UNIVERSIDAD DE VALLADOLID

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

TRABAJO DE FIN DE GRADO

GRADO EN INGENIERÍA DE TECNOLOGÍAS ESPECÍFICAS DE TELECOMUNICACIÓN MENCIÓN EN SISTEMAS ELECTRÓNICOS

Evolución de un prototipo de lector Braille digital en Raspberry

Autor:

Paula González Vuelta

Tutores:

Noemí Merayo Álvarez

Jesús M. Hernández Mangas

TÍTULO:	**Evolución de un Prototipo de lector Braille digital en Raspberry**
AUTOR:	**Dña. Paula González Vuelta**
TUTORES:	**Dña. Noemí Merayo Álvarez**
	D. Jesús M. Hernández Mangas
DEPARTAMENTO:	**Teoría de la Señal y Comunicaciones e Ingeniería Telemática**

TRIBUNAL

PRESIDENTE:	**Jesús M. Hernández Mangas**
VOCAL:	**Ramón de la Rosa Steinz**
SECRETARIO:	**Noemí Merayo Álvarez**
SUPLENTE:	**Jesús Arias Álvarez**
SUPLENTE:	**Juan Carlos Aguado Manzano**

FECHA:

CALIFICACIÓN:

Resumen

La tecnología avanza de forma exponencial y ha supuesto un gran salto en la calidad y el nivel de nuestras vidas. La tenemos tan al alcance de nuestras manos que es necesario aprovecharla para seguir mejorando cualquier ámbito. En este trabajo se sigue la línea continuista de diseñar un prototipo de lector Braille digital que acerque y facilite la lectura de cualquier libro a personas con discapacidad visual, independientemente de sus medios económicos. Es por ello, que se desarrolló un prototipo económico que ahora se mejorará su diseño para hacerlo un producto competitivo que esté a la altura de aquellos que se encuentran en el mercado actualmente. Por lo tanto, partiendo de un prototipo ya en funcionamiento, se analizará y actualizará, manteniendo sus características principales de funcionalidad implementadas.

Palabras Clave:

Raspberry, PCB, lector Braille digital, electrónica, hardware, software, motor paso a paso

Abstract

Technology increases exponentially and it has meant a great leap forward to improve our life standards. It is very close to us and that's the reason why we need to take advantage of it, because these changes suppose the opportunity to keep improving any area. Along the same lines as previous projects developed, we will try to update a digital Braille reader designed, making it as small as possible. The aim is to be able to develop a competitive and low-cost product, making it available for everybody. On the one hand, we will analyze the previous design to find the points we should improve. On the other hand, we will keep the main characteristics of the device implemented for the moment.

Keywords:

Raspberry, PCB, digital Braille reader, electronics, hardware, software, stepper motor

Agradecimientos

En primer lugar, debo agradecérselo a mi familia, que siempre ha estado ahí para apoyarme en mi carrera y ayudarme a cumplir todos mis sueños y sin ellos nunca hubiera tenido la oportunidad de llegar hasta donde he llegado.

También mencionar, sin duda alguna, a mis tutores del TFG Jesús y Noemí, que me han ayudado a desarrollar este trabajo guiándome de la mejor forma posible, y han estado ahí para todo, gracias a su dedicación y continua disponibilidad.

A la escuela de Ingenieros de Telecomunicación y a la Universidad de Valladolid, por haberme dado una formación tan completa durante estos 4 años, en especial, quería hacer una mención a los profesores del departamento de electrónica por todos los conocimientos que me han enseñado de un mundo tan complejo y apasionante como es la electrónica.

Por último, agradecer a mis amigos, tanto los de siempre como los que me ha dado la carrera, por todos los buenos momentos vividos y los que nos quedan y en especial, agradecer a Ángel por estar siempre ahí, mi apoyo incondicional durante estos últimos años.

Índice General

Índice de Figuras

Índice de tablas

1. Introducción

1.1 Motivación

En el mundo existen alrededor de 253 millones de personas con discapacidad visual. De esas personas, 36 millones presentan ceguera total y los 217 millones restantes presentan ceguera moderada o avanzada [1].

