



ugr | Universidad
de **Granada**

TRABAJO FIN DE GRADO
INGENIERÍA EN INGENIERÍA INFORMÁTICA

Desarrollo de un instrumento musical digital

Autor
Jesús Jiménez Sánchez

Directores
Alberto Guillén Perales



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE
TELECOMUNICACIÓN

Granada, julio de 2020

Desarrollo de un instrumento musical digital

Jesús Jiménez Sánchez

Palabras clave: batería, sonido, Raspberry Pi, Arduino, procesos, sensor

Resumen

Este proyecto consiste en el desarrollo una batería eléctrica utilizando una serie de sensores conectados a las placas Raspberry Pi y Arduino.

Los sensores están conectados a la placa Arduino, que recibe los valores leídos por los sensores de fuerza, y genera un mensaje que mandará a la Raspberry Pi.

En la Raspberry Pi, se leen los mensajes enviados por la Arduino y se decodifican para seleccionar un archivo de audio que se reproducirá por un altavoz.

Todo esto está colocado en una estructura que simula una batería.

En esta documentación se hablará sobre las placas Raspberry Pi y Arduino y sus modelos, hardware y alternativas. También se tratará el tema de los derechos de autor, su historia, la ley española y cuál es la protección de los archivos de audio utilizados en el proyecto.

Por último, se verá cómo se ha hecho el diseño e implementación del proyecto, comentando las decisiones tomadas, el funcionamiento del programa y los problemas encontrados en el desarrollo de la idea.

Development of a digital musical instrument

Jesús Jiménez Sánchez

Keywords: drums, sound, Raspberry Pi, Arduino, process, sensor

Abstract

This project is about the development of an electric drumset using sensors connected to the Raspberry Pi and Arduino boards.

The sensors are connected to the Arduino board, which receives the values read by the force sensors, and generates a message that will be sent to the Raspberry Pi.

In the Raspberry Pi, the messages are read and parsed to select an audio file to play in a speaker.

This is all set in an structure to simulate a drumset.

In this documentation I will talk about the Raspberry Pi and Arduino boards and its models, hardware and alternatives. I will also discuss the subject of copyright, its history, the Spanish law and what is the protection of the audio files used in this project.

Finally, we will see how the design and implementation of the project has been done, talking about the decisions made, how the program works and the problems found during the development of the idea.

Yo, **Jesús Jiménez Sánchez**, alumno de la titulación **Grado en Ingeniería Informática** de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación** de la **Universidad de Granada**, con DNI 1111111A, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Jesús Jiménez Sánchez

Granada a 5 de julio de 2020.

D. **Alberto Guillén Perales**, Profesor del Área de Arquitectura y Tecnología de Computadores del Departamento Arquitectura y Tecnología de Computadores de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado ***Desarrollo de un instrumento musical digital***, ha sido realizado bajo su supervisión por **Jesús Jiménez Sánchez**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 5 de julio de 2020.

Los directores:

Alberto Guillén Perales

Agradecimientos

A mi tutor, Alberto Guillén, por su ayuda en el desarrollo de este proyecto.

A mi tío José, por ayudarme con la madera de la batería.

A todos los profesores que me han dado clase hasta llegar a este momento. Gracias a su dedicación he conseguido llegar hasta aquí.

A mi madre Malena, mi padre Luis, mi hermana Ana Isabel y toda mi familia, por estar en todo momento y apoyarme siempre que lo necesitara.

Y a mi pareja Eva y todos mis amigos, por aguantarme en todos los momentos, los buenos y los malos, de este largo y sinuoso camino.

Índice general

1. Objetivos	19
2. Introducción	21
2.1. Historia de los instrumentos digitales	21
2.2. Partes de una batería	22
2.3. Planificación del trabajo	23
2.3.1. Etapas de desarrollo del trabajo	23
2.3.2. A priori optimista	23
2.3.3. A posteriori	24
3. Estado del arte	27
3.1. Raspberry Pi	27
3.1.1. ¿Qué es Raspberry Pi?	27
3.1.2. Modelos	28
3.1.3. Hardware	28
3.1.4. Alternativas	28
3.1.5. Lucha contra el Covid-19	29
3.2. Arduino	29
3.2.1. ¿Qué es Arduino?	29
3.2.2. Modelos	30
3.2.3. Alternativas	30
3.3. Opciones actuales en el mercado	31
3.4. Propuesta	31
4. Aspectos legales	33
4.1. Derechos de autor	33
4.1.1. ¿Qué son los derechos de autor?	33
4.1.2. Historia de los derechos de autor	33
4.1.3. Derechos de autor en España	34
4.2. Origen de sonidos de batería	35
5. Diseño de la propuesta	37
5.1. Tecnologías empleadas	37
5.1.1. Lenguaje de programación: C	37

5.1.2. Visual Studio Code	37
5.1.3. Github	38
5.1.4. Clockify	39
5.1.5. Overleaf	39
5.2. Descripción a alto nivel	40
5.2.1. Recogida de información	40
5.2.2. Tratamiento de la información y reproducción del sonido	41
5.3. Decisiones tomadas	42
5.3.1. Biblioteca de reproducción de sonido	42
5.3.2. Otras bibliotecas	43
5.3.3. if-else vs switch	43
5.4. Arduino vs Raspberry Pi	45
5.4.1. Conexión	45
6. Implementación de la propuesta	47
6.1. Reproducción de sonido	47
6.2. Sonido en paralelo	48
6.3. Principales problemas	50
6.3.1. Número de lecturas en botones	50
6.3.2. Hebras vs procesos	50
6.3.3. Número de lecturas en sensores	50
6.3.4. Construcción de paths	50
6.4. Pads	51
6.4.1. Madera MDF	51
6.4.2. Construcción	52
6.5. Tiempo	55
6.6. Coste y presupuesto	56
6.6.1. Mano de obra	56
6.6.2. Materiales	56
7. Conclusiones	57
7.1. Trabajo futuro	57
Bibliografía	59

Índice de figuras

2.1. Ejemplo de sampler digital de la marca AKAI Pro [27]	22
2.2. Partes de una batería [25]	23
2.3. Diagrama de Gantt de la planificación por etapas	24
2.4. Gráfica de los commits en el repositorio del código	24
2.5. Gráfica de los commits en el repositorio de la memoria	24
3.1. Raspberry Pi 3B [14]	27
3.2. Tabla modelos de Raspberry Pi [42]	28
3.3. Arduino Nano [4]	30
3.4. Tabla modelos de Arduino [2]	30
3.5. Resultados de la búsqueda de baterías eléctricas en Thomann [32]	31
5.1. Captura de pantalla de Visual Studio Code	38
5.2. Captura de pantalla de las issues de Github del proyecto . .	39
5.3. Captura de pantalla de Clockify	39
5.4. Captura de pantalla de Overleaf	40
5.5. Diagrama de la recogida de información	41
5.6. Diagrama de la estructura del mensaje (Valores posibles de 0 a 1023)	41
5.7. Diagrama del tratamiento de la información y la reproducción del sonido	42
5.8. Gráfica comparativa if-else vs switch	44
5.9. Esquema de conexión de sensores de presión [1]	45
6.1. Diagrama de flujo	48
6.2. Imagen de tablero MDF	51
6.3. Imagen de los pads una vez recortados	52
6.4. Imagen de los pads y un sensor tras añadir los agujeros . .	53
6.5. Imagen de los pads con la gomaeva	54
6.6. Imagen de la batería termimada	55

Capítulo 1

Objetivos

El objetivo principal de este proyecto es el de implementar y construir un instrumento musical digital, en este caso una batería, para producir música utilizando una Raspberry Pi.

Objetivos específicos:

- **OB-E1:** Reproducir sonidos de forma simultánea y reaccionando a estímulos en diferentes sensores con el menor retraso posible.
- **OB-E2:** Realizar la conexión de distintos sensores y leer los datos devueltos por estos para su posterior procesamiento.
- **OB-E3:** Creación de una experiencia de tocar la batería lo más cercana a la realidad posible.
- **OB-E4:** Mantener un presupuesto ajustado para fabricar una batería barata y sencilla.

Capítulo 2

Introducción

2.1. Historia de los instrumentos digitales

Los primeros intentos de guardar y reproducir el sonido fueron analógicos. Estos métodos captan las ondas y las almacenan en diferentes medios para su posterior reproducción, como en un disco de vinilo o un casete [29].

Con la aparición de la informática y los ordenadores se empiezan a almacenar estos sonidos en un formato digital, capaz de ser interpretado por ordenadores. En este caso, los sonidos se guardan en forma de bytes en diferentes formatos de archivo, como MP3, AAC, OGG... Al igual que el software, los formatos de audio pueden ser abiertos o cerrados. Por ejemplo, WMA [44] o AAC [35] son formatos propietarios, mientras que OGG [41] o ALAC [36] son formatos abiertos. MP3 pasó a ser un formato abierto a partir de 2017 [12].

Los formatos de archivo de audio se pueden ordenar en tres categorías principales:

No comprimidos	Compresión con pérdidas	Compresión sin pérdidas
AU, WAV, AIFF...	MP3, AAC, OGG...	FLAC, ALAC, WMA...

Cuadro 2.1: Tabla con ejemplos de formatos de las tres categorías [20]

- **No comprimidos:** Consisten en capturar las ondas del sonido y guardarlas en archivos sin ningún procesamiento posterior.
- **Compresión con pérdidas:** En el proceso de compresión se pierde algo de información y, con ella, algo de calidad. A cambio de la pérdida de calidad, se obtienen ficheros más ligeros.
- **Compresión sin pérdidas:** Al contrario que en la compresión con pérdidas, en este tipo de compresión no se pierde nada de información ni de calidad de audio, sin embargo, se obtienen archivos más pesados que en la categoría anterior.

