helguera o el jugador del Fútbol Club Barcelona, Gerard Piqué, compartieron en sus redes sociales fotos con máquinas de Universo Arcade. [25].

[19] https://disney.fandom.com/es/wiki/Era_Dorada_de_los_videojuegos_arcade
[20] https://museoarcadevintage.com/
[21] https://mentero.es/portada-pequena/entrevista-a-arcade-planet/
[22] https://www.arcade.cat/ca/front-page/
[23] https://museo.inf.upv.es/es/
[24] http://museo.inf.upv.es/es/retropolisvalencia/
[25] https://www.eldiario.es/hojaderouter/tecnologia/maquinas-arcade-recreativas-videojuegos-retro_1_3285482.html

Figura 2.5: Diseño propio de mueble por Bricoarcade

2.3 Trabajos relacionados

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

Un trabajo de final de grado que está directamente relacionado con el nuestro es el de Álvaro Roig, que también fue estudiante de la ETSINF[1]. A pesar de ser bastante similar al nuestro, cabe destacar numerosas diferencias que se han llevado a cabo en los dos trabajos.

Una de las principales diferencias claras es el desfase tecnológico que existe entre su trabajo y el nuestro. Lo que su autor pretende también, es crear una máquina arcade desde cero, que emule videojuegos de las máquinas originales. Para ello de nuevo hace uso de una Raspberry Pi con la diferencia que en su trabajo, se usa el modelo 1B anterior al nuestro, que carece de la potencia y de algunas conexiones externas que el nuestro ya posee de base. Por ello, para suplir estas carencias de conectividad hace uso de, además de la Raspberry Pi, de un Arduino Mega 2560 al que conecta un módulo radiofrecuencia. En añadido, las conexiones de sus controles se hacen mediante los pines entrada/salida a diferencia de los nuestros que usan conectores USB.

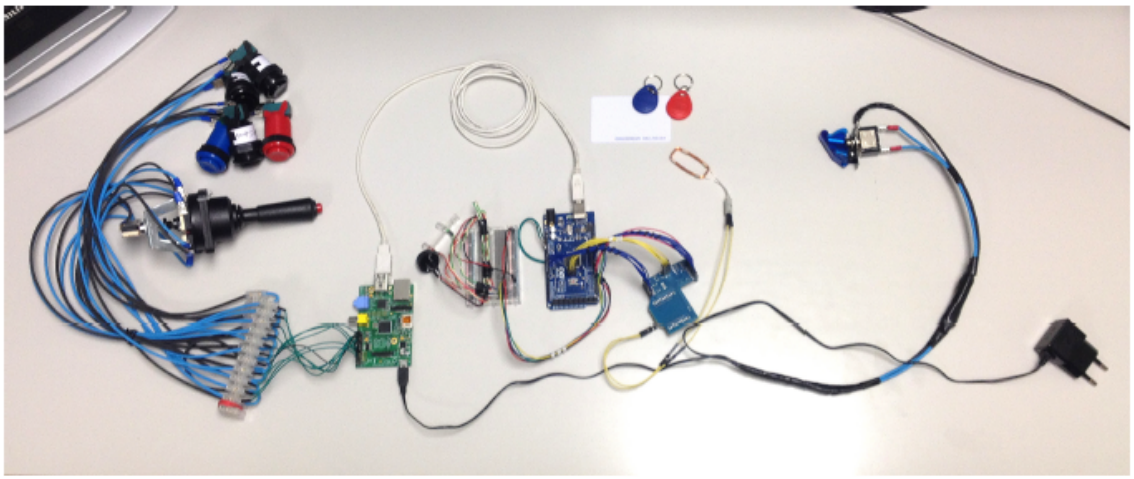
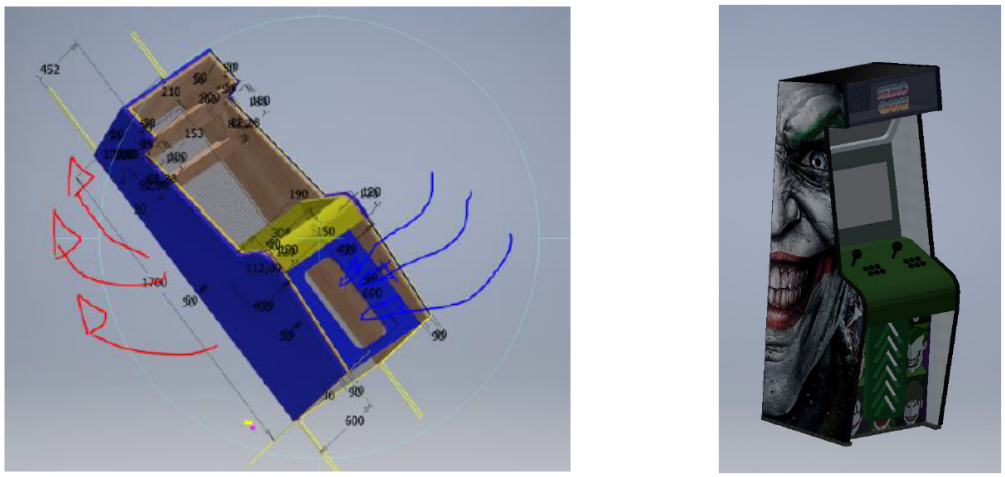


Figura 2.6: Conexiones hardware trabajo Álvaro Roig

Como se puede observar en la imagen (figura 2.6), Álvaro hace uso de hasta tres dispositivos Arduino para el módulo radiofrecuencia y gran cantidad de pines entrada/salida para los controles. Como se verá más adelante, nosotros haremos uso de únicamente un módulo radiofrecuencia conectado directamente a los pines I/O de la Raspberry y dos puertos USB para la conexión con los controles.

2.3.2. Diseño de un nuevo modelo de máquina recreativa personalizable

Otro trabajo que también está de alguna forma relacionado con el nuestro es el de Andrés Menárguez, esta vez de la escuela de industriales (ETSII)[2]. Al ser un trabajo de otra escuela, sus objetivos difieren un poco a los nuestros ya que se centra principalmente en la manufacturación externa de la máquina, sin entrar mucho en detalle sobre la parte *hardware* y *software* de la misma. Pese a ello, podemos extraer algunas características importantes en su trabajo como el diseño que toma.



(a) Ensamblaje de la máquina arcade por Andrés Menárguez

(b) Diseño externo de máquina arcade por Andrés Menárguez.

Figura 2.7: Distintos diseños externos de máquina arcade

2.4 Propuesta

De forma resumida, lo que este trabajo mostrará es de cómo mediante el uso de herramientas actuales como un microcomputador basado en arquitectura ARM, un servidor que hará uso de una base de datos relacional y una lógica de programa que pueda crear conexiones entre todos los componentes, se puede crear una consola recreativa con nuevas funciones más acorde a esta época. En ella, implementaremos un sistema de guardado en una base de datos externa, que podrá almacenar puntuaciones obtenidas en videojuegos, horas jugadas o estadísticas diversas.

La característica principal de nuestro sistema que hace que destaque frente al resto, será la lógica del programa que crearemos. En otros proyectos[1], hemos visto cómo se ha logrado crear una máquina arcade desde cero, similar a la nuestra, pero con la diferencia que ninguna aprovecha el potencial de la conexión de red que permiten los modelos actuales de Raspberry Pi. Además, se integrará un módulo de *RFID(Radio Frequency Identification)*, que permitirá identificar a un usuario con una tarjeta *RFID* o *NFC*. Posteriormente, se diseñará la máquina físicamente y se fabricará manualmente. Mediante esta característica, en posibles proyectos futuros, se podría crear una red que intercambiase información relativa a los juegos, un tipo de cliente *BitTorrent* para intercambiar ROMs

de juego[26], un modelo cliente-servidor para almacenar estadísticas de una amplia red de máquinas arcade, una aplicación para visualizar de forma más familiar estar puntuaciones, entre otras cosas.

[26] https://es.wikipedia.org/wiki/Imagen_ROM

CAPÍTULO 3
Análisis del problema

A continuación, se analizará detalladamente la idea propuesta incidiendo en cada parte a desarrollar. En cuanto a la parte física de la máquina, se verán los distintos tipos de diseño propuestos, teniendo en cuenta los factores de forma más adecuados para nuestro sistema. En cuanto a la persistencia que deberá tener nuestra aplicación, se deberá tener en cuenta qué se adapta más a lo que queremos, ya que tendrá que guardar distintos tipos de objetos lógicos, ya sean puntuaciones, jugadores, juegos, etc. Además, toda esta información se tendrá que manejar con la lógica de aplicación correcta ya que por cada juego, habrá de tratar la información relativa a las puntuaciones de forma distinta.

3.1 Especificación

La función principal que tendrá que cumplir nuestra máquina será la de funcionar de forma similar a una máquina arcade original con el añadido de que tendrá que poder conectarse a internet para compartir puntuaciones y datos estadísticos de los juegos que deseemos.

Lo primero que tendremos que analizar, será el número y tipo de casos de uso para los que queremos que se utilice nuestra máquina. Las máquinas arcade originales, solían estar en un mismo lugar durante mucho tiempo a causa de su tamaño y su peso. La nuestra a pesar de no ser tan grande ni tan pesada, la querremos estática en un lugar y su movilidad será reducida. Además, nuestra máquina hará uso de las tecnologías actuales como es la conexión internet mediante WiFi, por defecto en la mayoría de microcomputadores. Por último, identificará a cada usuario en el momento en el que inicia un juego con un identificador *RFID/NFC* y guardará su identificador además del resto de datos de la sesión de juegos en una base de datos. Por tanto, los requisitos y casos de uso de nuestra máquina serán los siguientes:

3.1.1. Requisitos

Requisitos Funcionales

- El usuario iniciará un juego previamente instalado en la máquina y se identificará mediante una tarjeta *RFID/NFC*. Si no se identifica con ninguna tarjeta, jugará la sesión con un usuario por defecto.

- El usuario puede identificarse en cualquier momento antes de que cierre la sesión de juego.

- Al terminar la sesión de juego, las puntuaciones obtenidas en los diferentes juegos se almacenarán en una base de datos externa al equipo.

- Si el usuario es administrador del sistema, podrá acceder a la base de datos y visualizar, actualizar o borrar los datos referentes a sesiones, juegos y usuarios.

- Además, el administrador del sistema, podrá agregar o eliminar juegos de forma sencilla. Si en un futuro hubiese más de una máquina en la red, este proceso tampoco tendría que ser costoso.

Requisitos no funcionales

- El guardado de datos de juego en la base de datos debe ser transparente para el usuario.

- Si al acabar la sesión, no ha sido posible conectarse a la base de datos, se almacenará la sesión de juego y el id del usuario y se subirá cuando sea posible.

- El sistema debe de estar implementado en un procesador de arquitectura ARM.

- Los datos referentes a cada sesión del jugador deben poder ser accedidos fácilmente desde la misma red local para posibles proyectos futuros.

- Los programas y componentes usados deben ser a ser posible de código libre.

3.1.2. Modelado conceptual: Casos de uso

A continuación, identificaremos los posibles casos de uso mediante la notación UML.

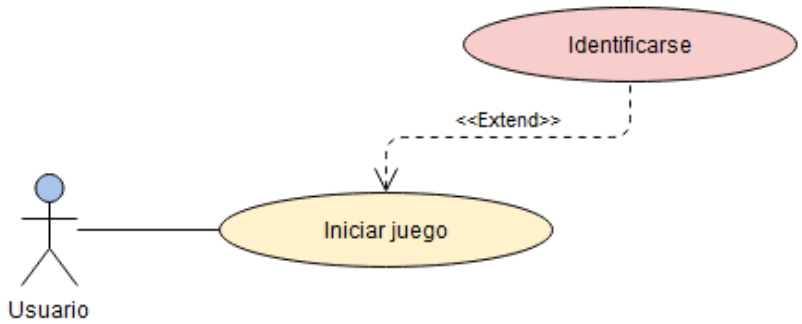


Figura 3.1: Diagrama de casos de uso del usuario

El único caso de uso que tendrá nuestro usuario/jugador, será el de jugar a videojuegos. Opcionalmente durante la duración de la sesión se podrá identificar con una tarjeta RFID/NFC. El usuario deberá estar dado de alta en el sistema antes de poder identificarse. A continuación, expondremos un caso de uso en concreto.

Caso de uso	Iniciar juego
Actor	Usuario
Descripción	El usuario realizará una sesión de juego con un inicio y un final determinado. Opcionalmente el usuario se podrá identificar.
Pre-condición	-El servidor web donde se almacenarán los datos relativos a la sesión debe de estar encendido cuando se finaliza la sesión. -El usuario debe de estar dado de alta antes de poder identificarse en el sistema.
Flujo general	-El usuario inicia un juego. -Antes o durante la sesión de juego, el usuario podrá identificarse o no. -El usuario cierra el juego.
Post-condición	- Los datos relativos a la sesión de juego se almacenarán en el servidor de base de datos. -Si el usuario no se ha identificado, se guardará la sesión con un usuario predefinido.

Tabla 3.1: Caso de uso de Iniciar juego

3.2 Análisis de las posibles soluciones

Para analizar las distintas soluciones posibles para nuestro problema, deberemos de tener en cuenta que para nuestra implementación necesitaremos tanto una parte física como lógica. Nuestra parte lógica tendrá los requisitos previamente descritos que analizaremos a continuación. Para la parte física, buscaremos un diseño de máquina arcade que se adapte a nuestras necesidades y lo plasmaremos en madera para montarlo posteriormente.

3.2.1. Diseño interno

En la parte lógica, existen dos secciones principales necesarias para el funcionamiento correcto de nuestro máquina. Las tecnologías de las que la máquina hará uso para que pueda hacer funcionar los juegos, además del código necesario para que envíe las puntuaciones a la base de datos y la base de datos que se encargará de dar una capa de persistencia.