Gracias al lenguaje Braille, un sistema de 6 u 8 puntos en relieve, las personas con discapacidad visual cuentan con una herramienta válida que les permite leer y escribir [2]. Sin embargo, el acceso a libros es muy limitado, porque solamente el 5% de ellos se traducen a lenguaje Braille.

Gracias a mi Trabajo Fin de Grado, se podrá solucionar este problema, porque el propio dispositivo que se va a diseñar convertirá cualquier documento al sistema Braille. Además, el desarrollo de este trabajo es posible gracias a los avances de la ciencia y la tecnología, que están cada vez más presentes en nuestro día a día y son un pilar fundamental en cualquier ámbito.

Por ello, cuando se me propuso la idea de desarrollar este proyecto no me lo pensé. El motivo principal, sin duda, fue el poder aplicar los conocimientos adquiridos a lo largo de estos años en el grado de Ingeniería de Telecomunicaciones, con mención en sistemas electrónicos, en ayudar a los demás, una faceta que a mi siempre me ha caracterizado.

Gracias a las habilidades desarrolladas seré capaz de implementar las ideas que se plantean en este trabajo, poniendo en práctica todos estos nuevos conocimientos con los que cuento para desarrollar un dispositivo Braille digital lo más óptimo posible.

1.2 Objetivos

El objetivo principal de este trabajo consiste en la optimización de un prototipo Braille digital previamente diseñado. Siguiendo la línea continuista se realizará un nuevo diseño donde el objetivo principal que se buscará será la optimización del tamaño del dispositivo y la realización de un diseño ergonómico de fácil manejo, para convertirlo en un dispositivo "de bolsillo" que se pueda llevar siempre encima. Para ello, se realizarán mejoras en el diseño electrónico.

Se buscará sobre todo mejorar el hardware del dispositivo integrando todos los componentes en una placa de circuito impreso donde las interconexiones se realizan mediante pistas, así se evitará tener el cableado tan voluminoso que presenta el dispositivo actual, y se conseguirá reducir notablemente su tamaño. El software se adaptará para que sea compatible con estos nuevos cambios, teniendo siempre de referencia el código actual, para mantener las funciones ya implementadas en la medida de lo posible.

Se buscará solucionar el problema que presentan los electroimanes en cuanto al ruido y el consumo de potencia. Para ello, se caracterizarán distintos micromotores paso a paso y se tomará una decisión sobre cuál podría ser el más adecuado para incorporar en el diseño en sustitución de los electroimanes.

Por otro lado, se intentarán añadir nuevas funcionalidades al lector Braille mediante la implementación de audio con la idea de que sea un apoyo a la lectura.

Todos estos objetivos que se plantean pretenden mantener una de sus características más significativas, y que diferencia este prototipo de otros de su competencia: ser un diseño económico y al alcance de todos.

1.3 Fases del proyecto

A continuación, se plantea, paso a paso, la estructuración del proyecto para llegar al resultado final, con distintos apartados claramente diferenciados entre sí:

1. Introducción
2. Descripción técnica del proyecto
3. Herramientas utilizadas para el desarrollo y evaluación del proyecto
4. Diseño y construcción del hardware actualizado del dispositivo
5. Desarrollo de la actualización software
6. Análisis de resultados
7. Conclusiones y líneas futuras
8. Bibliografía

En su totalidad, se pretende describir en cada uno de ellos, de la forma más clara posible, los aspectos más importantes del proyecto.

1.4 Estructura del proyecto

En el Capítulo 1 se realiza una breve introducción de los objetivos y las fases del proyecto que se van a desarrollar y tratar a lo largo de la memoria.

En el Capítulo 2, se realiza un breve análisis de mercado y un estudio previo del lector Braille digital del que se parte, analizando las partes en las que es necesario poner un foco de mejora y aquellas que se pueden mantener en este nuevo diseño. Además, se describe la funcionalidad y los requisitos que se desean tener en la aplicación y se seleccionan los componentes que vamos a utilizar.

En el Capítulo 3, se mencionan los distintos programas utilizados para desarrollar tanto el hardware como el software del dispositivo; así como la instrumentación electrónica para evaluar y verificar que el diseño funciona correctamente.