Los primeros intentos de instrumento musical no analógico se pueden encontrar en los sintetizadores. Estos instrumentos utilizan la electricidad para producir las ondas del sonidos pasándola por una serie de módulos. Hasta la década de 1980, cada fabricante utilizaba su propio estándar para la sincronización de los sonidos de los sintetizadores. En 1981, la empresa Oberheim Electronics comenzó a contactar con otros fabricantes para desarrollar un estándar, de ese modo apareció MIDI. Este estándar describe el protocolo de comunicación, la interfaz digital y las conexiones electrónicas que deben llevar los diferentes tipos de dispositivos electrónicos para reproducir, editar y grabar música [40].

En cuanto a instrumentos musicales electrónicos podemos encontrar desde un teclado o una guitarra a una batería.

En el área de instrumentos digitales nos encontramos con los instrumentos VST (Virtual Studio Technology). Estos instrumentos toman muestras de sonidos de diferentes instrumentos y, mediante un teclado y un ordenador, programar estos sonidos y componer y grabar cualquier tipo de canción [29].



Figura 2.1: Ejemplo de sampler digital de la marca AKAI Pro [27]

2.2. Partes de una batería

A continuación se explicarán las partes principales de una batería. Se ha elegido el instrumento musical de la batería porque, personalmente, es un instrumento que me gusta y toco desde hace unos años. Tengo una batería eléctrica y me parecía interesante hacer una por mí mismo.

Las partes de una batería son las siguientes:

- **Caja:** Su función principal suele ser la de marcar los compases.
- **Toms:** Son los tambores más numerosos en una batería.
- **Bombo:** Se toca con un pedal y produce el sonido más grave de la batería. Se utiliza para llevar la base del ritmo.

- **Platillo crash:** Se utiliza para dar énfasis y suele ir acompañado del bombo.
- **Platillo hi-hat:** Consta de dos platillos que se pueden abrir o cerrar con un pedal y se utiliza para llevar el ritmo de la canción.
- **Platillo ride:** Puede usarse para llevar el ritmo en lugar de con el hi-hat.



Figura 2.2: Partes de una batería [25]

2.3. Planificación del trabajo

2.3.1. Etapas de desarrollo del trabajo

- **1^a etapa:** Estudio del problema.
- **2^a etapa:** Búsqueda de bibliotecas de reproducción de sonido.
- **3^a etapa:** Implementación del software.
- **4^a etapa:** Construcción de la batería.
- **5^a etapa:** Documentación.

2.3.2. A priori optimista

En el siguiente diagrama de Gantt se detalla una planificación de cómo se espera desarrollar el proyecto en el tiempo.

De los nueve meses que se dedicarán al proyecto, seis estarían enfocados en el desarrollo del software que controlará la batería, dejando unos tres

meses a la creación de prototipos y versiones secuenciales del programa, y otros tres meses al desarrollo de la versión concurrente y final del software.

Los meses restantes, se dedicarían a la construcción de la batería y la escritura de la memoria, dejando el mes de mayo libre, en caso de que hubiera imprevistos y alguna de las fases se alargara.

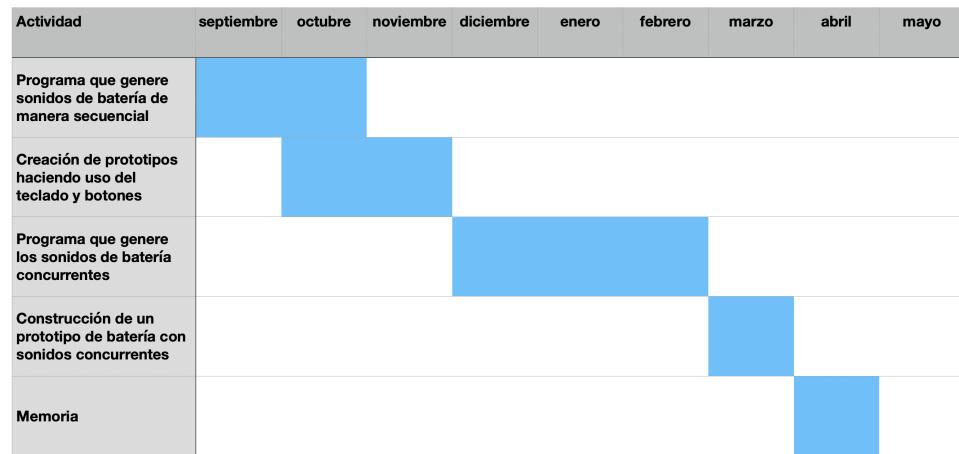


Figura 2.3: Diagrama de Gantt de la planificación por etapas

2.3.3. A posteriori

Una vez terminado el proyecto, se puede ver en las gráficas que proporciona Github sobre cuándo se han realizado los commits en los dos repositorios (código y memoria) cuál es la diferencia entre la planificación a priori y lo que ha terminado resultando.

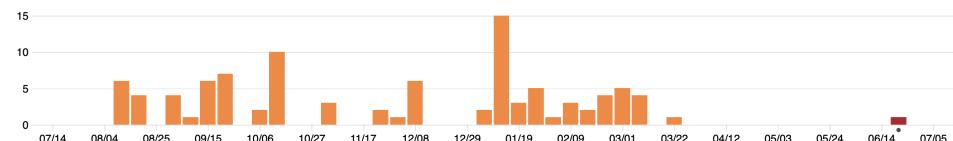


Figura 2.4: Gráfica de los commits en el repositorio del código

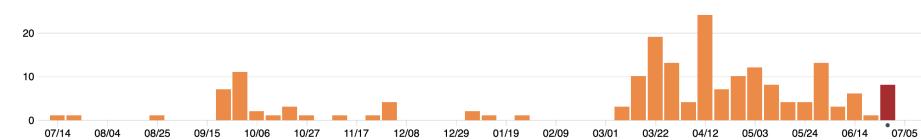


Figura 2.5: Gráfica de los commits en el repositorio de la memoria

Se puede ver en las imágenes que, además de haber empezado en agosto, en lugar de en septiembre, con el código, éste ha seguido bastante la planificación previamente indicada. Si nos vamos al historial de commits del repositorio, se puede observar también que el desarrollo del sistema de sonido concurrente también sigue a la planificación a priori bastante bien.

Es en la escritura de la memoria en la que se ven más diferencias, ya que toma gran parte de marzo y mayo, meses que no estaban planificados para la memoria. Esto se debe, mayormente a subestimar la dificultad del desarrollo de la misma y a poca experiencia en la escritura de este tipo de documentos.

La construcción de la batería se fue realizando a lo largo de los diez meses que he dedicado a este proyecto.

Capítulo 3

Estado del arte

Este capítulo se encarga de introducir las dos placas utilizadas en el proyecto, explicar las opciones actuales en el mercado en cuanto a baterías eléctricas y realizar la propuesta de batería que se llevará a cabo.

3.1. Raspberry Pi

3.1.1. ¿Qué es Raspberry Pi?

“Es un ordenador del tamaño de una tarjeta de crédito. Consta de una placa base sobre la que se monta un procesador, un chip gráfico y memoria RAM. Fue lanzado en 2006 por la Fundación Raspberry Pi con el objeto de estimular la enseñanza de informática en las escuelas de todo el mundo.”

(El Confidencial, 22 de Noviembre de 2013. [26])



Figura 3.1: Raspberry Pi 3B [14]

Se ha vuelto un producto tan popular que se vende para todo tipo de usos, desde centros multimedia [34] o espejos inteligentes [17], hasta respiradores [22] o este mismo proyecto.

3.1.2. Modelos

En sus ocho años de existencia, la Fundación Raspberry Pi ha lanzado cinco modelos de la Raspberry Pi, con diferentes variaciones. En la figura 3.2 se detallan alguno de los detalles de estos modelos [42]:

Familia	Modelo	Form Factor	Ethernet	Wireless	GPIO	Lanzado	Descontinuado	
Raspberry Pi 1	B	Estándar (85.60 x 56.5 mm)	Si	No	26 pines	2012	Si	
	A		No			2013	Si	
	B+		Si			2014		
	A+	Compacto (65 x 56.5 mm)	No			2014		
Raspberry Pi 2	B	Estándar	Si	No	40 pines	2015		
Raspberry Pi Zero	Zero	Zero (65 x 30 mm)	No	No		2015		
	W/WH			Si		2017		
Raspberry Pi 3	B	Estándar	Si	Si		2016		
	A+	Compacto	No			2018		
	B+	Estándar	Si			2018		
Raspberry Pi 4	B (1 GiB)	Estándar	Si	Si		2019		
	B (2 GiB)							
	B (4 GiB)							

Figura 3.2: Tabla modelos de Raspberry Pi [42]

3.1.3. Hardware

La Raspberry Pi 3B utilizada en este proyecto tiene el siguiente hardware [15]:

- **Procesador:** Broadcom BCM2837 de cuatro núcleos con arquitectura ARM Cortex A53 (ARMv8) a 1.2 GHz.
- **Memoria:** 1GB de memoria LPDDR2
- **GPU:** Broadcom VideoCore IV a 250 MHz
- **USB:** Cuatro puertos USB 2.0.
- **GPIO:** Hay 40 pines de GPIO (General Purpose Input/Output). Estos pines funcionan a 3.3V.
- **Internet:** Ethernet 10/100 Mbit/s y WiFi 802.11 b/g/n

3.1.4. Alternativas

Podemos encontrar una gran variedad de alternativas, algunas de ellas son las siguientes [11]:

- **Orange Pi Prime:** Esta alternativa está viendo un gran crecimiento en los últimos años. La compañía que la fabrica se centra en precios más baratos y una gran personalización de la placa.