Tecnologías para el desarrollo

En resumen, esta parte intenta responder a dos preguntas: Con qué y por qué. Con qué herramientas desarrollaremos nuestra máquina y por qué vamos a usar esas herramientas. Primero, deberemos conocer qué herramientas existen actualmente que se adecúen a nuestras necesidades. Nuestra máquina dispondrá de un sistema operativo base que tendrá la característica principal de tener que estar implementado para una arquitectura de procesador *ARM(Acorn RISC Machine)*. Es decir, que haga uso de un conjunto de instrucciones reducido. Lo que esto supondrá es que de media, aumentará el tiempo de ejecución de los programas pero la CPU será más eficiente energéticamente hablando[1].

Actualmente, existen varios sistemas operativos implementados para una arquitectura *ARM*. Algunos de ellos están todavía en desarrollo, pero los más importantes ya tienen una o varias versiones estables que podrían sustituir perfectamente a sistemas operativos basados en arquitectura *x86*[2]. Mayormente, nos encontraremos sistemas basados en Linux, aunque otras compañías como *Windows* y *Apple*, ya están empezando a implementar

[1] https://www.profesionalreview.com/2017/11/26/procesadores-x86-vs-arm-diferencias-ventajas-principales/
[2] https://es.wikipedia.org/wiki/X86

sistemas para esta arquitectura. Listamos los principales sistemas operativos de libre uso a continuación:

- **Raspberry PI OS:** Anteriormente denominado Raspbian, para muchos, el mejor sistema operativo para la Raspberry Pi. Es el sistema operativo oficial de la máquina y por tanto está diseñado específicamente para esta. Además de ser ligero, su principal diferencia respecto al resto es la comunidad que hay detrás ofreciendo soluciones para cualquier tipo de problema[3]. También es el único sistema operativo, diseñado por los ingenieros de Raspberry Pi.

- **Ubuntu Server and Desktop:** La conocida distro de Linux, ahora dispone de una versión para *ARM*. Posee las mismas características principales que la versión para *x86*. Esto puede causar cierta adversidad entre los usuarios porque por una parte, podemos tener prácticamente un PC de sobremesa por un precio alrededor de los 50$ pero por otra parte, la potencia de cómputo de los procesadores actuales *ARM* no puede igualar hoy en día a los *Intel* o *AMD* de gama más alta, creando así carencias en ciertos aspectos de rendimiento en sistemas como este.

- **Kali Linux:** Un sistema desarrollado por y para los amantes de la seguridad. Si bien es un sistema muy potente con más de 600 herramientas de penetración y testeo sobre las redes, se le puede dar un uso muy interesante sobre un máquina portable como lo es la Raspberry Pi[4].

- **Manjaro:** Basado en Arch Linux, enfocado principalmente a la facilidad de uso, puede llegar a ser un buen sustituto de sistemas algo más pesados para esta computadora[5].

Persistencia

A continuación, hablaremos de la forma en la que nuestra máquina hará uso de la información proporcionada por nuestros juegos para luego ser procesada por nuestra aplicación.

Hay diversas formas de almacenar información en un sistema como por ejemplo, ficheros planos de texto, árboles binarios de búsqueda, bases de datos relacionales... Cada una de ellas tiene una función distinta y según qué tipo de información se quiera guardar y cómo se quiera guardar, haremos uso de unas o de otras. Por lo tanto, primero hemos de saber qué información queremos guardar y cómo la queremos guardar.

Nuestra máquina estará diseñada específicamente para video-juegos arcade. Estos juegos, a diferencia de los actuales, guardan muy poca información a causa de la poca capacidad de memoria que tenían los sistemas de juego anteriores. Algunos no superaban los pocos *MB(MegaBytes)* de memoria incluyendo el juego. Por poner un ejemplo, el juego sobre el que trataremos más tarde *Pang (1989)*, guarda únicamente información de los diez mejores jugadores, incluyendo la puntuación, un acrónimo de tres caracteres que identifica al jugador y el nivel al que ha llegado. Tal archivo no supera un **kilobyte** de memoria. Pero bien, esta no es la única información que queremos guardar. Además, queremos identificar al usuario que está usando en ese momento en la máquina, relacionándolo con el juego que está jugando y la puntuación que ha obtenido. Para ello, usaremos una base de datos relacional.

[3] https://www.raspberrypi.org/forums/viewtopic.php?t=258238
[4] https://www.kali.org/docs/arm/
[5] https://manjaro.org/

Actualmente, existen varios sistemas de gestión de base de datos que según qué necesidades tengamos, nos será más adecuado uno u otro. Hablaremos de algunos de ellos centrándonos en su rendimiento y en el tipo de licencia que tiene.

- **MySQL:** MySQL es el gestor más popular de base de datos de código libre del mercado. Sus inicios se remontan a 1995 donde se lanzó al mercado la primera versión. El proyecto se vendió en 2008 a Sun Microsystems (actualmente en propiedad de Oracle). Las características principales de este es su gran eficiencia, disponibilidad y escalabilidad
- **MariaDB:** MariaDB es una bifurcación del proyecto original de MySQL después de la compra del proyecto a manos de Oracle. Sus fundadores se distanciaron de este para poder seguir con la filosofía original y no crear así un conflicto de intereses con MySQL. El rendimiento de MySQL y MariaDB son prácticamente idénticos ya que parten de la misma base. Su principal diferencia respecto al anterior es que es totalmente *open-source* a diferencia del primero que está sometido a las restricciones de Oracle[6].
- **SQLite:** Es uno de los sistemas de gestión más ligeros en el mercado. Por esto mismo, algunas de sus funciones se verán reducidas respecto al resto de opciones. Además, no es multi-usuario, hecho que supondrá que no se puedan hacer varias transacciones al mismo tiempo. Su instalación es sin servidor, ideal para dispositivos móviles que requieran de una base de datos para una aplicación sin conexión a internet.
- **PostgreSQL:** Es un muy potente sistema de gestión de base de datos objeto-relacional. Es libre y de código abierto (o *open-source*). Pese a ser muy potente es aún muy nuevo y hay pocas herramientas para gestionarlo. [7]

Hemos visto cuatro tipos de gestores de base de datos de los muchos que hay. Cabe destacar que todos ellos son *open-source* excepto MySql que pese a ser libre, tiene licencia dual con Oracle. Hemos visto algunas de sus características y de entre todos ellos hay algunas que no se nos adaptan a nuestras necesidades. Para empezar, hemos visto que nuestro sistema podría ser escalable en un futuro, permitiendo que varias máquinas se puedan conectar simultáneamente a nuestro servidor para realizar las consultas pertinentes. SQLite no cumple con esto ya que a causa de su ligereza, que podría haber resultado interesante en nuestro sistema, carece de un requisito importante y es el hecho de que pueda ser multi-usuario para en un futuro poder conectar varias máquinas arcade entre sí. La eficiencia y características de los otros tres no crea conflicto con nuestros requisitos, por estar bastante optimizados. Es más, quizá PostgreSQL sea demasiado pesado si cabe. A pesar de estar acorde con todo nuestros requisitos, está pensado más bien para dotar a un sistema de funciones complejas y a nosotros quizá no nos interesa por el tamaño de nuestra base de datos. Es cierto que en un futuro podría incrementarse mucho más, pero el número de tablas no variaría demasiado y con ello PostgreSQL no estaría aprovechando al máximo su potencial. Respecto a MySQL y MariaDB, es cierto que MySQL está más extendido y tiene una gran comunidad detrás pero tras la compra a manos de Oracle, el proyecto dejo de ser totalmente *open-source*. Además, los fundadores de MySQL son los que actualmente llevan el proyecto de MariaDB. Las funcionalidades de este y MySQL van a la par ya que los desarrolladores querían que hubiese compatibilidad completa entre estos dos. La principal diferencia es que la actual versión de MariaDB sí que es *open-source* y tiene una comunidad detrás de desarrolladores que cada día es más numerosa. Por esta razón nos decantamos finalmente por MariaDB.

[6] https://www.ionos.es/digitalguide/hosting/cuestiones-tecnicas/mariadb-vs-mysql/
[7] https://www.postgresql.org/

3.2.2. Diseño externo

Para tener en mente las dimensiones de una máquina original tendremos que remontarnos a los años 70 con la aparición de la primera máquina recreativa de la historia (figura 3.2). *Galaxy Game* o el juego de los marcianitos, como comúnmente se le conoce aquí, fue el primer juego recreativo del que se tiene constancia.

Figura 3.2: Galaxy Game en el museo de las computadoras de California

Como se puede observar en la figura, la máquina no sobrepasaba el metro de altura y el formato de forma difiere bastante de lo que se conoce hoy en día como máquina arcade. En la imagen siguiente podemos ver que el factor de forma del *Pac-man* (figura 2.3a) ya se asemeja un poco más a lo que conocemos hoy. Unas dimensiones de entre 1,5 y 2 metros de altura, controles debajo de la pantalla y una ranura para insertar monedas.

(a) Máquina recreativa Pac-man

(b) Bar-top arcade machine

Figura 3.3: Comparación entre una máquina de tamaño completo y *bar-top*

En los años venideros, hubieron muchas variantes del tamaño y los componentes pero las dimensiones de la misma no cambiaba demasiado. Una pantalla, controles debajo de ella y uno o dos ranuras para monedas según los jugadores. La altura rondaba la estatura media de una persona.

A medida que avanzaba la tecnología, los componentes se iban reduciendo y todo el tamaño que antes era necesario para cableado y placas se desperdiciaba en la máquina. Por consiguiente, surgieron las *bar-top arcade machine* (figura 3.3b) en las que habían recortado la parte de abajo para convertirlas en algo más portátil y menos pesado que, como su nombre menciona, se podría colocar encima de una mesa o barra de algún bar.

Para nuestro trabajo, debido al alto gasto que supondría una máquina de tamaño completo, nos decantamos por las de sobremesa o *bar-top*, dado que para demostrar nuestro concepto, estamos únicamente limitados por las dimensiones de nuestra pantalla que sería a lo que se debería ajustar el resto del mueble. Además, queremos que esté encima de una mesa o algún sitio elevado para recortar en gastos y aumentar en portabilidad.

Por lo tanto, un tamaño razonable debería estar entre 0.4 y 0.8 m^3, aproximadamente, a causa de la pantalla. Actualmente, los monitores más usados están entre 20" y 30". Un tamaño de unas 20 pulgadas para nuestro monitor será más que suficiente. Más adelante veremos diseños específicos con los que plasmar en madera nuestra idea.

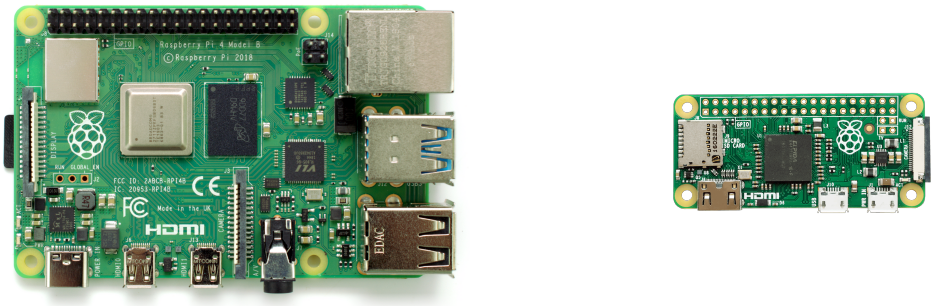
Componentes

Tal y cómo hemos mencionado, el núcleo principal de nuestro programa será una *Raspberry*. Este nombre, al igual que otras marcas de tecnología, no se refiere a un dispositivo en concreto sino a una serie de distintos computadores que según para qué usuarios van destinados, tienen unas características u otras. Sus funciones pueden llegar a ser similares a las de un computador de sobremesa, a diferencia de la potencia de cómputo y el precio. La licencia de estas computadoras está registrada pero su uso es libre, por lo que cualquiera que tenga los conocimientos oportunos y los componentes, podría crear una máquina similar, así que la variedad de estas es bastante amplia. En este apartado, se tendrá en cuenta únicamente los modelos de la compañía que la fundó, además de otros componentes que consideramos para nuestro núcleo principal y finalmente los descartamos. A continuación, se hablará brevemente de cada una de ellas.

- **Rasperry Pi 2B:** Este modelo fue lanzado en el año 2014 y su principal mejora respecto a los anteriores modelos era su procesador, ya que pasaba de uno a cuatro núcleos y de 700Mhz a 900MHz de frecuencia. En cuanto a puertos, contaba con cuatro conectores USB, un conector Ethernet RJ-45, un conector HDMI y un conector jack para la salida de audio. Además de los cuarenta pines de entrada/salida que ya se incluían en la anterior versión.

- **Raspberry Pi 3B+:** Lanzadas al mercado en el año 2018. Su principal diferencia respecto a los modelos anteriores es que pasa a tener una arquitectura de 64bit además de mejoras en rendimiento de CPU y en memoria.

- **Raspberry Pi 4B:** La Raspberry Pi 4 es el último modelo de la compañía inglesa, lanzado al mercado en el año 2019. Las novedades respecto a los modelos anteriores, fueron la adición de dos puertos microHDMI, sustituyendo al HDMI de tamaño completo de los modelos anteriores. Se añade un puerto HDMI pasando a tener 2 microHDMI en el que, teóricamente, pueden funcionar dos pantallas 4K a 60Hz. Además, se añaden dos puertos USB 3.0 y se mejora el procesador y la conectividad *Ethernet* y WiFi. Existen modelos de 2GB, 4GB y 8GB de memoria RAM.



(a) Raspberry Pi 4

(b) Raspberry Pi Zero

Figura 3.4: Distintos modelos de Raspberry

- **Raspberry Pi Zero/Raspberry Pi Zero W:** Por último, tenemos los computadores más pequeños de esta compañía. Fueron lanzados en el año 2015. Tienen un microprocesador Broadcom BCM2835 mono-núcleo que funciona a 1Ghz, 512MB de RAM y un puerto MiniHDMI. La gran ventaja de este *mini-computador* es su tamaño, pues este no supera a un billete de 5 dólares. Además, se han eliminado la mayoría de los puertos innecesarios para que se puedan incluir en sistemas de reducido tamaño, con casi las mismas prestaciones que el resto de modelos de mayor tamaño.