En el Capítulo 4, se plantea la captura esquemática final del diseño y se explican las conexiones realizadas entre los componentes. También se explica el proceso de diseño, fabricación y montaje de la PCB y se comprueba el funcionamiento y, en caso de que existan errores, se intentarán solucionar sin que sea necesario fabricar la placa de nuevo.

En el Capítulo 5, se detallarán los cambios y actualizaciones que se utilizarán para desarrollar el software del dispositivo para esta nueva versión, el cual será realizado en lenguaje *Python 3*.

En el Capítulo 6, se analizan los resultados que presenta el producto final para ver si se han cumplido los objetivos pactados.

En el Capítulo 7, se plantean líneas futuras para seguir mejorando este producto y se hace una reflexión sobre el trabajo realizado planteando las conclusiones que se consideren relevantes.

Por último, se recoge la bibliografía utilizada que ha servido de apoyo para tratar todos los temas tratados en la memoria.

2. Descripción técnica del proyecto

2.1 Análisis del prototipo de partida

Siguiendo la idea, se han buscado y analizado los distintos dispositivos similares que se han desarrollado para ser usados como lectores Brailles digitales.

En el ámbito comercial podemos encontrar diferentes alternativas, todas ellas con un objetivo común: accesibilidad a gran cantidad de libros que se codifican en código Braille para su lectura, aunque cada uno de ellos cuenta con características que hacen a cada dispositivo único.

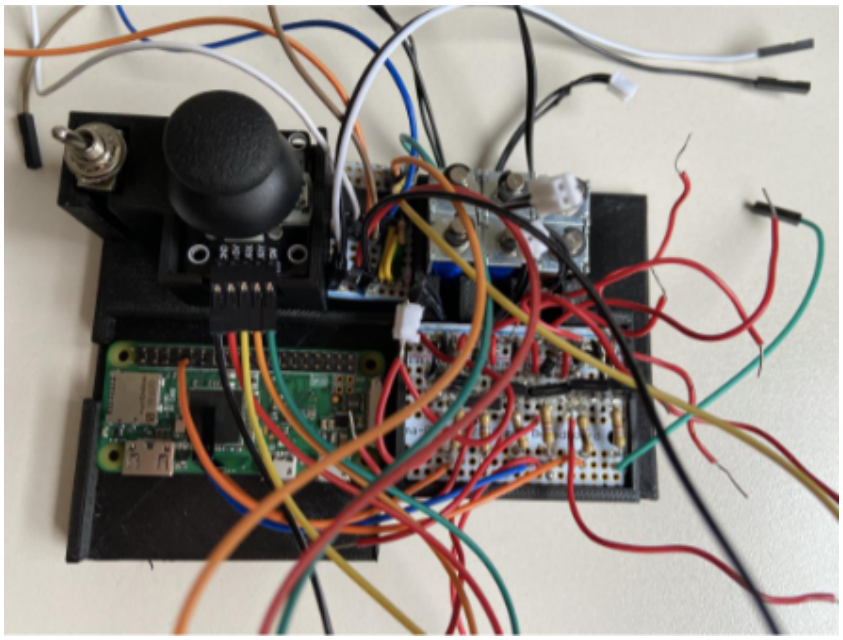
Entre ellos, el más destacado y el que se tomará como referencia debido a sus características de diseño es el lector *Braibook,* en la *figura 2.1.* Este modelo tiene un tamaño pequeño que hace que se pueda coger con una sola mano, lo que facilita su manejo, y cuenta con una celda Braille de 8 puntos. Entre las funcionalidades que ofrece están la posibilidad de escuchar audio y leer de forma simultánea, 5 horas de autonomía de la batería, conexión bluetooth y una memoria interna con capacidad de 8GB. Además, tiene un pequeño joystick en un lateral para navegar por el menú que ofrece opciones como subir y bajar la velocidad de lectura, subir y bajar el volumen de audio y avanzar o retroceder la lectura. Se puede obtener a un precio de 495€ [3].