- **Banana Pi M3:** Puede compararse en prestaciones con la Raspberry Pi 3, sin embargo, el precio es mayor, ya que esta cuesta alrededor de \$80.
- **ASUS Tinker Board:** Es la más compatible a nivel de software, además, cuenta con más potencia de cálculo, lo que reduce los tiempos de procesamiento.
- **Huawei HiKey 960:** La alternativa de Huawei es la que cuenta con más potencia. Utiliza el procesador que utilizan los teléfonos de la compañía (Kirin 960). Pero también es la más cara, \$300.

Tras ver estas alternativas, se ha decidido utilizar la Raspberry Pi, principalmente, por ser la estándar entre las placas. Es la que más documentación tiene en Internet, haciendo, por tanto, más sencillo el desarrollo del proyecto y, al ser la más utilizada, también es más sencillo hacerse con ella desde España.

En concreto, se utiliza el modelo Raspberry Pi 3B, por ser el modelo más avanzado en el momento de la compra, aunque un mes después se lanzó la Raspberry Pi 4, con mayor potencia y memoria, entre otras cosas.

3.1.5. Lucha contra el Covid-19

Debido a la reciente crisis del coronavirus Covid-19, muchas personas empezaron a utilizar la Raspberry Pi para ayudar a luchar contra el virus. La principal utilidad que se encontró para esta placa en esa situación fue la de crear respiradores, pudiendo usar un código publicado en Github [22] para programar la Raspberry Pi.

Las ventas, por tanto, se dispararon, llegando a las 640.000 unidades vendidas en el mes de marzo [19].

3.2. Arduino

3.2.1. ¿Qué es Arduino?

“Arduino es una plataforma electrónica de código abierto basada en hardware y software fácil de usar. Las placas Arduino pueden leer entradas (luz en un sensor, un dedo en un botón o un mensaje de Twitter) y convertirlo en una salida: activar un motor, encender un LED, publicar algo online. Para hacerlo, utiliza el lenguaje de programación Arduino y el Software Arduino (IDE).”

(Arduino, 2 de abril de 2020. [3])

El proyecto Arduino nació en el año 2003 en el Interaction Design Institute Ivrea en Italia y, tanto la placa como el software que utiliza son open-source.

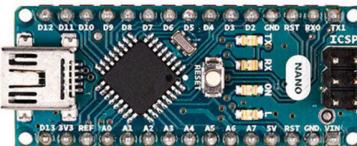


Figura 3.3: Arduino Nano [4]

3.2.2. Modelos

En la figura 3.4 se especifican los modelos de entrada de Arduino junto con algunas de sus especificaciones de hardware [2]:

Name	Processor	Operating/Input Voltage	CPU Speed	Analog In/Out	Digital IO/PWM	EEPROM [kB]	SRAM [kB]	Flash [kB]	USB	UART
101	Intel® Curie	3.3 V / 7-12V	32MHz	6/0	14/4	-	24	196	Regular	-
Uno	ATmega328P	5 V / 7-12 V	16 MHz	6/0	14/6	1	2	32	Regular	1
Leonardo	ATmega32U4	5 V / 7-12 V	16 MHz	12/0	20/7	1	2.5	32	Micro	1
Nano	ATmega168 ATmega328P	5 V / 7-9 V	16 MHz	8/0	14/6	0.512 1	1 2	16 32	Mini	1

Figura 3.4: Tabla modelos de Arduino [2]

El modelo utilizado en este proyecto es el Arduino Nano.

3.2.3. Alternativas

Al igual que para Raspberry Pi (3.1.4), hay una gran variedad de alternativas para la placa Arduino [7]:

- **NodeMCU:** Es la solución más barata. Además, añade la posibilidad de ejecutar código Lua.
- **Teensy 3:** Esta placa incluye un procesador más potente, haciéndola capaz de ejecutar tareas más pesadas. También se puede programar mediante el Arduino IDE, con la biblioteca Teensyduino.
- **MSP430 Launchpad:** Esta alternativa se centra en un consumo de energía más bajo que la Arduino.

Arduino, al igual que Raspberry Pi, es la estándar en este tipo de placas y cuenta con una mayor documentación y ejemplos disponibles. En concreto se utiliza el modelo Arduino Nano, que es uno de los modelos más pequeños de la placa, con potencia suficiente para este proyecto.

3.3. Opciones actuales en el mercado

Son múltiples las baterías eléctricas a la venta. Una búsqueda rápida en Thomann [32] nos muestra una gran variedad de baterías eléctricas en un amplio rango de precios, desde los 109€ hasta los 8.398€.

The screenshot shows the Thomann website's search results for 'Sets de batería electrónica'. The search bar at the top contains the query. The results page has a sidebar on the left with filters for brand (Roland, Millenium, Alesis, Yamaha), rating (★★★★★ to ★★★★★), price range (0 € to 8398 €), and other options like 'Include Rack' and 'Include pedal'. The main content area displays seven product cards for different electronic drum sets, each with an image, name, price, and rating. The products listed are:

- Set completo de batería electrónica Millenium MPS-850 E-Drum Set - 249 €
- Batería electrónica portátil Millenium MD-90 Mobile Drum - 109 €
- Set de batería electrónica Millenium MPS-150X E-Drum Mesh Set - 298 €
- Batería electrónica Millenium HD-120 E-Drum Set - 598 €
- Millenium MPS-150 E-Drum Set - 298 €
- Set de batería electrónica Roland TD-17KVX E-Drum Set - 1.299 €
- Millenium MPS-150X E-Drum Mesh Set - 1.599 €

Figura 3.5: Resultados de la búsqueda de baterías eléctricas en Thomann [32]

En el terreno de las baterías que los usuarios se pueden construir (DIY), los principales resultados [31] utilizan elementos como sensores piezoelectrinos y sintetizadores MIDI. Los precios de estos sintetizadores van desde 150€ a 2100€.

3.4. Propuesta

La propuesta que se presenta es un crear un programa de código abierto para que cualquier persona con unos conocimiento medios de informática pueda montar su propia batería eléctrica en casa.

Para completar con éxito el objetivo de este proyecto, los puntos más importantes serán:

- Reproducción del sonido con el menor delay posible.
- Reproducción de varios sonidos de manera concurrente.
- Sensación de tocar la batería lo más realista posible.

Capítulo 4

Aspectos legales

Los sonidos y los programas de ordenador pueden ser protegidos mediante leyes de derechos de autor, por este motivo se incluye este capítulo. A continuación se da un contexto de la historia de los derechos de autor, cómo afectan a los programas de ordenador y se explica el origen de los sonidos de batería utilizados y la forma en la que estos están protegidos.

4.1. Derechos de autor

4.1.1. ¿Qué son los derechos de autor?

Los derechos de autor son una serie de leyes que protegen la autoría de las obras. Estas pueden ser libros, películas, obras de teatro, programas informáticos...

Se cubren dos tipos de derechos: los derechos patrimoniales, que aseguran que el autor obtenga compensación financiera, y los derechos morales, que cubren todo lo que no esté relacionado con los derechos patrimoniales, por ejemplo, la prohibición de que se modifique la obra [24].

4.1.2. Historia de los derechos de autor

La historia de los derechos de autor comienza en 1710, cuando se publica el Estatuto de la Reina Anna [10] que fue el primer reglamento sobre los derechos de autor. En el momento de la publicación de este estatuto solo se contemplaban los derechos sobre los libros, pero en posteriores leyes se contemplan otros usos, como cine, radio, fotografías o programas de ordenador.

En la actualidad, los derechos de autor se protegen tanto con acuerdos y leyes internacionales, como leyes nacionales.

Desde 1974 en Estados Unidos con la CONTU [38] (Commission on New Technological Uses of Copyrighted Works) y de 1991 en la Unión Europea

con la Computer Programs Directive [13], se protegen los derechos de autor de los programas informáticos.

4.1.3. Derechos de autor en España

En España tenemos el Real Decreto Legislativo 1/1996, de 12 de abril, por el que se aprueba el texto refundido de la Ley de Propiedad Intelectual, regularizando, aclarando y armonizando las disposiciones legales vigentes sobre la materia.. En esta ley, la propiedad intelectual se define como:

“La obra literaria, artística o científica, expresadas en cualquier medio (libros, escritos, composiciones musicales, obras, coreografías, obras audiovisuales, esculturas, obras pictóricas, planos, maquetas, mapas, fotografías, programas de ordenador y bases de datos) que corresponde a su autor por el solo hecho de su creación, que tiene derecho a explotarla y disponer de ella a su voluntad.”

(La Ley de Propiedad Intelectual y los derechos de autor en España. [6])

Tipos de derechos

En la Ley de Propiedad Intelectual se definen dos grupos de derechos de autor:

- **Derechos morales:** Corresponden al autor de la obra y no se puede renunciar a ellos ni traspasarlos. Se encargan de proteger la identidad y reputación del autor.
- **Derechos patrimoniales:** Estos sí se pueden traspasar. Sirven para que el autor decida cómo se utiliza y representa su obra. No se puede usar la obra sin autorización expresa.

Programas de ordenador

La Ley de Propiedad Intelectual define *programa de ordenador* como:

“A los efectos de la presente Ley se entenderá por programa de ordenador toda secuencia de instrucciones o indicaciones destinadas a ser utilizadas, directa o indirectamente, en un sistema informático para realizar una función o una tarea o para obtener un resultado determinado, cualquiera que fuere su forma de expresión y fijación.