Los modelos 1A y 1B no se incluyen en esta lista debido a su antigüedad ya que, los primeros modelos de esta compañía salieron al mercado en 2012 pero ya no se fabrican, lo que supondría un problema para su obtención.

Pero además de estas opciones, al inicio de nuestro proyecto consideramos la posibilidad de no incluir una Raspberry Pi 4 como núcleo, si no otros computadores con una potencia de cómputo similar que debido a su año de salida se pueden encontrar en páginas de productos de segunda mano a un precio asequible. Es decir, reusar equipos viejos para darles una segunda utilidad que no requiera mucha carga. Eliminando los componentes innecesarios, podemos dejar únicamente la placa base del equipo para incluir las conexiones necesarias. Vemos algunos a continuación:

- **HP Compaq 6720s:** Esta computadora portátil fue lanzada al mercado en el año 2007. Entre las características que posee cuenta con un procesador Intel Core2 Duo con una frecuencia de hasta 2,4Ghz. Los gráficos se procesan con un Intel GMA X3100 y la memoria RAM es expansible hasta los 4096MB. Es cierto que en cuanto a cómputo está a la par de la Raspberry Pi 4, pero carece de la eficiencia que esta posee. En su punto álgido puede hacer gasto de unos 50W mientras que la Raspberry tan solo 5W. Además, nos sería imposible introducir el módulo de radiofrecuencia ya que no posee los pines de entrada/salida. Otro punto a favor de la Raspberry es que el procesador del HP tiene arquitectura *x86*, lo que produce ese aumento de consumo[8].

Resumiendo, los modelos 3B+ y 4B de la Raspberry Pi serían ideales para nuestras necesidades, por su capacidad de cómputo y conectividad. Debido a que el modelo 4B es más actual y posee una comunidad más activa que pueda haber resuelto ya los problemas que se nos vayan presentando por el camino, usaremos este en nuestro proyecto.

[8] https://support.hp.com/my-en/document/c01129747

3.3 Solución propuesta

El núcleo de nuestra máquina será una Raspberry Pi 4 por ser ampliamente conocida y tener una gran comunidad detrás que puede resolver los problemas que nos vayan apareciendo. La Raspberry Pi 4, como sus anteriores versiones y otros formatos de forma de la misma compañía usa Raspbian, un sistema operativo basado en Debian adaptado a estos computadores para poder acceder por ejemplo, a los pines de entrada/salida, conexiones externas a través de un conector CSI para una cámara o un conector DSI para una pantalla táctil. Estas características las hemos visto también en otro trabajos anteriormente mencionados con la diferencia que ahora, la tecnología usada integra alguna que otra funcionalidad que antes no lo hacía, además de ser más potente y eficiente. En el trabajo de Álvaro Roig[1] por ejemplo, se usaba también un Arduino UNO para el control del audio. Es nuestra propuesta, la Raspberry Pi 4 será la encargada de controlarlo puesto que incluye una salida de Jack para audio.

En este computador instalaremos un sistema operativo basado en Linux con una arquitectura de procesador *ARM* , por estar fuertemente ligado a la Raspberry Pi. Para llevar a cabo nuestro trabajo usaremos el sistema operativo Raspberry Pi OS, anteriormente denominado Raspbian. Para desarrollar los programas que nos harán falta para el manejo de puntuaciones de nuestros juegos, usaremos Python por ser un lenguaje con una curva de aprendizaje muy alta perfecto para nuestra necesidades, que no requieren de una alta demanda de optimización. Con ellos lo que se pretenderá hacer es almacenar todo la información relativa a nuestros juegos y usuario, ya sean puntuaciones, tiempo jugado o información propia del usuario. Además, usaremos una máquina virtual con Ubuntu 20 instalado para poder hacer uso de una base de datos relacional para nuestras consultas como lo es MariaDB. Sobre esta base de datos se podrán hacer las consultas pertinentes de forma local o externa si se configurase, además de guardar nuestra información en un lugar seguro.

3.4 Presupuesto

Como ya hemos mencionado, cuando acabemos la parte lógica de la máquina necesitaremos llevar a cabo la construcción de la misma, la cual necesitará una serie de materiales además de los componentes electrónicos. Además de estos materiales, se obvian las herramientas y utensilios facilitados que pueden ser de uso común.

Componente	Precio	Unidades	Subtotal
Raspberry Pi 4 de 4GB de RAM.	57,00€	1	57,00€
Tarjeta SD de 32GB	6,99€	1	6,99€
Cable HDMI a MicroHDMI	4,99€	1	4,99€
Módulo RFID	2,93€	1	2,93€
Joysticks Arcade Raspberry(kit 2 u.)	49,98€	1	49,98€
Pantalla LCD 21”	40,00€	1	40,00€
Regleta de enchufe	8,23€	1	8,23€
Tablón de madera(40x120cm)	15,99€	2	31,98€
Pintura en spray para madera	5,99€	6	35,94€
Bisagras	3,00€	1	3,00€
		Total:	241,94€

Tabla 3.2: Presupuesto aproximado de la máquina

CAPÍTULO 4
Diseño de la solución

A la hora de desarrollar la parte *software* se diseñará una base de datos relacional acorde a los requisitos de nuestro problema. Esta base de datos dará la funcionalidad de persistencia de datos, los cuales se procesarán mediante una serie de programas escritos en Python que resolverán nuestras necesidades. Por último, se verá un pequeño ejemplo de un juego en concreto, como lo es "Pang (1989)", originalmente desarrollado por *Mitchell Corporation* para distintos tipos de máquinas arcade. La funcionalidad básica de nuestro programa es la de procesar los archivos de puntuaciones del emulador y almacenarlos en la base de datos, para posterior uso.

4.1 Tecnología Utilizada

Como ya hemos mencionado, la CPU de la Raspberry Pi 4B está basada en la arquitectura *ARM(Acorn RISC Machine)*. Estos procesadores se caracterizan por contener un conjunto de instrucciones reducido *(Reduced Instruction Set Computer)* y con ello una menor cantidad de transistores que reducen el consumo eléctrico. Es ideal para dispositivos que requieran de una batería o posean una potencia limitada. A día de hoy los sistemas operativos basados en esta arquitectura están auge, distribuidos a nivel global en los *smartphones*, en sistemas empotrados y hasta en algunos portátiles como anunció Apple para su nueva serie de alta gama.

4.1.1. Sistema Operativo Base

El sistema operativo oficial de la Raspberry es Raspberry Pi OS o Raspbian. Basado en Debian, es el sistema más adecuado si se le va a hacer un uso común, como por ejemplo, usar la Raspberry como un computador personal, navegar por la web, ver vídeos, etc. Es ligero a la vez que rápido, y tiene la ventaja de estar diseñado específicamente para esta máquina. Se pueden usar muchos otros si la arquitectura lo permite, pero por estar altamente ligado a nuestro *hardware*, usaremos este. En la página oficial de Raspberry, hay tres variantes del mismo sistema operativo, además de software adicional para facilitar la instalación inicial en nuestra máquina.

- **Raspberry Pi OS with desktop and recommended software:** Contiene el núcleo principal del sistema, entorno de escritorio y software recomendado. Ocupa algo menos de 3GB
- **Raspberry Pi OS with desktop:** Contiene el núcleo principal del sistema y entorno de escritorio. Ocupa entorno a los 2GB.

- **Raspberry Pi OS Lite:** Solamente contiene el núcleo del sistema accesible mediante línea de comandos. Su capacidad no llega a los 500MB.

Por ser más ligero que el resto, y porque no haremos uso de la interfaz gráfica, predeterminada de Raspbian para nuestro proyecto, usaremos el último. Finalmente, para convertir nuestra Raspberry Pi en una estación de juego funcional, usaremos RetroPie.

4.1.2. Software principal

RetroPie es un programa o conjunto de programas que entre otras cosas permite emular una gran variedad de consolas, desde las más antiguas como *Atari*, hasta las más actuales como *PlayStation 1*, *PlayStation 2* o *PlayStation Portable*, según qué modelo de Raspberry usemos. El software base trae una serie de emuladores con los que se podrá emular la mayoría de sistemas, pero para usos más específicos haría falta instalar las librerías necesarias. Se puede montar encima de diversos sistemas operativos, pero para nuestro proyecto lo montaremos encima de *Raspbian*. Además de las funcionalidades principales, se puede cambiar la configuración de los controles, aumentar o disminuir la memoria de vídeo y diversos parámetros relativos a la pantalla y al procesamiento de imagen, para aumentar el rendimiento de los emuladores, sobretodo de los que emulan consolas más actuales que requieren de una alta carga de procesamiento.

Figura 4.1: Pantalla principal de RetroPie con diversos emuladores

El principal uso que le daremos a RetroPie será el de emular nuestros juegos, como ya hemos dicho, pero además de esto, RetroPie nos facilitará la creación de registros (o *logs*) del inicio de nuestros juegos. Al lanzar un juego, este puede ejecutar una serie de scripts tanto al inicio como al final del mismo. Nuestra funcionalidad parte de esta característica puesto que sin ella nos sería imposible saber el tiempo de inicio y fin de un juego. Esta funcionalidad se describe de forma más detallada en la página oficial de RetroPie[1]. Lo que hará ese *script* de inicio y final será escribir en un fichero de texto la hora de inicio de la sesión con una serie de parámetros que vemos a continuación. El *script* de finalización hará lo mismo pero con un identificador que diferenciará entre el inicio y el final de la sesión de juego. Los argumentos que le podremos pasar al *script* serán los siguientes:

- **$1:** El sistema sobre el que se está corriendo el juego.
- **$2:** El emulador que se está usando.
- **$3:** La ruta al archivo del juego.

[1] https://retropie.org.uk/docs/Runcommand/

- **$4:** El comando que se usa para lanzar el juego por consola de comandos.

En el apartado 6.2 se detalla la configuración de nuestro *software* con los archivos que hemos de modificar para hacer funcionar la característica de *runcommand*.

El emulador que vamos a usar para nuestras pruebas será *MAME(Multiple Arcade Machine Emulator)*. Como su nombre indica, es un emulador de máquinas arcade incluido en el software de RetroPie que tiene el objetivo principal de unificar cada una de las máquinas arcade en un único lugar.

El problema principal que enfrentó MAME en los inicios de su desarrollo fue la diversidad de *hardware* en cada una de las máquinas. Como cada juego estaba desarrollado para una placa específica la tarea de emular cada placa distinta era muy costosa por lo que surgió *MAME*. Por ello, para lograrlo lo que sus desarrolladores hicieron fue emular cada componente por separado. Sabiendo cada componente de las placas de los distintos juegos y cómo se conectan entre ellos, se consiguió que la tarea de emular estos juegos arcade fuera más sencilla.

4.1.3. Herramientas de persistencia

Además del gestor de base de datos que vamos a usar, necesitaremos almacenar la base de datos en algún lugar. Como más adelante veremos, nuestro sistema tendrá una arquitectura cliente-servidor típica, donde nuestra Raspberry será el cliente y se comunicará con nuestra base de datos que será el servidor. Para ello haremos uso de otro equipo donde instalaremos una máquina virtual y en ella todos lo elementos necesarios para levantar la base de datos y que nuestra máquina arcade se pueda comunicar. En un futuro, en lugar de hacer uso de una máquina virtual, se podría usar de servidor otra Raspberry Pi o un equipo más potente, pero para nuestra prueba de concepto instalaremos todo en una máquina virtual. Para el sistema operativo, usaremos Ubuntu por ser ampliamente conocido y por haberlo manejado en el grado. En el apartado 6 veremos cómo llevar a cabo la instalación de nuestros componentes.

A este tipo de organización de software se la conoce como *LAMP (Linux + Apache + MySql + PHP)*. Es ampliamente usado en el desarrollo de aplicaciones web por contener el software base que se necesita. Ya hemos hablado de nuestro sistema operativo basado en Linux y nuestra base de datos que tendrá MariaDb, basado en MySql. Para conectar ambas partes usaremos un servidor apache instalado en nuestra máquina virtual y PhpMyAdmin que hará de *front-end* para nuestra base de datos. De esta forma podremos consultar de una forma sencilla todo lo referente a nuestra base de datos, incluso desde un equipo externo al de nuestra máquina virtual[2].

4.2 Arquitectura

Nuestro sistema se divide en dos grandes bloques altamente relacionados entre sí. La base de datos, instalada en un sistema externo y la Raspberry Pi, integrada en nuestra máquina arcade, que contendrá la lógica de nuestra aplicación, encargada de mostrar de una forma más amigable los datos procesados. Además, tendremos un módulo RFID conectado mediante pines I/O que se encargará de identificar al usuario que ha jugado. Cabe destacar, que nuestro sistema está pensado para poder añadir más máquinas basadas en una Raspberry Pi en un futuro si quisiéramos, creando así una red de máquinas arcade que podrían compartir estadísticas sobre cada juego instalado, transferir ROMs

[2] https://randomnerdtutorials.com/raspberry-pi-apache-mysql-php-lamp-server/

de juegos entre sí, en tiempo real mediante FTP(File Transfer Protocol), enlazar partidas entre jugadores para juegos multijugador, etc.

4.3 Diseño detallado interno

4.3.1. Diseño de la base de datos

En nuestra máquina, queremos persistencia para almacenar los datos de cada juego generado por el jugador. El principal uso que le daremos será el de almacenar las puntuaciones, los datos generales de cada juego, así como los datos de la máquina, con posterior expansión a varias de ellas, y los datos del jugador que se registre en nuestro sistema.

La base de datos es la que se encarga de almacenar los datos que nuestra Rasperry Pi le envía referente a los juegos que un usuario ha estado jugando. Tendría una arquitectura clásica cliente-servidor, ya que nuestra Raspberry sería la encargada de mandar una petición de guardado de los datos de juego a nuestro servidor. Si más adelante se añadiesen más máquinas, cada sistema tendría una copia local de lo que se ha hecho en su equipo. El sistema de base de datos sería el único que sabría lo que se ha hecho en cada una de las distintas máquinas.