Figura 2.1: *Lector Braibook*

2.2 Análisis del diseño previo del lector Braille digital

Como punto de partida, se necesita analizar las características del modelo del que partimos, para decidir cuales son los posibles puntos de mejora y, finalmente, decidir cuales serán los cambios que realizar. En la *figura 2.2* podemos ver el montaje del dispositivo, que tiene unas dimensiones de 10,8 cm de ancho, 14,6 cm del largo y 6,2 cm de alto [4].

La funcionalidad del lector Braille digital es la siguiente: consiste en un dispositivo que permite convertir un libro a codificación Braille y reproducirlo en tiempo real gracias al uso de 6 motores que emulan una celda Braille. Los libros se introducen mediante una memoria externa USB o se almacenan previamente en la memoria interna del dispositivo.



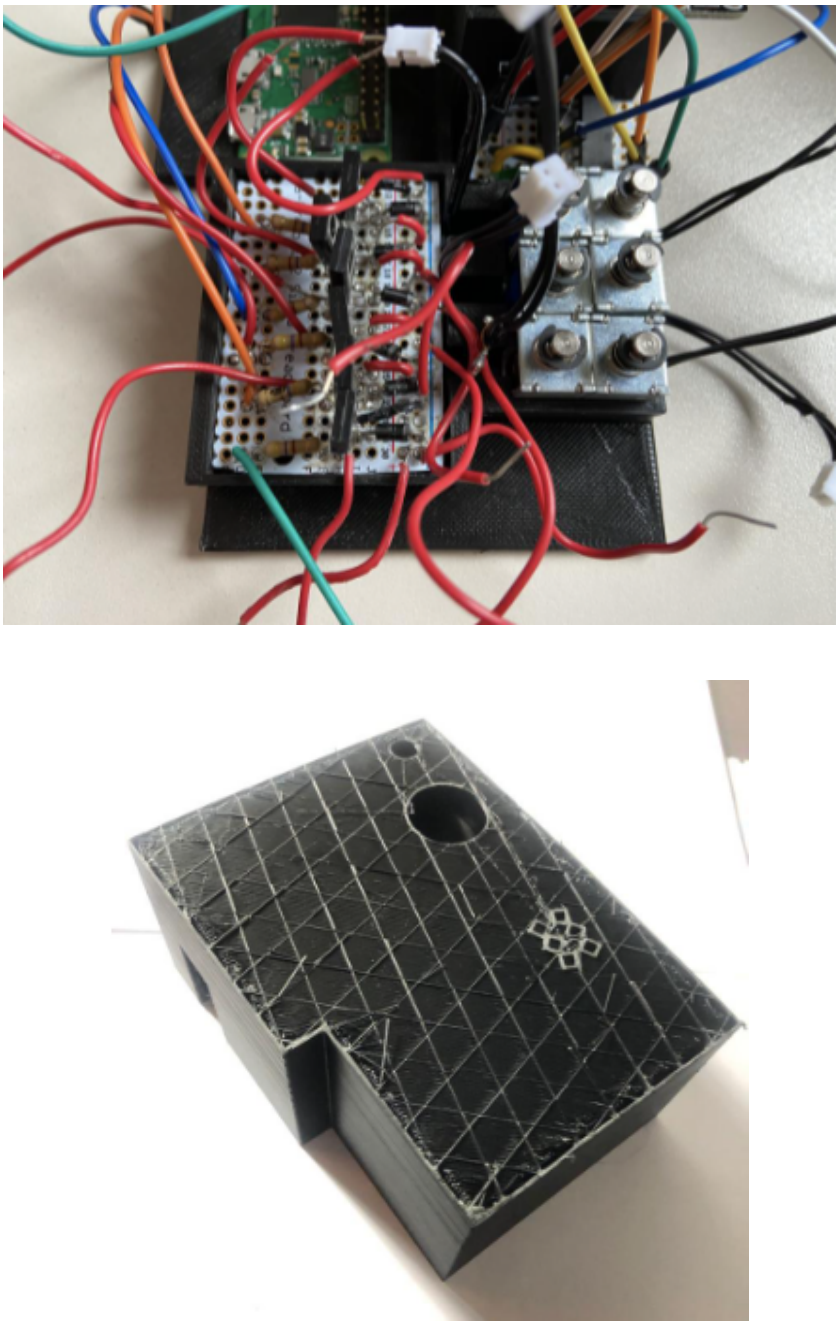


Figura 2.2: *Diseño previo del lector Braille (extraído del TFG anterior [4])*

El núcleo principal del prototipo es la *Raspberry Pi Zero W*, encargada de controlar y enviar las órdenes a los distintos componentes para que realicen sus funciones, y más adelante se comentarán las características más importantes que presenta. A mayores, el resto de los componentes que forman el dispositivo son:

- Un módulo de joystick modelo B07BWLJPC5
- Un conversor analógico digital modelo MCP3008
- Seis electroimanes modelo SPRK-ROB-11015
- Seis resistencias de 470Ω y una de 10kΩ
- Seis transistores modelo BD137 NPN
- Seis diodos
- Un interruptor
- Una batería externa modelo BOTKK mini batería

En este diseño encontramos tres partes claramente diferenciadas que se encargan de dar forma a las funcionalidades del lector digital:

- **Celda Braille:** Tiene la función de representar los distintos caracteres, uno a uno, durante la lectura. Consiste en 6 electroimanes que suben y bajan según el carácter que se va representando en cada momento. Como característica a destacar de estos motores encontramos que cada uno de ellos necesita una corriente mínima de 300 mA y una tensión de 5V, es decir, una potencia de 1,5W. Será un punto que mejorar en cuanto a tamaño y consumo de corriente ya que en el peor de los casos los electroimanes consumirán 1,8A si se encuentran funcionando de forma simultánea.
- **Módulo Joystick:** Permite moverse por las distintas opciones del menú y seleccionar la deseada. La comunicación de este módulo con la raspberry se realiza a través de un conversor analógico digital, porque esta no cuenta con entradas analógicas.
- **Alimentación:** Se implementó una batería externa para un dispositivo con autonomía. El encendido y apagado es controlado mediante un interruptor. Sin embargo, su función más destacada es que sirve para alimentar los electroimanes que emulan la celda braille, ya que la raspberry por si sola no es capaz de proporcionar la potencia suficiente para ello.

En cuanto al software del dispositivo, cuenta con dos versiones anteriores, la primera está desarrollada en *Python versión 2.7*, cuenta con 8 ficheros y ofrece 3 funcionalidades básicas al usuario seleccionables desde el menú principal [4]:

- Reanudación de la lectura en el punto en que se dejó.
- Elección del documento que se desea leer.
- Cambiar la velocidad de la lectura modificando la rapidez en la subida y bajada de los electroimanes.

La actualización del software se desarrolló en *Python versión 3* e incluye la novedad de que se pueden leer libros en formato PDF [5].

2.3 Estudio de las especificaciones

Cuando se pretende desarrollar cualquier producto hay una fase previa en la que se plantean las especificaciones que necesitará cumplir el diseño final. Algunas de ellas ya han sido mencionadas en apartados anteriores, pero es en este punto donde se van a detallar dichos requisitos de forma más extendida.

Como ya hemos visto, en el caso de este proyecto, se parte de un prototipo ya implementado, y lo que se busca es una mejora de este. Teniendo esto en cuenta, se analiza previamente el dispositivo para identificar aquellas características que se desean optimizar y los cambios en el diseño que ello supone, además de mantener aquellas características que si sean de nuestro interés.

En primer lugar, se debe tener en cuenta que el dispositivo tiene que proporcionar la potencia suficiente para ser capaz de representar los distintos caracteres cumpliendo ciertos requisitos temporales que permitan al usuario una lectura fluida. Esto significa que, en el peor de los casos, tendremos los 6 motores en movimiento de forma simultánea. Cada uno de estos motores va a consumir una corriente mínima cuando esté en funcionamiento, por lo que deberemos analizar la peor situación para adaptarnos a las necesidades requeridas.

A continuación, hay que asegurarse de que con el joystick se detecte correctamente el movimiento para navegar de forma adecuada a través de las distintas opciones del menú cuando se mueva el joystick arriba o abajo, permitir la selección de una de estas opciones para avanzar a otro submenú, en alguno de los casos, y permitir el retroceso al menú anterior si se mueve el joystick hacia la izquierda.