A los mismos efectos, la expresión programas de ordenador comprenderá también su documentación preparatoria. La documentación técnica y los manuales de uso de un programa gozarán de la misma protección que este Título dispensa a los programas de ordenador.”

(Real Decreto Legislativo 1/1996, de 12 de abril. [9])

El autor del programa será la persona o grupo de personas que lo hayan creado y tendrán derechos de explotación sobre el programa durante la vida del autor y setenta años después de su muerte.

El autor tendrá derecho a realizar o autorizar:

- Reproducción total o parcial.
- Traducción, adaptación, arreglo o cualquier cambio.
- Distribución pública.

4.2. Origen de sonidos de batería

Los sonidos de batería han sido obtenidos de la biblioteca de sonidos de GarageBand para macOS. Estos sonidos son libres y gratuitos para usarse en composiciones musicales o proyectos de audio originales, tal y como estipula la página oficial de Apple [5].

Capítulo 5

Diseño de la propuesta

En este capítulo se explican las tecnologías empleadas en el proyecto, se detallan las principales decisiones tomadas y se realiza una descripción a alto nivel de este trabajo.

5.1. Tecnologías empleadas

5.1.1. Lenguaje de programación: C

El lenguaje utilizado para la mayor parte del proyecto ha sido C. C es un lenguaje desarrollado en 1972 por Dennis Ritchie en los Laboratorios Bell. Fue creado para la creación de sistemas operativos, como UNIX, y es muy valorado por su eficiencia.

En la actualidad, es uno de los lenguajes más populares. Está disponible en muchas plataformas, desde superordenadores, hasta sistemas empotrados. Fue diseñado para mapear las instrucciones escritas en C a lenguaje máquina de forma muy eficiente, con pocas instrucciones, esto hace que sea muy eficiente en cualquier plataforma en la que se utilice [37].

Es por esta eficiencia, por la que se ha utilizado en el proyecto. Cualquier instrumento musical, y en especial una batería, requiere que las acciones realizadas por el músico tengan una reacción rápida por el sistema, y C puede proporcionar esta velocidad.

5.1.2. Visual Studio Code

Visual Studio Code es un editor de texto diseñado por Microsoft para Windows, Linux y macOS, lanzado al público en 2016.

Su sencillez, compatibilidad con una gran cantidad de lenguajes de programación y una extensa biblioteca de extensiones han hecho que VS Code sea uno de los editores de texto más populares entre los programadores. [43]

La captura de pantalla muestra la interfaz de Visual Studio Code. En la parte superior, se ven los titulares de los documentos abiertos: main.c, play.c, read.c, Makefile, play.h, read.h, shared.h y force.h. El Explorador de archivos (izquierda) muestra una jerarquía de directorios y archivos, incluyendo 'src' que contiene 'buttons.c', 'keys.c', 'main_antiguo.c', 'main.c', 'play.c', 'read-serial.c', 'read.c' (selecciónada), 'sensor.c', 'test_buttons.c', 'LICENSE', 'README.md', 'rpl_SCH_3plus_Io_reduced.pdf' y 'time_if-switch.csv'. El Editor de código (derecha) muestra el contenido del archivo 'read.c'. El pie de pantalla indica que el archivo fue modificado hace un mes, tiene 1 línea, 1 columna y 26 minutos de duración, y que el tamaño de tabulación es 4 (UTF-8). Los iconos en la barra de herramientas incluyen un ícono de GitHub.

```

    70     |     PressToPlay(r_bass);
    71   }
    72 }
    73
    74 void readSerial() {
    75     struct termios options;
    76     int fd, i, j;
    77     char buf[60] = {'\0'};
    78
    79     /* open serial port */
    80     fd = open("/dev/ttyACM0", O_RDWR | O_NOCTTY);
    81
    82     /* wait for the Arduino to reboot */
    83     // usleep(3500000);
    84
    85     /* get current serial port settings */
    86     tcgetattr(fd, &options);
    87
    88     /* set 9600 baud both ways */
    89     cfsetspeed(&options, B9600);
    90     cfsetspeed(&options, B9600);
    91
    92     /* 8 bits, no parity, no stop bits */
    93     options.c_cflag |= PARENB;
    94     options.c_cflag |= CSTOPB;
    95     options.c_cflag |= CSIZE;
    96     options.c_cflag |= CS8;
    97
    98     /* Canonical mode */
    99     options.c_lflag |= ICRON;
    100
    101     /* commit the serial port settings */
    102     tcsetattr(fd, TCSANOW, &options);
    103
    104     /* Send byte to trigger Arduino to send string back */
    105     write(fd, "0", 1);
    106
    107     while (true) {
    108         /* Receive string from Arduino */
    109         read(fd, buf, 60);
    110
    
```

Figura 5.1: Captura de pantalla de Visual Studio Code

5.1.3. Github

Github es un servicio de alojamiento de código abierto y control de versiones mediante Git.

En este proyecto se ha utilizado principalmente para llevar ese control de versiones del código y para organizar los diferentes pasos a dar y los errores encontrados durante el desarrollo mediante su capacidad de crear *issue*. Aparte, Github permite el acceso al código por parte de cualquier persona y ofrece la posibilidad de contribuir al proyecto a todo el que quiera.

El repositorio con el código del proyecto disponible para que cualquier persona pueda verlo está en este enlace: <https://github.com/jesusjimsa/Drum-It-Yourself>.

Además, el texto en formato LaTeX y las imágenes de esta memoria están disponibles también en este repositorio: <https://github.com/jesusjimsa/Memoria-Drum-It-Yourself-TFG>.

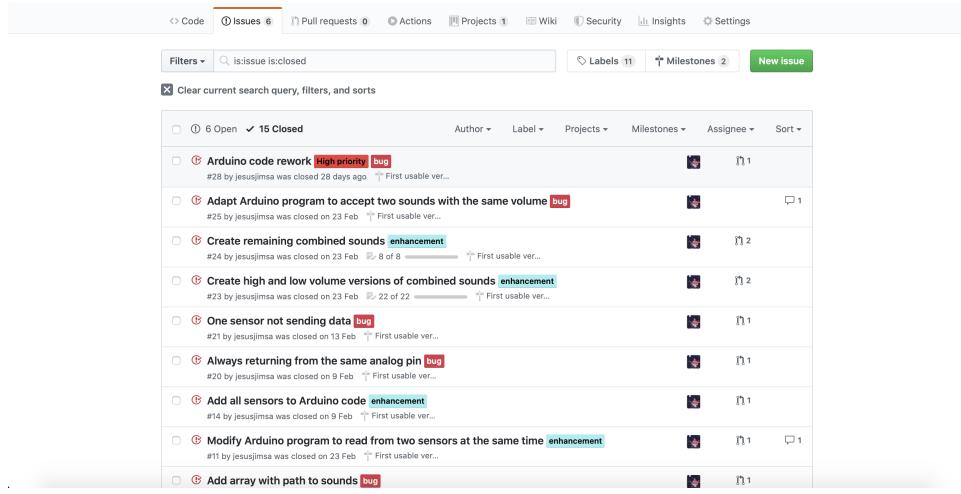


Figura 5.2: Captura de pantalla de las issues de Github del proyecto

5.1.4. Clockify

Clockify es una aplicación de seguimiento de tiempo. Ha sido utilizada para llevar un seguimiento de cuánto tiempo ha sido dedicado a la creación y desarrollo de este proyecto y su memoria.

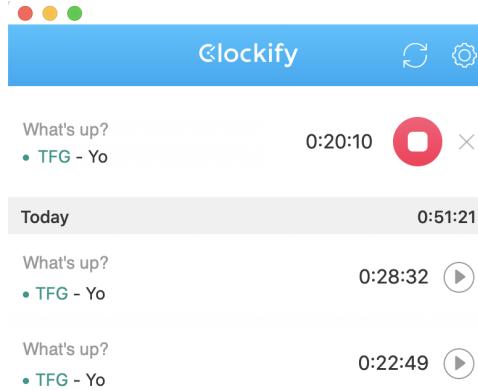


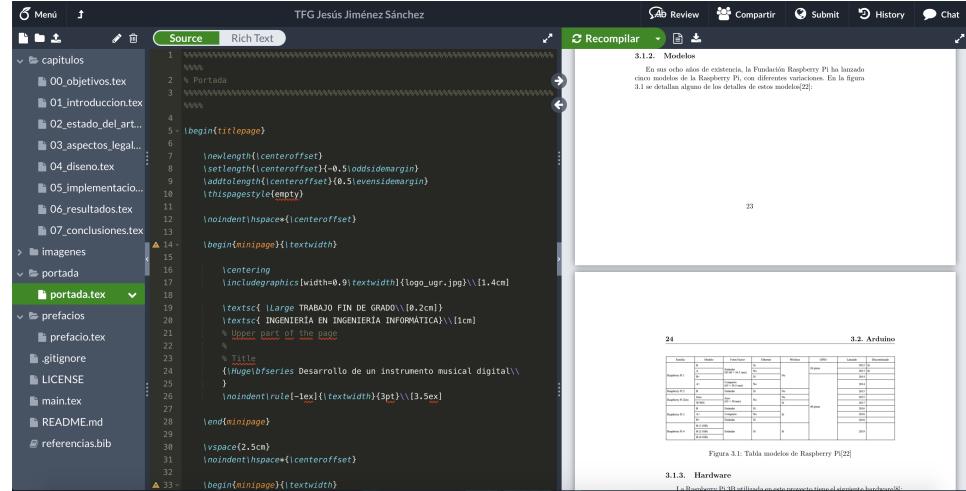
Figura 5.3: Captura de pantalla de Clockify

5.1.5. Overleaf

Overleaf es un editor de LaTeX online y colaborativo. Permite crear documentos en LaTeX y la colaboración de hasta dos personas en su modalidad gratuita.