Diseñaremos las tablas de cada objeto, además de las relaciones entre cada tabla. Para ello, necesitamos abstraer la información de la que se va a sacar partido y diferenciarla en clases, para así poder plasmarla en las tablas relacionales de la base de datos. Así pues, el diseño lógico de nuestra BDD quedaría de la siguiente forma:

- **Usuario**: Esta tabla contendrá la información relativa al jugador que usa el sistema.
  - **id:** Identifica al jugador.
  - **nombre:** Nombre del jugador.
  - **apellidos:** Primer y segundo apellido del jugador.
  - **id_nfc:** Sirve para identificar la tarjeta *NFC* o *RFID* con la que se identificará el jugador cuando inicie cada sesión de juego. El usuario se podrá identificar únicamente con una tarjeta.
- **Juego:** Con esta tabla, podremos guardar la información relativa a cada juego.
  - **id_juego:** Identifica el juego.
  - **nombre:** Nombre del juego.
  - **anyo_lanzamiento:** Año en el que se lanzó al mercado el juego en cuestión.
  - **genero:** Género en el que el juego está clasificado.
  - **publisher:** Compañía que publicó originalmente el juego.
  - **ruta:** Ruta absoluta de la localización del juego en nuestro sistema.
- **Sesion:** Relaciona el jugador y el juego mediante el tiempo jugado y su puntuación si la hubiera.
  - **id_s:** Identificador de la sesión.
  - **id_u:** Identificador del jugador que se relaciona con la tabla **usuario**.
  - **id_j:** Identificador del juego que se relaciona con la tabla **juego**.
  - **fechaInicio:** Fecha y hora en la que se ha empezado a jugar el juego por el jugador.

- **tiempo:** Tiempo de juego transcurrido desde que el jugador ha iniciado el juego hasta que lo ha cerrado.
- **score:** Puntuación obtenida en el juego si este lo permitiese.

Con todo esto, tenemos las tablas necesarias para empezar a guardar nuestra información que extraigamos de la lógica de la aplicación.

4.3.2. Diseño de la lógica

La lógica de nuestra aplicación se encargará de los siguiente. Por una parte tendrá que detectar cuando un juego ha sido iniciado y cuando se ha cerrado, procesando el tiempo de inicio y el de fin para ser enviado a la base de datos. Además, deberá detectar si hay una tarjeta *NFC* que identifica al jugador que lo ha iniciado o si por el contrario no la hay y el jugador es genérico. Por último, extraerá la información correspondiente a la puntuación de ese juego y la mandará también a la base de datos. Nuestras clases serán desarrolladas en Python por ser un lenguaje muy versátil y fácil de aprender, además de venir por defecto en todas las versiones de Raspbian. Cada una de las clases tiene una funcionalidad específica que queda de la siguiente forma:

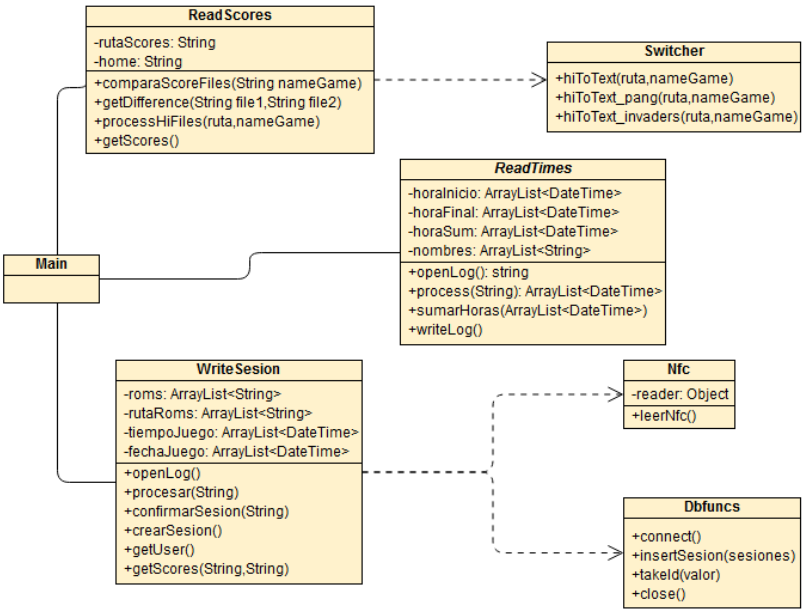


Figura 4.2: Diagrama de clases.

- **main.py:** Será la clase principal encargada de dirigir todas las operaciones desde que un usuario inicia un juego, con o sin tarjeta *RFID* hasta que el usuario cierra el juego y manda la información de la sesión a la base de datos.
- **nfc.py:** Esta clase se encargará de detectar si una tarjeta *RFID* está próxima al lector de tarjetas y de mandar su **id** a la clase principal.
- **readscores.py:** Será la encargada de leer las puntuaciones del juego. Puesto que cada juego almacena su información de forma distinta, esta clase creará una instancia de la clase *switcher* para procesar de una forma u otra las puntuaciones de cada tipo.

- **switcher.py:** El objetivo de esta clase, será el de contener un método por cada tipo de juego del que se desee extraer información. Contendrá una estructura de casos que variará la salida según qué parámetros se le pase desde la clase *readscores.py*. Si se desea procesar las puntuaciones de otro juego, es en esta clase donde habrá que añadir los métodos necesarios para analizar los distintos formatos de puntuación de cada juego.

- **readtimes.py:** Esta clase va a ser común independientemente del juego. Gracias a RetroPie, podemos ejecutar un script al iniciar y al finalizar un juego. Al iniciar el juego, escribiremos en un archivo de texto el momento en el que fue iniciado, además del nombre del juego. Al finalizar el juego se escribirá lo mismo pero con el momento en el que el juego finalizó. La función de esta clase será procesar este archivo, para calcular cuándo un juego fue iniciado y cuánto tiempo estuvo en ejecución.

- **writesesion.py:** Con esta clase leeremos el archivo de texto en el que se ha escrito toda la información de la sesión de juego, y escribiremos esta información en la base de datos haciendo uso de la clase *dbfuncs.py*. Hará uso también de la clase *nfc.py* para detectar la tarjeta de identificación y consultar el usuario correspondiente a ella.

- **dbfuncs.py:** Se encargará de crear una capa de abstracción para las funciones de la base de datos. Contendrá las funciones que serán usadas por la clase *writesesion.py* para insertar la información relativa a la sesión en la base de datos.

4.3.3. Componente RFID/NFC

Por último, para que un usuario se pueda identificar en cada sesión de juego, añadiremos un módulo de radiofrecuencia o *RFID(Radio Frequency Identification)*, que leerá una tarjeta compatible con el mismo. Usaremos un módulo que admita conexiones mediantes pines de entrada/salida. Este tipo de conexiones, son bastante frecuentes en los dispositivos de los conocidos Arduino[3] y similares. Usaremos el módulo *RFID RC522* por su precio y su sencillez de montaje. En la página web *PyMyLifeUp*[4], se muestra cómo realizar las conexiones entre el módulo y la Raspberry Pi. Cabe destacar que este módulo en concreto puede actuar también como etiqueta *NFC* con lo que podríamos leerlo mediante un dispositivo móvil compatible aumentando así las posibilidades de una aplicación móvil en un futuro.

[3] https://www.arduino.cc/
[4] https://pimylifeup.com/raspberry-pi-rfid-rc522/

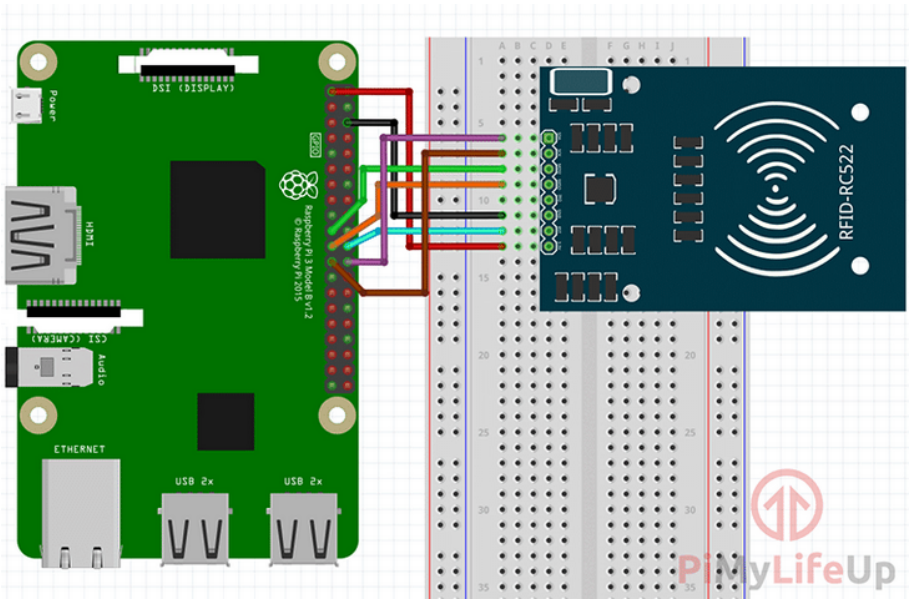


Figura 4.3: Conexión entre Raspberry PI y módulo de radiofrecuencia.

La diferencia principal entre un módulo *NFC* y un módulo *RFID* radica en que el módulo *NFC* puede actuar tanto de lector como de transmisor[5]. Estos se localizan en la mayoría de dispositivos móviles actuales y usan ondas electromagnéticas de alta frecuencia para sus transmisiones que se dividen en tres tipos según el rango de distancia:

- **Baja frecuencia(LF):** 125-134KHz. Para distancias largas hasta 25 metros.
- **Alta frecuencia(HF):** 13.56MHz. Hasta 1 metro.
- **Ultra Alta frecuencia:** 856MHz a 960MHz.

Tanto *RFID* como *NFC* operan en el rango de Alta frecuencia bajo el estándar de la ISO/IEC 14443 y parte de la ISO/IEC 18092. En la documentación oficial del módulo, se puede observar la relación entre lector y la Raspberry Pi.[3][4]

La lógica asociada al módulo radiofrecuencia será la descrita en la clase «nfc.py». Antes de iniciar un juego o durante el transcurso de la sesión, el usuario podrá dejar la tarjeta de identificación próxima al lector para que este identifique al usuario una vez termine de jugar. Cuando este finalice, la clase correspondiente al módulo hará uso de «dbfuncs.py» para recuperar de nuestra base de datos el usuario correspondiente a la tarjeta *RFID/NFC*, crear la sesión de ese usuario mediante la clase «writesesion.py» y escribirla en la base de datos.

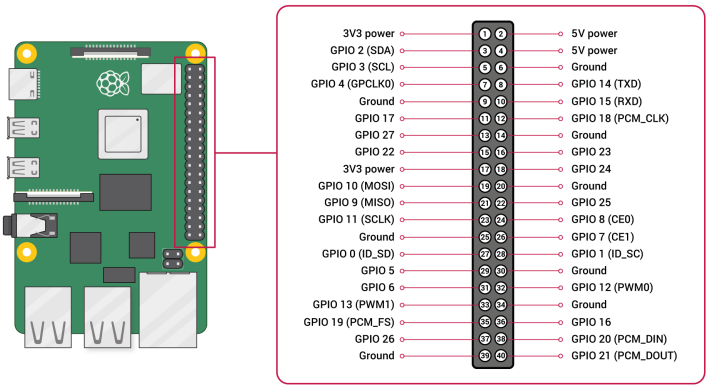
4.4 Diseño detallado externo

Lo siguiente que tendremos que ver será qué tipo de externo queremos para plasmarlo posteriormente en madera. En el capítulo de análisis hemos visto que nuestro tamaño ideal sería una máquina *bar-top* por su fácil movilidad y su tamaño. En la web existen multitud de diseños que podemos escoger gratuitamente. Entre los muchos que existen, *TheGeekPub*[6] contiene una excelente guía con distintos tipos de diseño altamente detallados. Este en específico no es gratuito pero el precio del diseño es ínfimo comparado

[5] https://cursos.mcielectronics.cl/2019/06/18/rfid-vs-nfc-cual-es-la-diferencia/
[6] https://www.thegeekpub.com/member-only-content/bartop-arcade-plans-platinum-edition-millimeters/

RFID-RC522 Module	Arduino Uno
1 - SDA	Digital 10
2 - SCK	Digital 13
3 - MOSI	Digital 11
4 - MISO	Digital 12
5 - IRQ	--unconnected--
6 - GND	Gnd
7 - RST	Digital 5
8 - 3.3V	3.3v

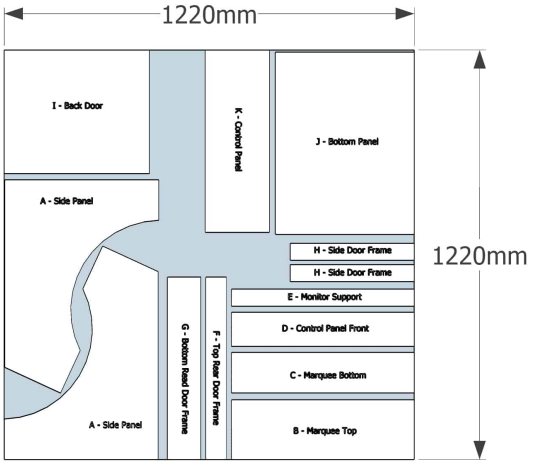
(a) Pines del módulo radiofrecuencia



(b) Pines entrada/salida Raspberry Pi 4.

Figura 4.4: Comparación de los diferentes tipos de pines.

con la cantidad de planos y guías que se obtiene cuando se compra. Entre otras cosas, obtenemos un diagrama con el tamaño de todas las partes a cortar.



(a) Diagrama del diseño

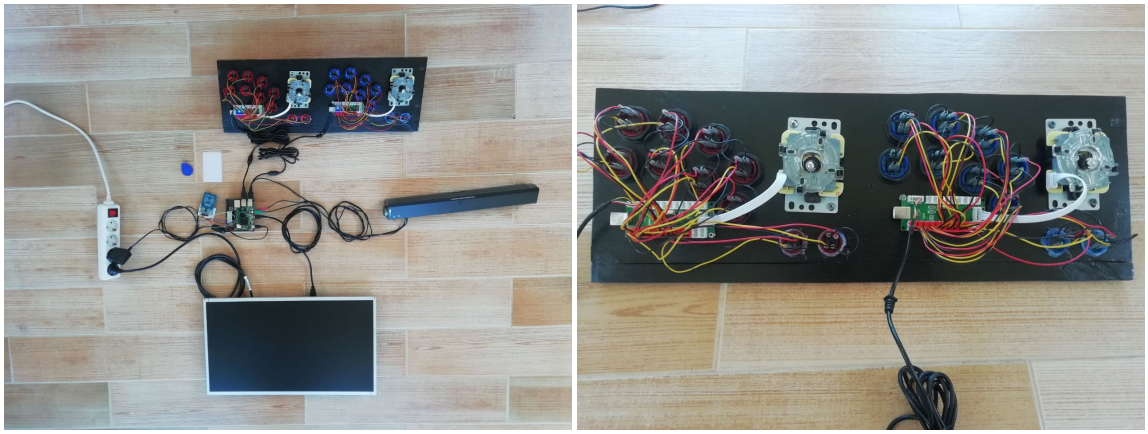
Part ID	Part Description	Material	Qty	Length	Width	Thickness
A	Side Panels	MDF or Plywood	2	457mm	636mm	19mm
B	Marquee Top	MDF or Plywood	1	543mm	185mm	19mm
C	Marquee Bottom	MDF or Plywood	1	543mm	121mm	19mm
D	Control Panel Front	MDF or Plywood	1	543mm	104mm	19mm
E	Monitor Support	MDF or Plywood	1	543mm	55mm	19mm
F	Top Rear Door Frame	MDF or Plywood	1	543mm	67mm	19mm
G	Bottom Rear Door Frame	MDF or Plywood	1	543mm	114mm	19mm
H	Side Door Frame	MDF or Plywood	2	368mm	51mm	19mm
I	Back Door	MDF or Plywood	1	365mm	438mm	19mm
J	Bottom Panel	MDF or Plywood	1	543mm	413mm	19mm
K	Control Panel	MDF or Plywood	1	543mm	171mm	19mm
J	VESA Mount (Optional)	MDF or Plywood	1	543mm	127mm	19mm

(b) Medidas del diseño

Figura 4.5: Componentes del diseño de la máquina.

A la izquierda vemos el diseño de cada una de las partes y en la derecha se detallan sus medidas. Cabe destacar que nuestro tablón base no tiene las mismas dimensiones con lo que se desperdiciará un poco más de madera, pero las medidas finales serán las mismas.

Otra parte a tener en cuenta, será el cableado interno que llevará la máquina. En el Apéndice A se detalla la instalación del mismo. Las conexiones realizadas antes de instalarlas en la máquina quedarían de la siguiente forma:



(a) Conexiones con Raspberry **(b)** Conexión de los Joystick

Figura 4.6: Conexiones de la máquina.

CAPÍTULO 5
Desarrollo de la solución propuesta: Pang

Llegados a punto, comenzamos con la elaboración del código necesario para suplir las necesidades de nuestro proyecto. A lo largo de la misma, nos hemos ido encontrando con dificultades que han causado que se modifique un poco la idea que se tenía sobre el funcionamiento de las clases a usar. A modo de ejemplo, nos centraremos en el juego *Pang(1989)* que hemos tomado de base para ir probando la funcionalidad a medida que avanzábamos en el código. Más adelante, desarrollaremos esta misma funcionalidad para otro videojuego de prueba y veremos los cambios que serían necesarios para futuros juegos en el caso de que quisiésemos adaptar la funcionalidad.

Uno de los problemas que nos encontramos a la hora de desarrollar la funcionalidad para este juego en concreto es la forma en la que se guardan las puntuaciones en el sistema. Como el juego fue diseñado originalmente para una máquina arcade, para que RetroPie y más específicamente *MAME(Multiple Arcade Machine Emulator)*[1] lo interprete correctamente, las puntuaciones se han de guardar en formato «*.hi», que es el formato que usa MAME para almacenar las puntuaciones de los diferentes juegos arcade. Aquí es donde se almacena la puntuación obtenida por los diez mejores jugadores, para este tipo de juego en concreto. Más adelante veremos que para otro tipo de juego solamente se almacena la máxima puntuación. Pues bien, este archivo tiene un formato específico que no se puede leer con un editor de textos normal y para poder extraer la puntuación del jugador que acaba de jugar hemos de realizar una serie de pasos específicos. Esta funcionalidad está distribuida entre las clases «readscores.py», «switcher.py» y «writesesion.py».

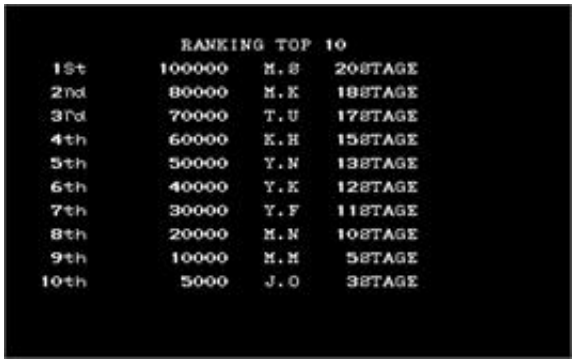


Figura 5.1: Puntuaciones del juego Pang.

[1] https://www.mamedev.org/

En la figura 5.1 observamos que el formato de puntuaciones tiene una estructura de diez líneas en las que cada línea almacena cuatro componentes: La clasificación, la puntuación, el acrónimo y la pantalla a la que se ha llegado. Cada uno de esos componentes tiene una codificación binaria distinta que para pasarlo a un formato legible tendremos que averiguar. En la imagen siguiente se muestra el funcionamiento de esta parte.

Hasta que se llegó a la solución final, se tuvo que realizar mucho ensayo a base de prueba y error para dar con la decodificación correcta. Dado que es un juego de hace más de 30 años, no se encontró documentación relativa al formato de guardado de las puntuaciones y por eso se tuvo que hacer ingeniería inversa con el fichero. No fue así para el juego de *Space Invaders* que muestra la única puntuación en texto plano y fue mucho más fácil de decodificar.

```
def hiToText_pang(self,ruta,nombre):
    fw = open(home+rutaStats+nombre+".txt","w+")
    with open(ruta+rutaScoresMame2003+nombre+".hi","rb") as f:
        byte = "empty"
        while byte:
            scores = []
            byte = f.read(3) #Leemos 3 bytes para la puntuación
            score = str(int(byte.hex())*10) #Formateamos los 3 bytes a string
            scores.append(score)

            byte = f.read(3) #Leemos 3 bytes para el acrónimo.
            name = str(byte,"latin1") #Los pasamos a string con codificación Latin1
            scores.append(name)

            byte = f.read(1) #Leemos 1 byte para la pantalla(stage)
            scores.append(byte.hex())

            nothing = f.read(7) # 7 bytes vacios
            byte = f.read(1).hex() # 1 byte con la posición en la tabla.
            if byte == '':  byte = "10"
            scores.append(byte)

            line = c.join(scores) # Cada linea contiene 16 bytes
            fw.write(line+jump)

            byte = f.read(1)
    fw.close()
```

Figura 5.2: Decodificación de puntuaciones Pang.

En la figura 5.2 vemos cómo para decodificar cada uno de los bloques de una sola línea usamos un formato distinto. Para decodificar la puntuación hemos de pasarlo a hexadecimal mientras que para el acrónimo tenemos que decodificarlo mediante el formato de codificación de caracteres «*latin1*». Además, cada uno de estos bloques ocupa tres *bytes* a diferencia de la pantalla o la posición que ocupan uno.

```
def hiToText_invaders(self,ruta,nombre):
    fw = open(home+rutaStats+nombre+".txt","w+")
    with open(ruta+rutaScoresMame2003+nombre+".hi","rb") as f:
        byte = f.read(1)
        low = byte.hex()
        byte = f.read(1)
        hi = byte.hex()

        score = str(hi)+str(low)
        fw.write(score)
        print(score)
    fw.close()
```

Figura 5.3: Decodificación de puntuaciones Space Invaders.

Por último vemos cómo se llama a la clase «switcher.py» desde «readscores.py» que se encarga de identificar el juego y usar el método de la clase «switcher.py» correspondiente. Esto se hace no de forma trival desde «readscores.py».

```python
def processHiFiles(self,ruta,nameGame):
    sw = Switcher()
    sw.hiToText(ruta,nameGame)

def getScores(self,ruta,rom):
    nameGame = rom.rstrip('.zip')
    self.processHiFiles(ruta,nameGame)
    return self.compareScoreFiles(nameGame)
```

(a) Readscores.py

```python
def hiToText(self,ruta,nombre):
    method_name = 'hiToText_' + str(nombre)
    method = getattr(self, method_name)
    return method(ruta,nombre)
```

(b) Switcher.py

Figura 5.4: Llamada a los métodos de procesamiento de puntuaciones.

La lógica principal radica en el método «*hiToText*» de «switcher.py» que llama a los métodos correspondientes de su clase según que juego sea el que hay que analizar. Si por ejemplo desde «readscores.py» se tiene que procesar las puntuaciones que refieren a *invaders*, este devolverá una función que procesará las puntuaciones del juego *Space Invaders* llamada «*hiToText_invaders*». En el apartado 4.1.2 hemos hablado de los argumentos que se pueden usar con RetroPie a la hora de lanzar los *scripts* de inicio y final. Uno de ellos ($3) contiene la ruta de la ubicación del juego a usar. Si bien es cierto que puede haber más de un juego en la misma ruta, no habrá dos nombres de juegos iguales con lo que la clave primaria es el nombre del juego y por tanto, cada juego tendrá una método de procesamiento distinto, evitando así errores en el funcionamiento.

CAPÍTULO 6
Fase de implantación

En este apartado se especificará por pasos, cómo hemos implantado nuestra solución final. Se hablará de la instalación del *software* que nos ha hecho falta para nuestro proyecto y de los pasos a seguir para montar los componentes necesarios, tanto la parte *hardware* como *software*.

6.1 Instalación inicial

Lo primero que debemos instalar, serán los componentes *software* base. Sistema operativo y programas necesarios. Para instalar el sistema operativo, lo primero que debemos hacer es descargarlo desde la página oficial de Raspberry Pi[1] y *flashearlo* dentro de una tarjeta SD para que nuestra Raspberry lo reconozca. Para ello, usaremos el programa BalenaEtcher[2], de descarga gratuita. Será necesario un equipo con Windows para llevarlo a cabo. Otra forma más habitual para usuarios poco habituados a Linux, es mediante el

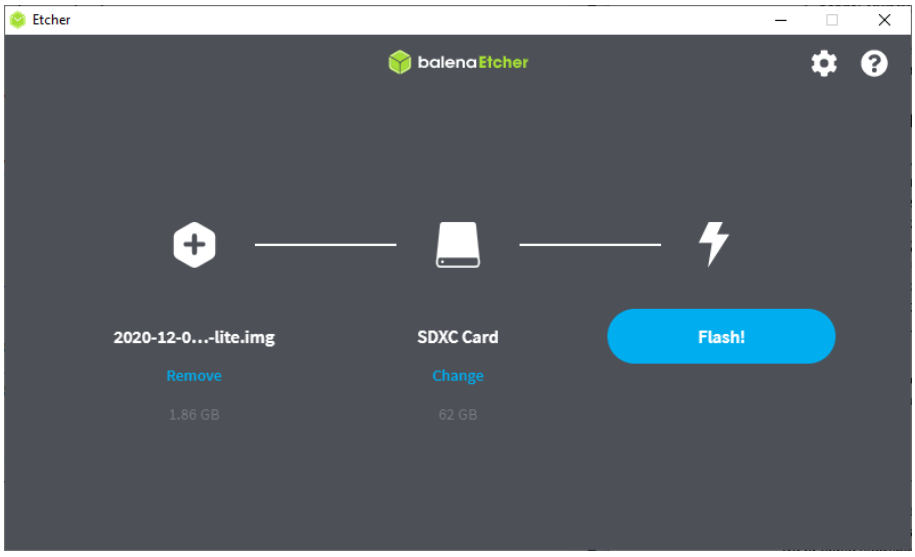


Figura 6.1: Flashear la tarjeta SD mediante BalenaEtcher

instalador NOOBS ubicado en la misma que el sistema operativo. De forma sencilla se podrá elegir uno o varios SO que introducir dentro de la tarjeta SD[3].

[1] https://www.raspberrypi.org/software/
[2] https://www.balena.io/etcher/
[3] https://www.raspberrypi.org/documentation/installation/noobs.md

Una vez instalado, conectamos un teclado y una pantalla a nuestra Raspberry Pi e iniciamos el sistema. El usuario y contraseña por defecto para el sistema son *pi* y *raspberry* respectivamente. Por seguridad, los cambiaremos más adelante. Lo primero que debemos hacer al iniciar será conectarnos a nuestra red y habilitar *ssh* para poder conectarnos remotamente y así trabajar de forma más sencilla. Después de acceder con nuestro usuario y contraseña haremos lo siguiente.

```
1 pi@raspberrypi: ~$ sudo raspi-config
```

Con esto, abriremos la interfaz de configuración que proporciona Raspbian, para poder configurar los aspectos básicos del sistema de forma sencilla.