En este proyecto, ha facilitado la compartición de esta memoria entre alumno y tutor para su corrección, a parte de servir como editor principal

de LaTeX.



The screenshot shows the Overleaf interface with the LaTeX source code for a title page. The code includes commands for page layout, graphics inclusion, and text content. A sidebar on the left lists the project files, and a right-hand panel shows the rendered document preview.

```

\begin{titlepage}
\begin{center}
\includegraphics[width=0.9\textwidth]{logo_ugr.jpg}\\[1.4cm]
\Large TRABAJO FIN DE GRADO\\[0.2cm]
\textsc{INGENIERIA EN INGENIERIA INFORMATICA}\\[1cm]
% Upper part of the page
\% Title
\huge Desarrollo de un instrumento musical digital\\
\% Author
\large Alumno: Daniel Jiménez Sánchez\\
\% Date
\large Fecha: 2018-06-20\\
\% Advisor
\large Director: Juan José Gómez\\
\% Committee
\large Comité: \\
\% University
\large Universidad de Granada\\
\% Faculty
\large Facultad de Informática\\
\% Department
\large Departamento de Ingeniería Informática\\
\% Address
\large Avda. de la Pintor Sorolla, 33\\
\% City
\large 18071 Granada\\
\% Country
\large España\\
\end{center}

```

Figura 5.4: Captura de pantalla de Overleaf

5.2. Descripción a alto nivel

La solución propuesta consta de dos partes principales. La primera parte es la recogida de información y la segunda, el tratamiento de la información y reproducción del sonido.

5.2.1. Recogida de información

En esta primera fase se recoge la información que devuelven los sensores de fuerza en la placa Arduino. Los cinco sensores se conectan a la Arduino y devuelven un valor entre 0 y 1023. Este valor se junta a los valores leídos por todos los sensores y se manda en un solo mensaje a la Raspberry Pi para empezar la segunda fase.

Modelo	Memoria	CPU	Placa	USB	Serial	Bluetooth
Raspberry Pi 3 Model B	1 GB	ARM Cortex-A53	90 mm x 50 mm	4x USB 2.0	4x Serial	1x Bluetooth
Raspberry Pi 3 Model B+	1 GB	ARM Cortex-A53	90 mm x 50 mm	4x USB 2.0	4x Serial	1x Bluetooth
Raspberry Pi 3 Model A+	0.5 GB	ARM Cortex-A53	90 mm x 50 mm	2x USB 2.0	2x Serial	1x Bluetooth
Raspberry Pi 2 Model B	0.5 GB	ARM Cortex-A7	90 mm x 50 mm	2x USB 2.0	2x Serial	1x Bluetooth
Raspberry Pi 2 Model B+	0.5 GB	ARM Cortex-A7	90 mm x 50 mm	2x USB 2.0	2x Serial	1x Bluetooth
Raspberry Pi Zero W	0.5 GB	ARM Cortex-A7	40 mm x 30 mm	1x USB 2.0	1x Serial	1x Bluetooth
Raspberry Pi Zero	0.5 GB	ARM Cortex-A7	40 mm x 30 mm	1x USB 2.0	1x Serial	None

Figura 5.1: Tabla modelos de Raspberry Pi[2]

Nombre	Modelo	Funciones	Alimentación	Pines	USB	Serial	Bluetooth
Raspberry Pi 3 Model B	B	1 GB	5V	40	4	4	1
Raspberry Pi 3 Model B+	B+	1 GB	5V	40	4	4	1
Raspberry Pi 3 Model A+	A+	0.5 GB	5V	26	2	2	1
Raspberry Pi 2 Model B	B	0.5 GB	5V	40	2	2	1
Raspberry Pi 2 Model B+	B+	0.5 GB	5V	40	2	2	1
Raspberry Pi Zero W	W	0.5 GB	5V	26	1	1	1
Raspberry Pi Zero		0.5 GB	5V	26	1	1	None

Figura 5.2: Tabla modelos de Raspberry Pi[2]

Nombre	Modelo	Funciones	Alimentación	Pines	USB	Serial	Bluetooth
Raspberry Pi 3 Model B	B	1 GB	5V	40	4	4	1
Raspberry Pi 3 Model B+	B+	1 GB	5V	40	4	4	1
Raspberry Pi 3 Model A+	A+	0.5 GB	5V	26	2	2	1
Raspberry Pi 2 Model B	B	0.5 GB	5V	40	2	2	1
Raspberry Pi 2 Model B+	B+	0.5 GB	5V	40	2	2	1
Raspberry Pi Zero W	W	0.5 GB	5V	26	1	1	1
Raspberry Pi Zero		0.5 GB	5V	26	1	1	None

Figura 5.3: Tabla modelos de Raspberry Pi[2]

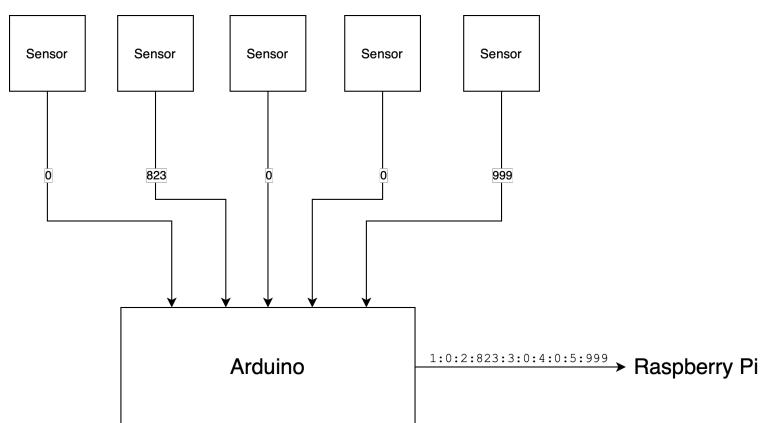


Figura 5.5: Diagrama de la recogida de información

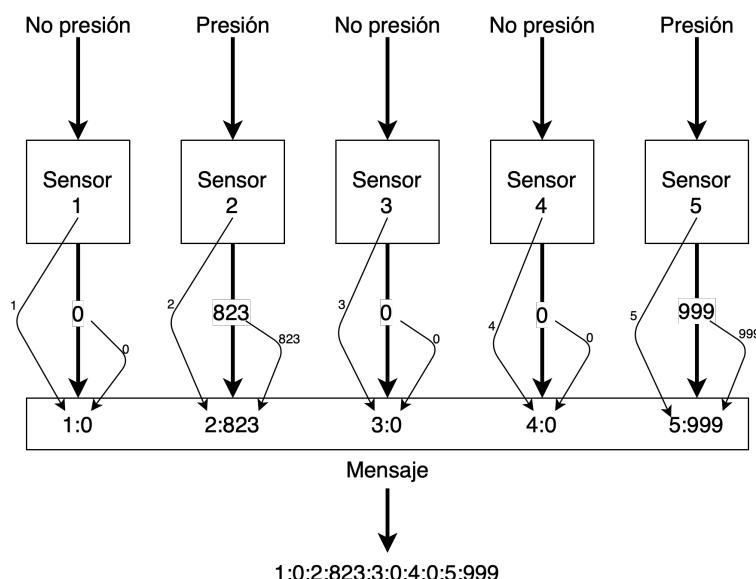


Figura 5.6: Diagrama de la estructura del mensaje (Valores posibles de 0 a 1023)

5.2.2. Tratamiento de la información y reproducción del sonido

En esta segunda parte, se recibe la información recogida por la Arduino y se reproducen los sonidos de batería correspondientes. La Arduino envía un mensaje por el log del monitor serie y la Raspberry Pi analiza este mensaje. Una vez separado el mensaje en pares de instrumento y volumen, se reproducen los sonidos correspondientes.

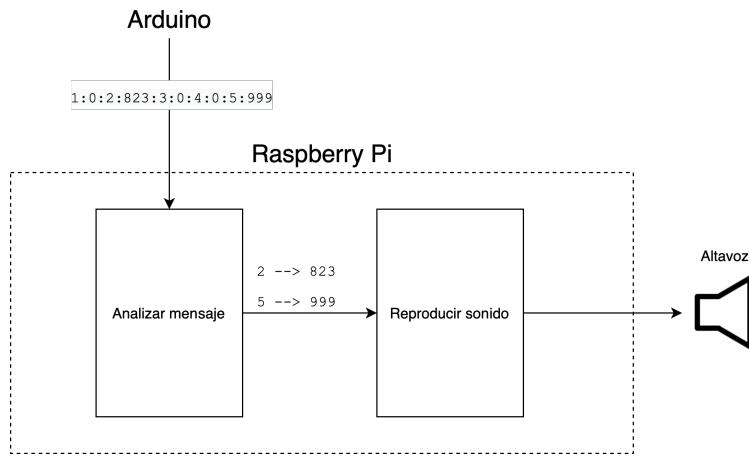


Figura 5.7: Diagrama del tratamiento de la información y la reproducción del sonido

5.3. Decisiones tomadas

La primera decisión fue entre hacer detección binaria de la entrada, es decir, si el parche ha sido golpeado o no, y hacer que estas entradas sean concurrentes (al golpear dos parches, el sonido de ambos suena al mismo tiempo), o hacer detección de distintos sonidos en un mismo parche, dependiendo de cómo se golpee el parche (en el centro, en el lateral, con más o menos fuerza...) el sonido emitido es diferente.

Se decide empezar con la primera alternativa y dejar la segunda para más adelante, en caso de tener tiempo.