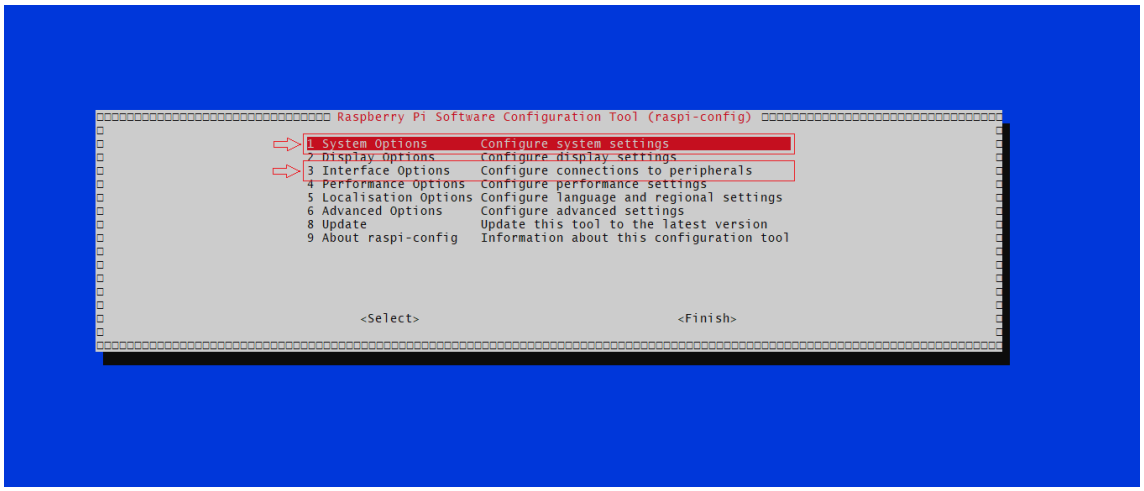


Figura 6.2: Interfaz de configuración de Raspbian.

En el primero, podremos configurar lo aspectos que nos harán falta para continuar la instalación del resto de componentes.

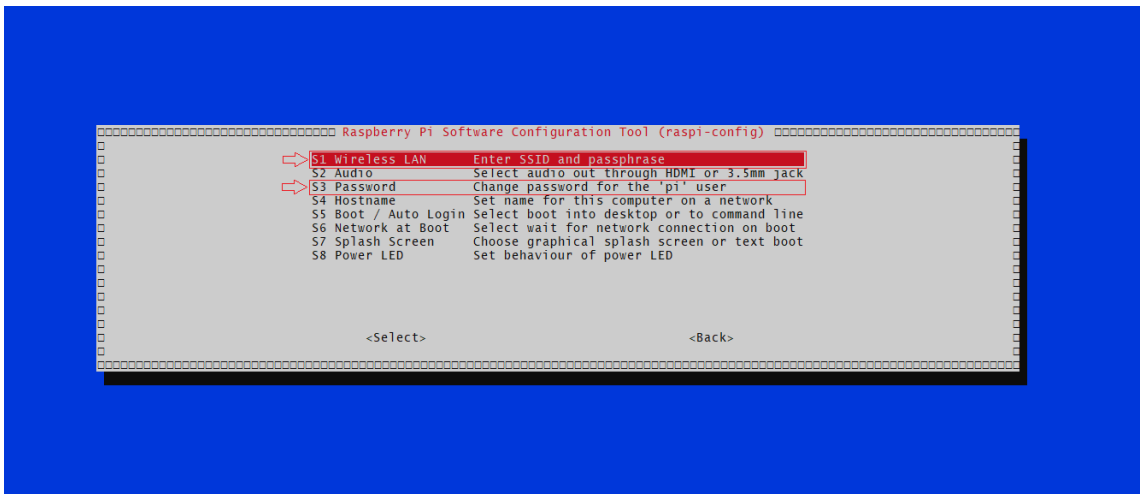


Figura 6.3: Opciones básicas del sistema.

Tendremos que configurar el WiFi si no disponemos de un cable ethernet al que conectar nuestra Raspberry Pi para realizar nuestra primera instalación. Por seguridad, cambiaremos también la contraseña del dispositivo. En el apartado de *Interface Options*, habilitaremos el protocolo *ssh* para trabajar más cómodamente desde otro dispositivo, además del inicio automático para que se inicie con el usuario predefinido después de un reinicio. A continuación, reiniciamos la máquina y nos disponemos a actualizar el sistema:

```
pi@raspberrypi:~ $ sudo apt update && sudo apt upgrade
```

Cuando finalicen, reiniciamos la máquina de nuevo y nos dispondremos a instalar RetroPie. Para instalar RetroPie, primero hemos de instalar *git*, que es de donde descargaremos los ficheros principales del sistema.

```
pi@raspberrypi:~ $ sudo apt install git lsb-release
pi@raspberrypi:~ $ git clone --depth=1 https://github.com/RetroPie/RetroPie-
    Setup.git
```

Finalmente, ejecutamos el script que instalará RetroPie.

```
pi@raspberrypi:~ $ cd RetroPie-Setup
pi@raspberrypi:~/RetroPie-Setup $ chmod +x retropie_setup.sh
pi@raspberrypi:~/RetroPie-Setup $ sudo ./retropie_setup.sh
```

A lo largo de la instalación, podremos elegir que tipo de instalación queremos. Como nuestra instalación será la primera, tendremos que seleccionar *Basic Install*.

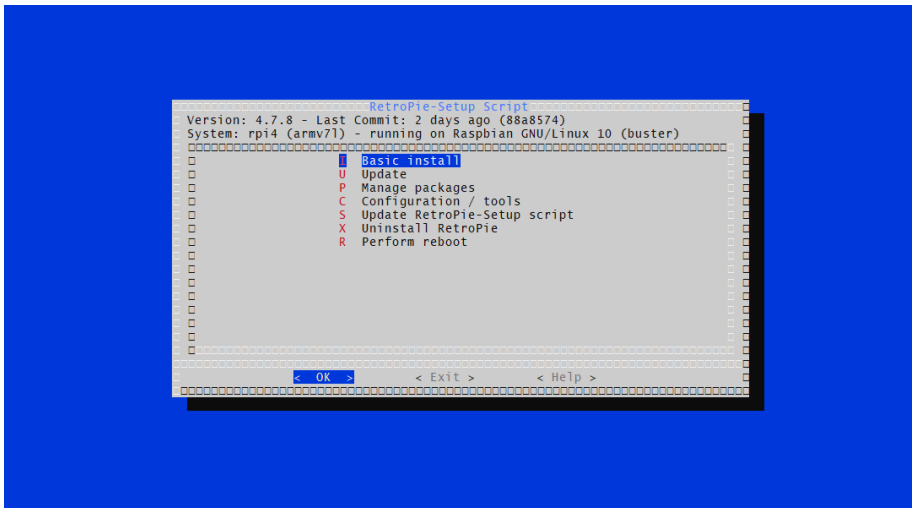


Figura 6.4: Instalación RetroPie

Cuando finalice la instalación, tendremos que habilitar el inicio automático de RetroPie para no tener que introducir credenciales al reiniciar el sistema. Con esto, habremos finalizado la instalación inicial de RetroPie.

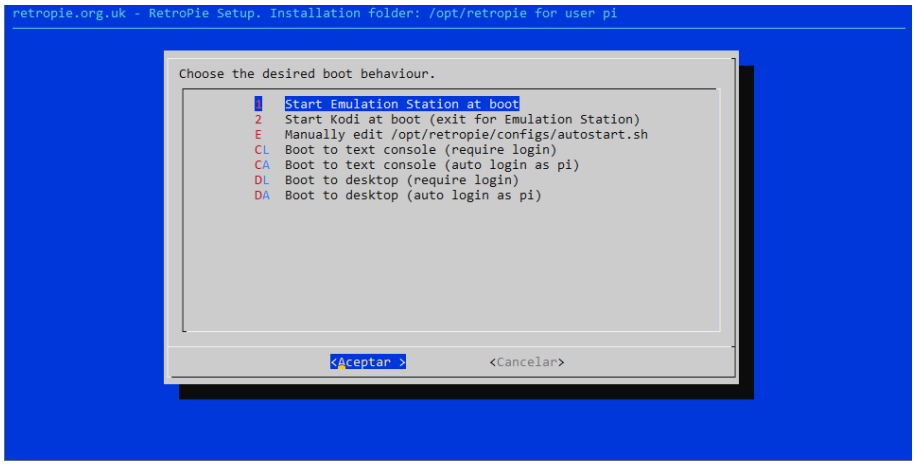


Figura 6.5: Inicio automático RetroPie

6.2 Instalación de nuestro software

Para la instalación de nuestro software deberemos tener al menos una ROM de un juego en la máquina. Para nuestras pruebas, hemos desarrollado el software en base al fichero de puntuaciones de dos juegos. *Pang(1989)* y *Space Invaders*. Por tanto, tendremos que transferir estos dos juegos al sistema. Se puede hacer de dos formas, mediante línea de comandos o por *WinSCP*, un programa que te permite transferir archivos a otra máquina que esté en la misma red mediante el protocolo *ssh*. Por línea de comandos nos tendríamos que dirigir a la carpeta donde tenemos la ROM del juego y escribiríamos lo siguiente:

```
scp pang.zip pi@192.168.1.39:/home/pi/RetroPie/roms/mame-libretro/
```

De esta forma, transferiríamos el archivo del juego a la carpeta correspondiente del emulador. Al ser un juego Arcade, habría que transferirlo a la carpeta de MAME, un emulador para juegos de máquinas arcade. Mediante WinSCP se podría realizar lo mismo de forma gráfica.

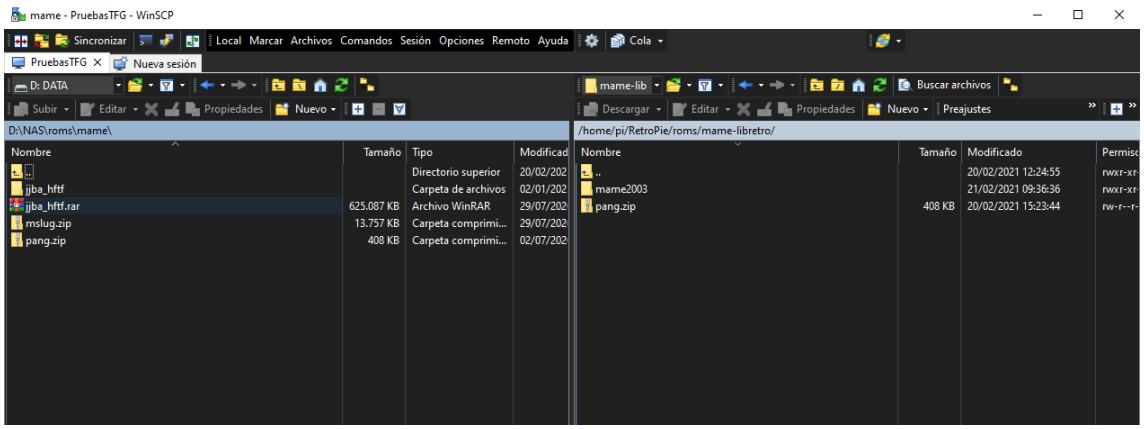


Figura 6.6: Uso de WinSCP

A continuación, modificaremos los archivos necesarios para el correcto funcionamiento de nuestro software. Para empezar, tendremos que modificar los scripts que se ejecutan al iniciar y al cerrar un juego, para que escriba en un fichero la fecha de inicio y de fin respectivamente, además de otras variables que nos harán falta. En el mismo script de finalización, lanzaremos nuestro programa.

```
pi@raspberrypi:~ $ sudo echo > /opt/retropie/configs/all/runcommand-onstart.sh
pi@raspberrypi:~ $ sudo echo > /opt/retropie/configs/all/runcommand-end.sh
```

Escribiendo en cada archivo lo siguiente, respectivamente.

```
echo $(date +"%H:%M:%S %d/%m/%Y")','start','$1','$2','$3','$4 >> ~/RetroStats/
    logs/game_stats.log
```

```
echo $(date +"%H:%M:%S %d/%m/%Y")','end','$1','$2','$3','$4 >> ~/RetroStats/
    logs/game_stats.log
python3 ~/RetroStats/main.py
```

Creamos un directorio en nuestra carpeta personal llamado *RetroStats* y transferimos ahí los archivos de nuestro código.

```
pi@raspberrypi:~ $ mkdir RetroStats
pi@raspberrypi:~ $ mkdir RetroStats/logs
pi@raspberrypi:~ $ mkdir RetroStats/scores
```

El siguiente paso para que todo funcione correctamente será descargar los módulos necesarios de *Python3* para nuestro código. Primero importaremos el módulo que permite controlar los pines *GPIO* de la *Raspberry* desde *Python*, y a continuación las librerías que nos harán falta para usar nuestro dispositivo RFID.

```
pi@raspberrypi:~ $ sudo apt-get install python-pip
pi@raspberrypi:~ $ sudo apt-get install python3-dev python3-rpi.gpio
pi@raspberrypi:~ $ sudo apt-get install picap
pi@raspberrypi:~ $ picap-setup
```

Y para el módulo RFID, previamente tendremos que haber habilitado *SPI kernel module* desde las opciones del comando *raspi-config*.

```
pi@raspberrypi:~ $ sudo pip3 install spidev
pi@raspberrypi:~ $ sudo pip3 install mfrc522
```

Además, para el conector con la base de datos importaremos el módulo siguiente.

```
pi@raspberrypi:~ $ sudo pip3 install mysql-connector-python
```

Con todo esto, nuestro programa funcionará correctamente la próxima vez que juguemos.

6.3 Configuración de la máquina virtual

Nuestro proyecto hará uso de un servidor que tendrá una base de datos instalada. Este servidor lo instalaremos en una máquina virtual dentro de nuestro operador personal para realizar las pruebas de nuestro proyecto. Más adelante, esta base de datos se podría exportar a otro sistema más permanente como por ejemplo otra Raspberry Pi o un equipo que dejemos permanentemente encendido.

Lo primero que tendremos que hacer será instalar el sistema operativo que vamos a usar para nuestra máquina virtual. Para ellos descargamos la imagen de Kubuntu, que es el sistema operativo que usaremos para nuestra máquina, y lo instalamos en VirtualBox.

A continuación, instalamos el resto de componentes de nuestro servidor LAMP: Linux, Apache, MySql y PHP. Ya tenemos Linux por tanto continuamos con Apache.

```
user@user-VirtualBox:~ $ sudo apt-get update
user@user-VirtualBox:~ $ sudo apt-get install mariadb-server mariadb-client
user@user-VirtualBox:~ $ sudo apt-get install apache2
user@user-VirtualBox:~ $ sudo nano /etc/apache2/mods-enabled/dir.conf
```

Y nos aseguramos que el fichero exista un componente PHP.

```
<IfModule mod_dir.c>
    DirectoryIndex index.html index.php index.htm
</IfModule>
```

Reiniciamos Apache e instalamos PHP.

```
user@user-VirtualBox:~ $ sudo systemctl restart apache2
user@user-VirtualBox:~ $ sudo apt-get install php php-cgi libapache2-mod-php
    php-common php-pear php-mbstring
```

Y Finalmente, instalamos phpMyAdmin para poder acceder desde el navegador a nuestra base de datos.

```
user@user-VirtualBox:~ $ sudo a2enconf php7.0-cgi
user@user-VirtualBox:~ $ sudo systemctl reload apache2
user@user-VirtualBox:~ $ sudo apt-get install phpmyadmin php-mbstring php-
    gettext
```

Con esto, habremos creado correctamente nuestra base de datos y la podremos acceder desde la misma máquina virtual mediante la dirección *http://localhost/phpmyadmin*. A continuación veremos qué cambios realizar para poder acceder a esta base de datos desde una red local.

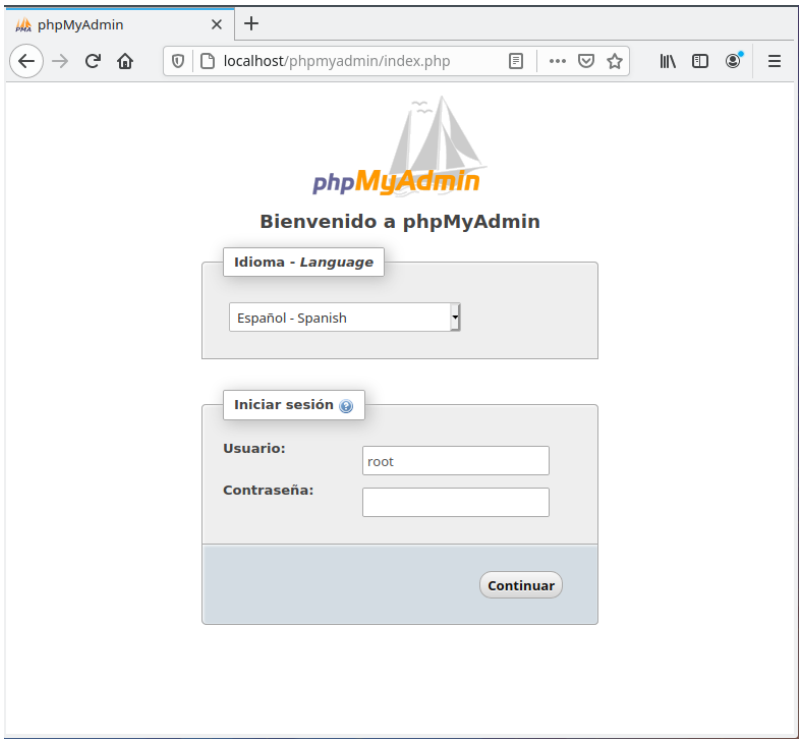


Figura 6.7: Acceso a phpMyAdmin

6.3.1. Configuración de red de la VM

A causa de que estamos creando nuestro servidor en una máquina virtual, tendremos que establecer la configuración de red de una forma específica para poder acceder, por ejemplo, desde nuestro ordenador anfitrión y así no tener que hacer uso de nuestra máquina virtual. Si en lugar de una máquina virtual fuese otro equipo independiente no haría falta realizar estas configuración ya que estaría conectado directamente a nuestra red local.

Cerramos nuestra máquina virtual y en la configuración de la misma, nos dirigimos a Red y creamos otra interfaz de red que corresponderá a la tarjeta de red de nuestro Host. Es decir, tendremos ahora dos interfaces de red en la máquina virtual. Una en modo NAT que nos permitirá conectarnos a la web y otra en modo puente (o *bridge*) que hará de enlace con nuestro ordenador principal. Finalmente, accedemos desde nuestro Host a la dirección *http://ipVm/phpmyadmin* siendo *ipVm* la dirección IP que se le asignará a nuestra máquina virtual una vez que conectemos el cable ethernet, en nuestro caso **192.168.1.86**. Con todo esto, ya tendríamos nuestro servidor LAMP totalmente configurado.

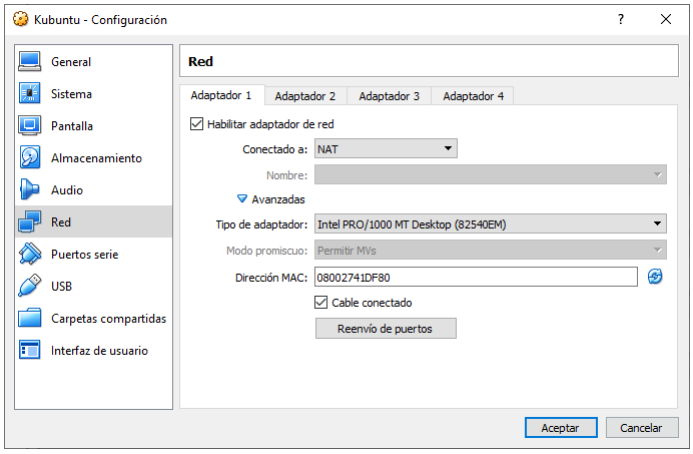


Figura 6.8: Adaptador de red 1

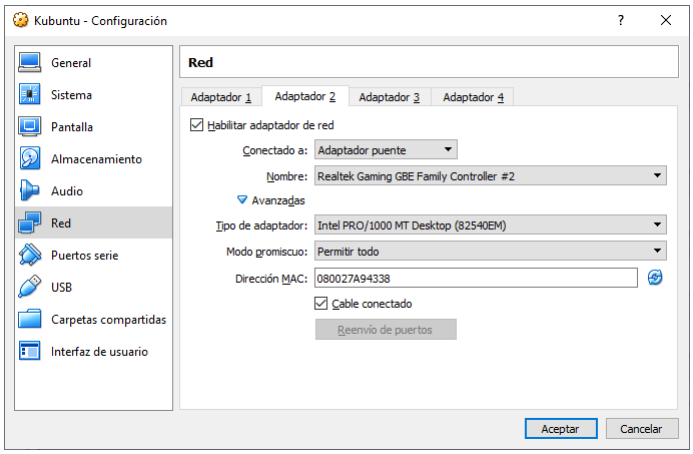


Figura 6.9: Adaptador de red 2

Por tanto las conexiones de nuestra máquina quedarían de esta forma:

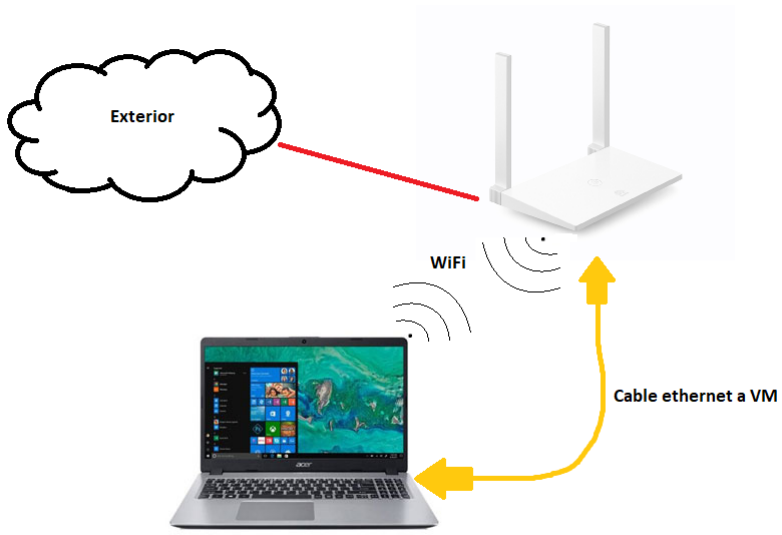


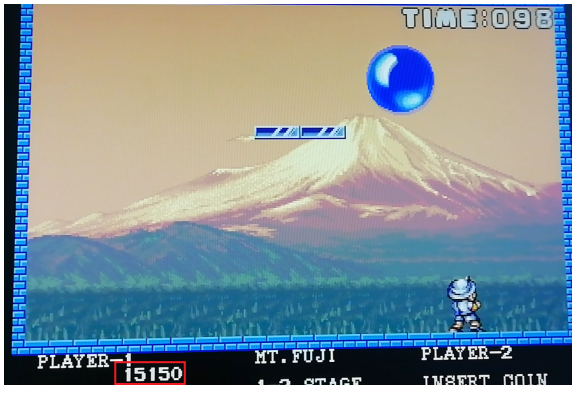
Figura 6.10: Conexiones a la máquina virtual

CAPÍTULO 7
Resultados

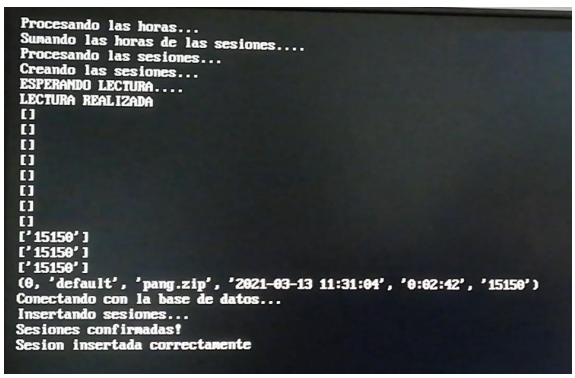
Por último, realizaremos una serie de pruebas en nuestra máquina para comprobar que la lógica de nuestro programa se ejecuta correctamente. Asimismo, veremos el acabado final del mueble.

En primer lugar, comprobaremos que nuestro programa se ejecuta como es debido al iniciar una sesión de juego. Para ello, tendremos que encender nuestra máquina virtual con nuestro servidor y realizar las conexiones oportunas. Cabe destacar que realizaremos las pruebas finales en el videojuego *Pang(1989)* que es sobre el que se ha realizado cada prueba a lo largo de todo el proyecto. La peculiaridad de este juego es que tan solo guarda las puntuaciones de los diez mejores jugadores siendo 5.000 la puntuación más baja al crear un nuevo archivo de juego guardado. Es por ello que para que nuestro programa funcione tendremos que superar esa puntuación mínima. En otros videojuegos como *Space Invaders* no hay mínimo inicial así que funcionaría siempre que superásemos la máxima puntuación.

Iniciamos el juego en cuestión y realizamos una sesión de juego hasta que perdamos. Al acabar, tendremos que cerrar el videojuego con la combinación de teclas correspondiente, previamente configuradas al conectar un nuevo mando a RetroPie. Para esta prueba, hemos usado un usuario por defecto, es decir, no nos hemos identificado con ninguna tarjeta. En las imágenes siguientes se muestra el funcionamiento.



(a) Puntuaciones al finalizar una sesión en Pang.



(b) Script de escritura de las puntuaciones.

En esta prueba, la puntuación antes de que el jugador pierda es de 15150. Posteriormente vemos como al cerrar el juego, se crea una sesión con un jugador por defecto (*default*) con una hora y una duración determinada, además de la puntuación que ha obtenido. Si nos dirigimos a nuestra base de datos comprobamos que efectivamente se ha insertado la sesión correspondiente.

Figura 7.2: Sesión con un usuario por defecto

Figura 7.3: Tabla de usuarios.

Más tarde, realizamos las mismas pruebas esta vez haciendo uso del dispositivo de identificación. En la imagen anterior podemos observar que se ha añadido una fila más por cada sesión de cada usuario distinto. Como prueba, la tabla de puntuaciones al finalizar las tres sesiones en el juego queda de la siguiente forma:

Figura 7.4: Tabla de puntuaciones al finalizar las sesiones.

Nuestro programa además solventa fallos de conexión con el servidor. Hemos visto que nuestra máquina hace uso de una conexión a internet para conectar con el servidor que contiene la base de datos y así recuperar el usuario que se identifica con la tarjeta *RFID* y guardar la sesión. En la práctica no todo es perfecto y el fallo más frecuente es que no se pueda conectar a la base de datos porque bien o la red o el servidor estén saturados y no se puedan recuperar los datos del usuario. De ser así, las sesiones se guardarían localmente y se podrían subir la próxima vez que hubiese conexión. Si esto ocurriera, la lógica de nuestro programa guardaría localmente un archivo de texto con el

identificador de la tarjeta *RFID* y la sesión que ha realizado. La próxima vez que cualquier usuario jugase se consultaría primero el usuario que tuvo problemas de conexión y luego el nuevo usuario.

Así mismo, en la imagen siguiente podemos ver el acabado de la máquina. En el Apéndice A incluimos más ilustraciones de la misma.

Figura 7.5: Acabado externo de la máquina.

CAPÍTULO 8
Conclusiones

En este trabajo hemos visto cómo diseñar e implementar una máquina arcade desde cero con la ayuda de tecnologías actuales. Además, hemos implementado una funcionalidad extra a la máquina que permite compartir puntuaciones de videojuegos a una base de datos externa. Con todo ello hemos completado exitosamente los objetivos expuestos en el primer apartado, encontrándonos sin embargo con dificultades por el camino que nos han hecho cambiar un poco el rumbo final del proyecto.