Finalmente, por temas de coste, se decide no realizar la detección de distintos en un mismo parche, ya que cada sensor de fuerza cuesta 10,19€ por lo que añadir más sensores a cada parche haría que el precio final de desarrollar el proyecto se elevara demasiado.

5.3.1. Biblioteca de reproducción de sonido

`playsound`

La primera idea de lenguaje de programación para implementar el proyecto fue Python, por ser un lenguaje sencillo y con gran variedad de bibliotecas. Al investigar las bibliotecas de reproducción de sonidos disponibles para Python, la más recomendada era `playsound` [21]. Se empezaron a hacer pruebas con esta librería, pero la salida del sonido era demasiado lenta para un proyecto como este, en el que el tiempo que pasa entre que se golpea un parche y se produce la salida del sonido, tiene que ser el menor posible. Por esta razón, se decide descartar el uso de Python y `playsound`.

mpg123

Tras decidir no utilizar playsound de Python, se empezó a investigar bibliotecas de otros lenguajes de programación. La biblioteca de C, Mpg123 [23], es una de las más mencionadas en la reproducción de sonido. Por esta razón, se hacen pruebas con la biblioteca y se confirma que es lo suficientemente rápida para el proyecto, así que se decide seguir adelante con ella y es la biblioteca con la que se ha desarrollado el programa que gestiona los sonidos.

Tiene, sin embargo, un problema en el uso de hebras POSIX para reproducir varios sonidos al mismo tiempo. Para solucionar este problema, se utilizan procesos (lanzados por `fork()`) en lugar de hebras POSIX, como se explica en la sección 6.3.2.

libao

Para utilizar la biblioteca Mpg123 para la reproducción de sonidos, hay que utilizarla en conjunto con libao [33]. Mpg123 se encarga de descodificar el archivo MP3 que contiene el sonido correspondiente y prepararlo para su reproducción, mientras que libao se encarga de mandar la señal al sistema operativo para reproducirlo. Utilizando estas dos funciones conjuntamente es como se reproduce el sonido en este proyecto.

5.3.2. Otras bibliotecas

wiringPi

En los primeros prototipos del proyecto se utilizaban botones en lugar de sensores. Estos botones estaban directamente conectados a la Raspberry Pi, ya que aún no se contaba con la Arduino. Por este motivo apareció la necesidad de encontrar una biblioteca que controlara estos botones y la información que le mandaban al programa. La biblioteca más utilizada para este propósito es wiringPi [18]. Siendo incluso recomendada por la propia Raspberry Pi Foundation [16].

Finalmente, como se explica en la sección 5.4, se decide utilizar una placa Arduino para la lectura de los sensores, con lo cual la biblioteca wiringPi ya no es necesaria y no es utilizada en la versión final del proyecto.

5.3.3. if-else vs switch

Al pulsar una tecla, el número leído se envía a una función que selecciona qué sonido hay que reproducir en ese momento, dependiendo de qué sonido corresponda a ese número. Este proceso de selección se puede hacer con una estructura de *if-else* anidados o con un *switch-case*.

Para decidir cuál de las dos soluciones se implementa en la versión final se realizó un test en el que cada vez se ejecutan más iteraciones del programa

cambiando de sonido en cada una de ellas. Se empieza con 1 iteración y se termina con 10000000 iteraciones.

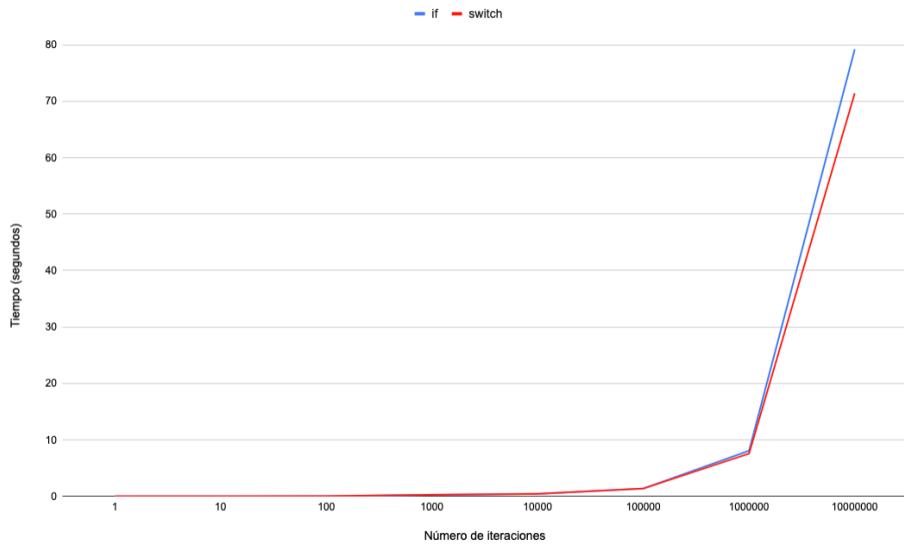


Figura 5.8: Gráfica comparativa if-else vs switch

iterations	if	switch
1	0.000243	0.000270
10	0.002797	0.002485
100	0.027775	0.027261
1000	0.260075	0.261464
10000	0.431544	0.425668
100000	1.368561	1.374575
1000000	8.070825	7.560718
10000000	79.199539	71.409653

Como se puede ver en la figura 5.8, la diferencia no es apreciable hasta las 1000000 iteraciones, pero después pasa a casi 8 segundos de diferencia en 10000000 iteraciones. Por esta razón se ha decidido que la función utilice la estructura `switch-case`.

Finalmente, debido a la manera en la que realizan las comprobaciones de qué botones y sensores son utilizados, aunque un `switch-case` es más rápido, esta estructura se reserva para la versión del programa que reproduce los sonidos leyendo del teclado. En el programa que controla los sensores se utiliza una estructura `if-else`.

5.4. Arduino vs Raspberry Pi

Para la lectura de las señales de los sensores de presión RP c18.3 y RP S40, se plantean dos opciones, se puede utilizar la propia Raspberry Pi en la que se ejecuta el programa que maneja los sonidos o una Arduino Nano. En el proyecto resultante se utiliza finalmente la Arduino debido a dos razones principales.

La primera razón es el precio y la escalabilidad, una Raspberry Pi cuesta 39,95€ mientras que una Arduino Nano cuesta 10€. Una Arduino Nano cuenta con menos pines de E/S, pero añadir una placa es más barato y sencillo que añadir una placa de Raspberry Pi.

La segunda razón es la implementación del programa que se encarga de el sensor de presión. En Internet se pueden encontrar ejemplos y tutoriales refiriéndose a cómo implementar el sistema en una Arduino, pero no es tan fácil encontrar información para hacerlo desde una Raspberry Pi.

Por estas razones se elige realizar la recepción de las señales del sensor de presión desde la Arduino, haciendo el proceso más sencillo y más barato.

5.4.1. Conexión

Para realizar la conexión de los sensores se utilizan cables de protoboard conectados de la forma explicada en la figura 5.9:

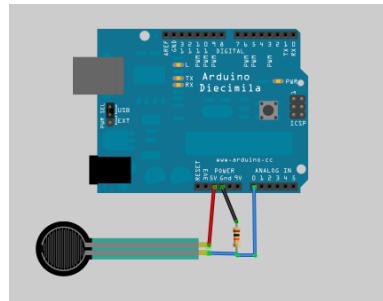


Figura 5.9: Esquema de conexión de sensores de presión [1]

Capítulo 6

Implementación de la propuesta

Este capítulo está dedicado a la explicación de cómo se ha desarrollado el proyecto. Se detalla la forma en la que se reproduce el sonido y cómo se hace de forma paralela, se explican los principales problemas encontrados en el desarrollo y se comenta el presupuesto y el tiempo empleado.

6.1. Reproducción de sonido

La reproducción del sonido se realiza utilizando las dos bibliotecas mencionadas en el capítulo anterior 5.3.1, mpg123 y libao.

El proceso completo que se sigue desde que se pulsa uno de los sensores hasta que se reproduce el sonido es el siguiente:

1. Uno de los sensores de presión realiza una lectura y manda el valor leído a la placa Arduino.
2. El Arduino, prepara un mensaje que escribe en el log del monitor serie (6.2).
3. La función `readSerial` lee de `/dev/ttyACM0`, la salida del monitor serie, filtra que no sea un mensaje vacío o de que no hay que reproducir nada y se manda a parsear.
4. En `parseInstruments` se procesa el mensaje leído.
5. En la función `PressToPlay` se selecciona qué instrumento debe sonar.
6. Finalmente, en la función `play`, se descodifica el archivo de sonido correspondiente y se reproduce utilizando las dos bibliotecas anteriormente mencionadas.