Para empezar, dado que el proyecto tenía una parte tanto física como lógica, los procesos de creación del mueble requirieron de algo más de tiempo del supuesto. Al principio se pensaba terminar en una semana a lo sumo pero se vio alargado a dos puesto que hubieron dificultades en el camino, como por ejemplo, la falta de experiencia en este tipo de proyectos, los errores cometidos en el proceso que hicieron que se tuviese que volver a repetir alguna parte, la inexactitud con los planos originales, el proceso de secado entre una capa de pintura y otra o el tiempo de espera para la llegada de algunos materiales. El resultado final quedó mejor acabado de lo que se esperaba en un principio pero menos que el modelo original ya que, pese a tener bastantes herramientas, no teníamos las suficientes para que tuviese un acabado realmente profesional.

Por otra parte, aunque la parte lógica no llevo mucho tiempo gracias al lenguaje elegido que era fácil de entender, sí que tuvimos algunas dificultades a la hora de procesar cada las puntuaciones de nuestro juego en cuestión. *Pang(1989)*, guarda las puntuaciones en formato hexadecimal pero pese a ello, tuvimos que hacer un proceso de ingeniería inversa para averiguar qué significaba cada uno de los bytes, pasándolo a binario primero. Además, cada uno de esos bytes en hexadecimal tenía una codificación distinta y tuvimos que hacer un proceso de prueba y error hasta dar con la buena. Por otra parte, nuestro código lo desarrollamos en nuestro ordenador principal y para probarlo teníamos que enviarlo a la máquina con lo que perdíamos algo más de tiempo que de hacerlo en la propia Raspberry.

Pese a todo esto, los conocimientos adquiridos han sido numerosos, enriqueciéndonos gratamente en diversos aspectos como la cultura de las máquinas arcade, la estructura de los sistemas basados en *Linux*, *Python* y sus cualidades, además de dotarnos de experiencia en trabajos de carpintería.

Consecuentemente, hemos cumplido nuestro objetivos planteados en el apartado 1.2. Hemos visto la situación actual de los trabajos relacionados con nuestro proyecto dotándonos de cultura relativa a los videojuegos arcade y sus máquinas. En añadido, mediante la instalación de un sistema basado en *Linux* que soportase la emulación de estos juegos, hemos indagado en la estructura de este y hemos aprendido mejor cómo se organiza internamente. Como resultado, hemos implementado un sistema de gestión de puntuaciones para videojuegos arcade además de la construcción de la máquina física.

En definitiva, este proyecto ha sido ampliamente gratificante por todas la facetas que hemos ido descubriendo, tanto externas del trabajo como internas de uno mismo. Además, el apoyo de nuestro tutor Jorge González han sido un pilar fundamental para el desarrollo del mismo.

8.1 Relación del trabajo con los estudios cursados

Sin duda, en el proyecto se han visto temas altamente relacionados con los estudios cursados en la Universidad Politècnica de València. Por esto mismo, cabe destacar diversas asignaturas cursadas a lo largo del grado que han tenido un gran impacto en este.

Las más destacadas por estar más próximas al tema del proyecto serían las cursadas en la rama de Tecnologías de la Información. Base de datos y Tecnología de base de datos han sido la base de nuestro trabajo. Con estas hemos podido comprender cómo funciona realmente un servidor con base de datos plasmándolo en nuestro esquema. También, mediante los conocimientos de Desarrollo Web, hemos podido levantar correctamente nuestro servidor y comprender cómo funciona el *front-end* del mismo. Con el proyecto de Sistemas de Redes Empresariales, incidimos más profundamente en un sistema basado en Linux.

Asimismo, han habido diversas asignaturas que nos han proporcionado una base destacable vista ahora. Fundamentos de los Sistemas Operativos, nos dió características de un sistema operativo como el usado además de nociones del lenguaje C, precursor de Python y de prácticamente todo lenguaje hoy en día. Ingeniería del Software, experiencia con los entornos de desarrollo y los modelos de capas de todo proyecto *software* que se precie.

Por último, es necesario mencionar todas aquellas que han afectado transversalmente y que han proporcionado matices clave como Matemática Discreta; Lenguajes, Tecnologías y Paradigmas de la programación; Introducción a la Programación; Programación o Estructuras de Datos y Algoritmos.

8.2 Trabajos futuros

A pesar de todo el esfuerzo, en el desarrollo del proyecto siempre aparecen nuevas ideas que no nos da tiempo a implementar ya sea por falta de tiempo o porque se escapan fuera del ámbito en el que estamos trabajando. Es por ello, que a continuación se nombra una serie de características o funcionalidades que se podrían implementar en un futuro.

- **Aplicación móvil:** Con la estructura de base de datos, damos pie a un proyecto futuro que pueda recuperar los datos recopilados por esta y mostrarlos de una forma amigable para el usuario. Se puede llevar la aplicación al extremo y hacer una aplicación que muestre estadísticas de todo tipo. Tiempo jugado diaria, semanal o mensualmente. Ranking de los mejores jugadores en los juegos que queramos, competiciones en la misma aplicación, chat entre usuarios que quieran competir en tiempo real, noticias sobre nuevas máquinas en la red,etc.

- **Red de pruebas:** En nuestro trabajo sólo ponemos en marcha una máquina recreativa pero el potencial de este no se puede ver realmente sin varias con la misma lógica de programa que funcionen en un entorno real y recopilen datos de usuarios diversos. Además, se puede aumentar el número de videojuegos que admitan esta funcionalidad, incrementando así las estadísticas de juego mostradas por la

aplicación móvil del anterior apartado. Por otra parte, se podría implementar una funcionalidad nueva en las máquinas y es que con esta topología de red, se podría llevar a cabo un tipo de cliente *BitTorrent* que compartiese roms entre las distintas máquinas. Si un juego es más jugado en una zona que en otra o si se ha añadido un juego nuevo a la red, estas podrían intercambiar automáticamente los archivos de juego y facilitar así la tarea del administrador.

- **Otros factores de forma:** Este trabajo ya sería prácticamente similar al nuestro con la diferencia del exterior. Hemos visto que en el mercado existen gran variedad de microcomputadores, algunos más pequeños que el nuestro como es el caso de la Raspberry Pi Zero. Esta se podría incluir por ejemplo no en una máquina arcade, si no en una consola de mano que pudiese conectarse a internet vía una tarjeta SIM y ofreciese la misma funcionalidad en cualquier lugar con cobertura o conexión *Wi-Fi*.
- **Aumento de seguridad:** La seguridad es importante el cualquier proyecto que manejen datos sensibles de usuarios. En nuestro caso no haría falta hacer uso de una seguridad extrema pero sí sería interesante a modo de aprendizaje para una posible segurización de una red de computadoras.
- **Modelo de negocio:** Visto todo esto, la idea de una red de máquinas arcade con una aplicación móvil se podría monetizar o al menos idear un modelo de negocio.

Bibliografía

[1] Diseño e implementación de una máquina Arcade con Raspberry Pi y Arduino. Consultado en https://riunet.upv.es/handle/10251/53962.

[2] Diseño de un nuevo modelo modular de máquina recreativa personalizable. Consultado en https://riunet.upv.es/handle/10251/157776?show=full

[3] MFRC522Standard performance MIFARE and NTAG frontend. Consultado en https://bit.ly/2NOotuB.

[4] RFID Quick Start Guide: Arduino. Consultado en https://bit.ly/3aFV7HY.

[5] Comparative analysis of power variables in high performance embedded and x86 architectures using GNU/Linux. Consultado en https://doi.org/10.1016/j.compeleceng.2011.04.012

[6] Arnold S. Berger Hardware and Computer Organization Chapter 10 - The Intel x86 Architecture. Consultado en https://doi.org/10.1016/B978-075067886-5/50034-3

APÉNDICE A
Construcción de la máquina

En esta apartado se detallarán los pasos realizados para la construcción de la máquina.

A.1 Creación de las piezas

En primer lugar, tuvimos que comprar los materiales necesarios detallados en el presupuesto del apartado 3.4. Una vez que teníamos los materiales preparados, tuvimos que plasmar el diseño en la madera de forma técnica para que las piezas que cortásemos posteriormente quedasen rectas. Partimos de nuestro diseño para la realización de nuestras piezas.

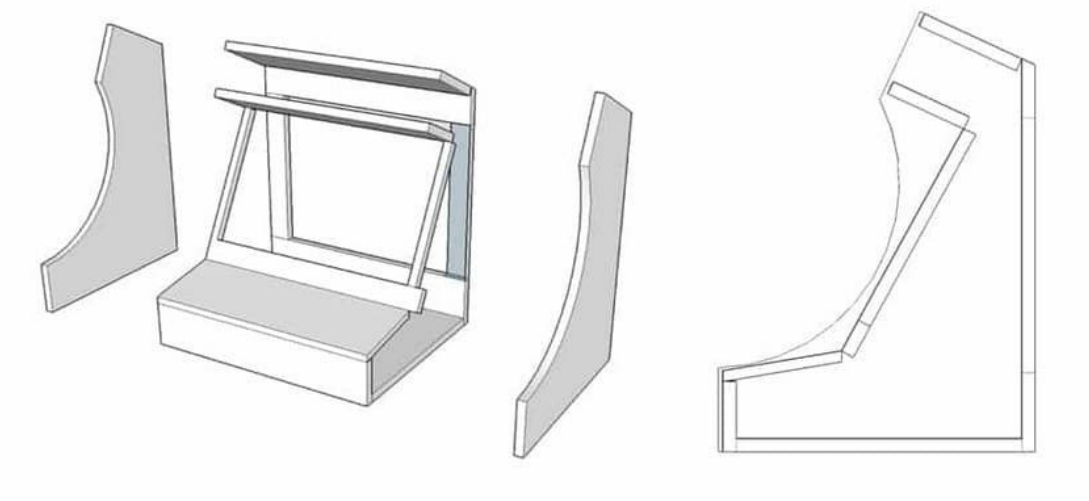


Figura A.1: Diseño externo de nuestra máquina

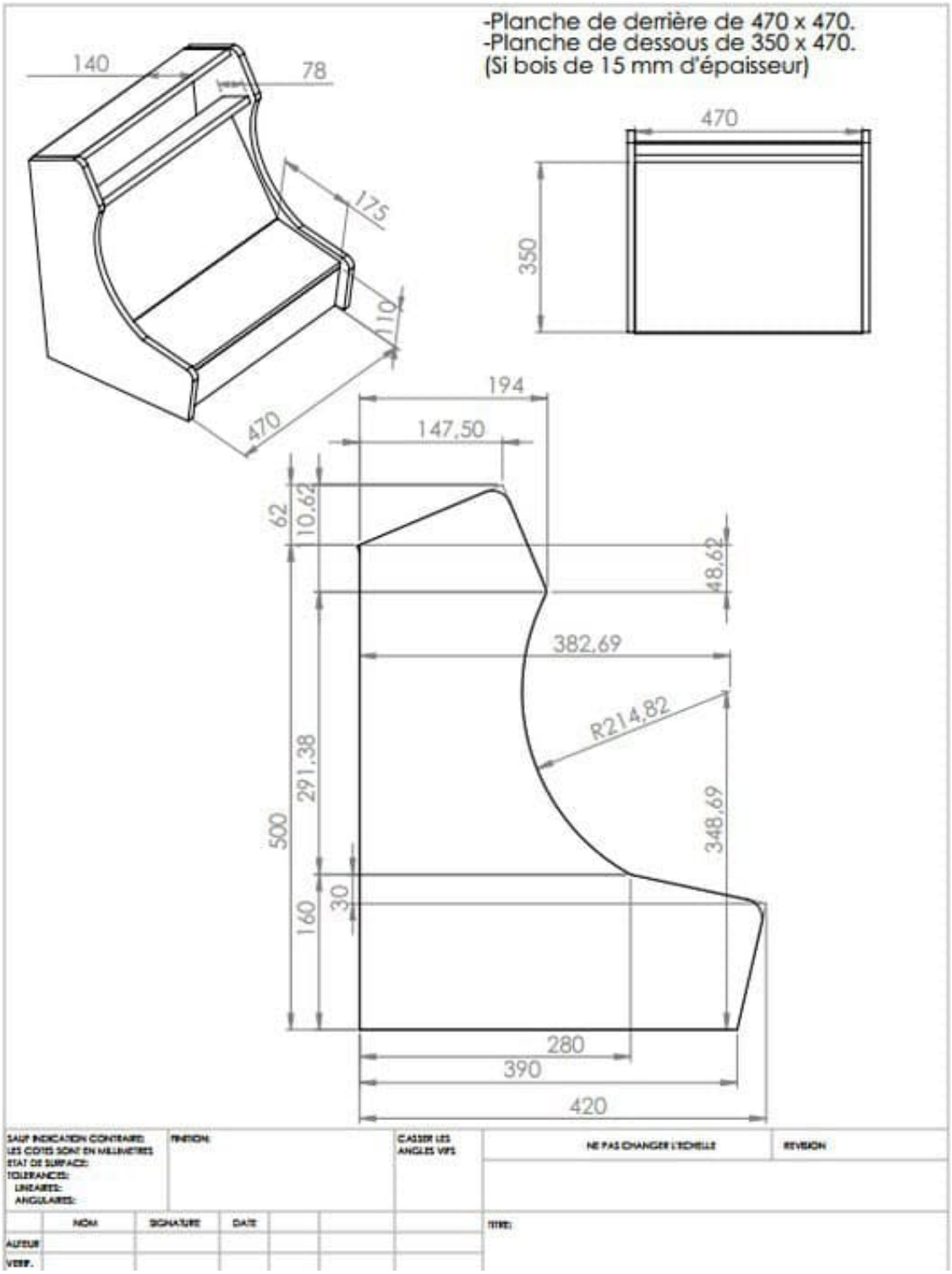


Figura A.2: Esquemática de nuestro diseño

Cortamos los componentes y montamos la estructura de madera base.

Figura A.3: Montaje de la estructura de madera

A continuación, cortamos los orificios para los controles.

Figura A.4: Controles

Le damos un acabado de pintura.

Figura A.5: Acabado de pintura

E introducimos una pantalla para realizar las pruebas oportunas.

Figura A.6: Resultados finales

A.2 Resultados finales