El flujo del programa es el representado en la figura 6.1:

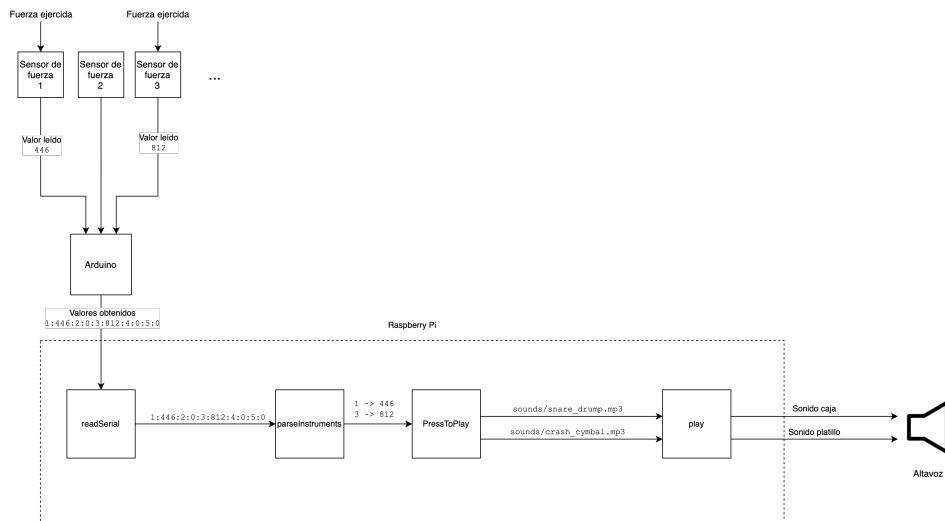


Figura 6.1: Diagrama de flujo

6.2. Sonido en paralelo

Hasta febrero, todas las versiones del programa fueron diseñadas pensando solo en emitir un sonido cada vez, pero lo preferible en un proyecto como este es que se emita más de un sonido al mismo tiempo.

El primer acercamiento fue mediante un sistema de máximos y sonidos combinados. En esta primera solución, se escogía el mayor de los seis sonidos y, si había un segundo sonido con un valor mayor que 200 (el mínimo para emitir sonido), se enviaba un mensaje a través del log de Arduino seleccionando un sonido combinado. Estos sonidos habían sido combinados previamente utilizando los sonidos ya presentes en el proyecto.

Finalmente, esta solución no funcionó correctamente (no se enviaba la señal del sonido combinado) y se procedió a diseñar una nueva. Esta nueva solución es la que se utiliza actualmente en el proyecto. Consiste en mandar todas las señales al mismo tiempo. Antes de la implementación de esta solución, el log era así:

```

0:0
0:0
2:364
0:0
0:0
  
```

En este mensaje, se indica que el pad 2 registra una presión de intensidad 364 (sobre 1023). Los 0:0 indican que ningún pad registra presión.

Tras la implementación de la solución, el log es así:

```
0:0
0:0
1:446:2:0:3:812:4:0:5:0
0:0
0:0
1:0:2:0:3:0:4:902:5:0
0:0
```

En estos mensajes se envía la siguiente información:

1. Los mensajes de 0:0 indican que ningún pad registra presión.
2. El primer mensaje indica que se han tocado:
 - El pad 1 con intensidad 446
 - El pad 2 no registra presión
 - El pad 3 con intensidad 812
 - El pad 4 no registra presión
 - El pad 5 no registra presión
3. El segundo mensaje indica que se han tocado:
 - El pad 1 no registra presión
 - El pad 2 no registra presión
 - El pad 3 no registra presión
 - El pad 4 con intensidad 902
 - El pad 5 no registra presión

En cuanto un sensor detecta una presión mayor a 200, se manda el mensaje de todos los sensores al mismo tiempo. En caso de que el valor leído sea menor que 200, se manda como 0, pero si es mayor, se manda con los demás. Este mensaje es leído y procesado por la Raspberry Pi, que lanzará los procesos necesarios con los sonidos que hagan falta según los sensores que hayan sido presionados.

Con esta segunda propuesta, el programa envía correctamente la señal de todos los sensores y los sonidos son emitidos correctamente.

6.3. Principales problemas

6.3.1. Número de lecturas en botones

En el prototipo con botones, al pulsar o dejar pulsado un botón, el programa reproduciría el mismo sonido muchas veces. Para solucionar esto se crea una hebra por cada botón, cuando se pulsa, entrará en un bucle infinito del que no saldrá hasta que el botón no sea soltado. Al usarse hebras, nos permite pulsar más botones al mismo tiempo.

6.3.2. Hebras vs procesos

La primera aproximación a cómo realizar este proyecto fue usando hebras POSIX para lanzar los sonidos al mismo tiempo, por ser más ligeras, en cuanto a uso de recursos, que los procesos lanzados por `fork`. Sin embargo, debido a la implementación de biblioteca utilizada para reproducir sonidos, al lanzar estas hebras para reproducir varios sonidos al mismo tiempo, se producía un error de *segmentation fault*. La solución a este problema fue sustituir las hebras POSIX por procesos generados por la llamada a `fork()`.

6.3.3. Número de lecturas en sensores

El sensor de presión devuelve muchas lecturas por segundo. Para solucionar esto a la hora de reproducir los sonidos hay dos formas de solucionarlo: una es introduciendo un *delay* lo suficientemente grande para diferenciar dos toques del sensor, la otra solución, que ha sido la implementada, trata de bloquear el sensor cada vez que se entra en uno de los tres intervalos de volumen que se han elegido. Cada vez que entra en uno de estos intervalos de deja de leer hasta que no baje la presión lo suficiente. Si la presión sube, tampoco enviará señal para que reproduzca sonido.

6.3.4. Construcción de paths

Al añadir el sensor y el Arduino, el programa que controlaba los sonidos emitidos recibía las mediciones del Arduino y, dependiendo de los datos entregados por éste, se emite un sonido a un volumen concreto. La construcción de la cadena de texto que contenía el *path* se hacía mediante las funciones de copia y concatenación *strcat* y *strdup*. El problema es que al recibir el *path*, la biblioteca de reproducción de sonidos lanzaba el siguiente error:

```
malloc(): corrupted top size
make: *** [Makefile:19: run] Segmentation fault
```

Tras muchas pruebas, como aumentar la cantidad de memoria reservada para el *path* o para el buffer que se utiliza en la función de reproducción, o probar a que siempre se enviara el mismo path, sin leer del Arduino (reproduciendo el sonido satisfactoriamente), finalmente se decide cambiar la forma en la que se genera el path, *hardcodeándolo* en el programa. Esto resulta funcionar y es la solución que ha sido implementada en el programa.

6.4. Pads

Los pads son las superficies que son golpeadas para generar los sonidos. Para este proyecto se han fabricado usando madera, cola y gomaeva [39]. El coste total de fabricar un pad es de 1,54€. Comparado con otros productos similares como los de Prologix [28], cuyo kit de práctica de 4 pads cuesta \$224,99, el precio de la solución propuesta en este proyecto es sensiblemente inferior.

6.4.1. Madera MDF

La madera utilizada para fabricar los pads es de tipo MDF. Este tipo de madera se refiere a tableros fabricados utilizando fibras de madera y resinas sintéticas para aportar una mayor resistencia [30].



Figura 6.2: Imagen de tablero MDF

6.4.2. Construcción

Los pads están hechos a partir de un tablero de madera MDF de 600x300x10 mm. Este tablero se cortó en un principio en dos círculos de unos 20 cm de diámetro, con la intención de tener unos sensores de fuerza lo suficientemente grandes como para tener sensibilidad de la mayor parte de estos pads. Sin embargo, los sensores que finalmente se han usado en el proyecto son más pequeños de lo esperado inicialmente, de 4x5,5 cm. Debido a esto, se tuvieron que recortar nuevos pads más pequeños, para que los sensores cubrieran la mayor área del pad posible. Estos nuevos pads contaban con unos 12 cm de diámetro. El resultado después de recortarlos es el indicado en la figura 6.3.



Figura 6.3: Imagen de los pads una vez recortados

Tras ser recortados, el siguiente paso es añadir agujeros en el pad para que pasen los cables del sensor. Estos agujeros hacen que, cuando se golpea el pad con la baqueta, se reduzca la posibilidad de dañar los cables debido a los golpes. El resultado es el de la figura 6.4



Figura 6.4: Imagen de los pads y un sensor tras añadir los agujeros

Una vez hecho eso, se añade la gomaeva para proteger los sensores y dar una mayor sensación de realismo al ser golpeado con la baqueta.



Figura 6.5: Imagen de los pads con la gomaeva

Finalmente, solo queda añadir los pads a la estructura para tener la batería montada.



Figura 6.6: Imagen de la batería terminada

6.5. Tiempo

Para medir el tiempo que ha llevado realizar este trabajo, se ha utilizado la aplicación Clockify [8].

En total se han utilizado 173 horas y 50 minutos. De estas horas, la división del tiempo ha sido la siguiente:

- Código: 102 horas y 5 minutos
- Memoria: 44 horas y 36 minutos
- Construcción: 14 horas y 52 minutos
- Presentación: 9 horas y 55 minutos
- Reuniones con el tutor: 2 hora y 20 minutos

6.6. Coste y presupuesto

6.6.1. Mano de obra

Teniendo en cuenta las horas trabajadas que se han comentado en la sección anterior (6.5), y el sueldo de un ingeniero técnico en Granada, se hará una estimación de cuánto costaría la mano de obra de este proyecto.

Un ingeniero técnico recién salido de la universidad cobra alrededor de 6,81€/hora. Teniendo esto en cuenta, se puede determinar que el coste de la mano de obra sería de unos 1184,26€.

6.6.2. Materiales

- Dos hojas de goma eva: 1,20€
- Tres tablas madera MDF 600x300x10 mm: 7,47€
- Cables protoboard: 2,60€
- Sensor de fuerza Exing c18.3: 5,91€
- Cinco sensores de fuerza Exing RP de S40: 50,95€
- Raspberry Pi 3B: 39,95€
- Tarjeta micro SD 8 GB: 5€
- Caja Aukru + cable alimentación: 11,99€
- Arduino Nano: 15,43€ (incluye gastos de envío)

El coste de los materiales se queda, finalmente, en 140,50€.

Capítulo 7

Conclusiones

Llegamos al final del proyecto. Durante los meses que se han invertido en la realización del trabajo, se han aprendido varias tecnologías y áreas.

Se ha aprendido a conectar y programar sensores a placas Raspberry Pi y Arduino. Siguiendo con este tema, se ha visto cómo conectar los dos sistemas, para que la Arduino envíe información a la Raspberry Pi para el tratamiento de los datos recogidos por los sensores instalados en la Arduino.

Por otro lado, se ha conseguido a reproducir archivos de sonido en entornos Linux/Unix de manera eficiente y simultánea utilizando el lenguaje de programación C.

Finalmente, se ha alcanzado un resultado que cumple con los objetivos que se marcaron al comienzo del trabajo. Utilizando lo aprendido anteriormente, se logra una experiencia de tocar la batería parecida a la realidad, incluyendo la construcción de una batería física que poder tocar.

7.1. Trabajo futuro

Durante la realización de este proyecto, ha surgido una serie de ideas que mejorarían la experiencia de tocar la batería, pero que, por falta de tiempo, no ha sido posible desarrollar. Las más destacables son:

- **Creación de una interfaz web:** Esta interfaz web podría utilizarse para arrancar y pausar la batería de forma sencilla, sin necesidad de entrar a la terminal para ello.
- **Sonidos personalizados:** Añadir un apartado a la interfaz web que permitiera al usuario añadir sus propios sonidos personalizados para configurar la batería a su gusto.
- **Aprender canciones:** Mediante una serie de LEDs, que se encienden en el momento preciso, se añadiría una canción a la batería y el usuario aprendería a tocarla de forma sencilla. Encendiendo el LED del pad que haya que tocar y apagándolo cuando haya sido tocado.

- **Guardar canción tocada:** Grabar lo que toque el usuario durante una cantidad limitada de tiempo en un archivo de audio para poder reproducirlo más tarde.

Bibliografía

- [1] Adafruit. Connecting to an fsr. <https://learn.adafruit.com/force-sensitive-resistor-fsr/connecting-to-an-fsr?view=all>. Consultado por última vez el 15/03/2020.
- [2] Arduino AG. Arduino compare. <https://www.arduino.cc/en/Products/Compare>. Consultado por última vez el 02/04/2020.
- [3] Arduino AG. Arduino introduction. <https://www.arduino.cc/en/Guide/Introduction#>. Consultado por última vez el 02/04/2020.
- [4] Arduino AG. Arduino nano. <https://store.arduino.cc/arduino-nano>. Consultado por última vez el 05/05/2020.
- [5] Inc. Apple. Cómo utilizar bucles libres de derechos en garageband con trabajos comerciales. <https://support.apple.com/es-es/HT201808>, feb 2015. Consultado por última vez el 12/04/2020.
- [6] DELVY ASESORES. La ley de propiedad intelectual y los derechos de autor en españa. <https://delvy.es/espana-la-propiedad-intelectual/>. Consultado por última vez el 26/04/2020.
- [7] Ian Buckley. 6 best arduino alternative microcontrollers. <https://www.makeuseof.com/tag/best-arduino-alternative-microcontrollers/>, feb 2018. Consultado por última vez el 22/04/2020.
- [8] Clockify. Clockify. <https://clockify.me>.
- [9] Ministerio de Cultura. Real decreto legislativo 1/1996, de 12 de abril... <https://www.boe.es/buscar/act.php?id=BOE-A-1996-8930&p=20190302>, apr 1996. Consultado por última vez el 23/10/2019.
- [10] Parlamento de Gran Bretaña. Estatuto de la reina anna. <http://www.copyrighthistory.com/anne.html>. Consultado por última vez el 24/09/2019.

- [11] Parijat Dutta. The 20 best raspberry pi alternatives available in 2020. <https://www.ubuntupit.com/best-raspberry-pi-alternatives/>, 2020. Consultado por última vez el 22/04/2020.
- [12] Fraunhofer IIS Erlangen. mp3. <https://www.iis.fraunhofer.de/en/ff/amm/consumer-electronics/mp3.html>. Consultado por última vez el 06/05/2020.
- [13] Unión Europea. Council directive 91/250/eec... <https://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31991L0250:EN:HTML>, 1991.
- [14] Raspberry Pi Foundation. Raspberry pi 3 model b. <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>. Consultado por última vez el 05/05/2020.
- [15] Raspberry Pi Foundation. Raspberry pi hardware. <https://www.raspberrypi.org/documentation/hardware/raspberrypi/README.md>. Consultado por última vez el 01/04/2020.
- [16] Raspberry Pi Foundation. Spi - software. <https://www.raspberrypi.org/documentation/hardware/raspberrypi/spi/README.md#software>. Consultado por última vez el 23/05/2020.
- [17] Oscar González. Cómo hacer un smart mirror con raspberry pi conectado con amazon alexa. <https://blog.bricogeek.com/noticias/raspberry-pi/como-hacer-un-smart-mirror-con-raspberry-pi-conectado-con-amazon-alexa/>, aug 2018. Consultado por última vez el 26/03/2020.
- [18] Gordon Henderson. Wiring pi. <http://wiringpi.com>. Consultado por última vez el 4/10/2019.
- [19] Alejandro Alcolea Huertos. La raspberry pi está viviendo un momento histórico: se están vendiendo más unidades que nunca. <https://computerhoy.com/noticias/tecnologia/ventas-raspberry-pi-momento-historico-coronavirus-621745>, apr 2020. Consultado por última vez el 23/04/2020.
- [20] Joel Lee. The 10 most common audio formats: Which one should you use? <https://www.makeuseof.com/tag/audio-file-format-right-needs/>, dec 2019. Consultado por última vez el 06/05/2020.
- [21] Taylor S. Marks. playsound. <https://github.com/TaylorSMarks/playsound/>. Consultado por última vez el 04/07/2019.

- [22] Marco Mascorro. pandemic-ventilator-2.0. <https://github.com/Mascobot/pandemic-ventilator-2.0>. Consultado por última vez el 23/04/2020.
- [23] Thomas Orgis Michael Hipp. Mp123. <https://www.mpg123.de/>. Consultado por última vez el 20/07/2019.
- [24] OMPI. ¿qué es el derecho de autor? <https://www.wipo.int/copyright/es/>. Consultado por última vez el 24/09/2019.
- [25] Edisson Oroxom. Batería – historia, partes, tipos y clases, precios y más. <https://instrumentosmusicales10.net/bateria>. Consultado por última vez el 29/04/2020.
- [26] Alfredo Pascual. Dos millones de razones para saber qué es exactamente raspberry pi. https://www.elconfidencial.com/tecnologia/2013-11-22/dos-millones-de-razones-para-saber-que-es-exactamente-raspberry-pi_56003/, nov 2013. Consultado por última vez el 26/03/2020.
- [27] AKAI Pro. Akai professional mpk mini mkii. <https://www.akapro.com/mpk-mini-mkii>. Consultado por última vez el 21/04/2020.
- [28] Prologix. Drumset. <https://www.prologixpercussion.com/products-drumset>. Consultado por última vez el 30/10/19.
- [29] Ana Sofía Rivera. Instrumentos musicales digitales. <https://www.unprofesor.com/musica/instrumentos-musicales-digitales-3606.html>, feb 2020. Consultado por última vez el 14/04/2020.
- [30] Maderas Santana. Qué es y características de los tableros o madera mdf. <https://www.maderassantana.com/caracteristicas-tableros-madera-mdf/>. Consultado por última vez el 13/04/2020.
- [31] Karen Stackpole. How to build your own electronic kit. <https://drummagazine.com/how-to-build-your-own-electronic-kit/>, may 2012. Consultado por última vez el 02/04/2020.
- [32] Thomann. Resultado de búsqueda en tienda de música thomann. https://www.thomann.de/es/e-drum_sets.html?setViewMode=block&smcs=822050_100. Consultado por última vez el 25/03/2020.
- [33] Varios. libao. <https://xiph.org/ao/>. Consultado por última vez el 21/07/2019.

- [34] Rubén Velasco. Convierte tu raspberry pi en un centro multimedia con estos sistemas. <https://www.redeszone.net/2017/02/09/convierte-raspberry-pi-centro-multimedia-estos-sistemas/>, feb 2017. Consultado por última vez el 26/03/2020.
- [35] Wikipedia. Advanced audio coding. https://en.wikipedia.org/wiki/Advanced_Audio_Coding. Consultado por última vez el 06/05/2020.
- [36] Wikipedia. Apple lossless. https://en.wikipedia.org/wiki/Apple_Lossless. Consultado por última vez el 06/05/2020.
- [37] Wikipedia. C (programming language). [https://en.wikipedia.org/wiki/C_\(programming_language\)](https://en.wikipedia.org/wiki/C_(programming_language)). Consultado por última vez el 05/04/2020.
- [38] Wikipedia. Contu. <https://en.wikipedia.org/wiki/CONTU>. Consultado por última vez el 4/10/2019.
- [39] Wikipedia. Etilvinilacetato. <https://es.wikipedia.org/wiki/Etilvinilacetato>. Consultado por última vez el 24/09/2019.
- [40] Wikipedia. Midi. <https://en.wikipedia.org/wiki/MIDI>. Consultado por última vez el 23/04/2020.
- [41] Wikipedia. Ogg. <https://en.wikipedia.org/wiki/Ogg>. Consultado por última vez el 06/05/2020.
- [42] Wikipedia. Raspberry pi. https://en.wikipedia.org/wiki/Raspberry_Pi. Consultado por última vez el 29/03/2020.
- [43] Wikipedia. Visual studio code. https://en.wikipedia.org/wiki/Visual_Studio_Code. Consultado por última vez el 05/04/2020.
- [44] Wikipedia. Windows media audio. https://en.wikipedia.org/wiki/Windows_Media_Audio. Consultado por última vez el 06/05/2020.