Además, en esta nueva versión a parte de mantener las funcionalidades antes mencionadas: continuar leyendo, escoger libro y cambiar velocidad, se busca añadir una nueva funcionalidad al prototipo: activar o desactivar el audio para seguir la lectura. Por ello, se debe incorporar un bloque de audio con salida desde la raspberry para que el dispositivo funcione también como un audiolibro. Para una mayor utilidad de esta nueva característica se permitirá tanto el uso de altavoz como de auriculares, para aquellas situaciones en las que resulte complicado seguir el audio a través del altavoz. En este bloque, se añadirá a mayores un potenciómetro para permitir al usuario el control del volumen del altavoz.

Se desea que el dispositivo sea portable, para ello, será necesario contar con un bloque de alimentación que incluya una batería y que también se permita su recarga. La carga de la batería se controlará mediante un led que indicará cuando se ha cargado del todo. Por otro lado, la capacidad de la batería tendrá que ser suficiente para que la autonomía sea razonable, por ello, se deberá analizar el consumo eléctrico de todos los componentes. Finalmente, el encendido y apagado del dispositivo se controlará mediante un interruptor que bloqueará o permitirá el paso de corriente desde la batería hacia el dispositivo.

2.4 Estudio del sistema y selección de componentes

Una vez se han establecido las especificaciones de nuestro diseño, se decidirán y se describirán las características principales de los componentes con los que contará nuestro sistema. Como ya se ha comentado, se mantendrán algunos componentes del diseño previo del que partimos, otros los modificaremos y, además, se añadirán

algunos nuevos. En la *figura 2.3* se puede ver un primer planteamiento de los distintos bloques con los que contará el diseño que se va a realizar.

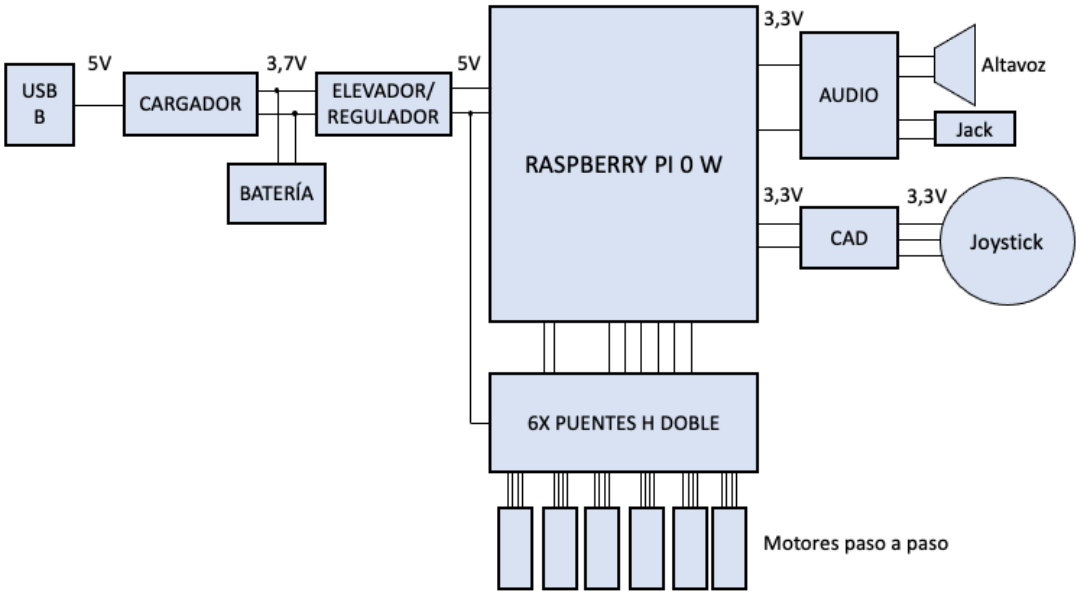


Figura 2.3: *Diagrama de bloques del lector Braille*

En los siguientes apartados se describirán con detalle cada uno de los bloques del lector Braille que aparecen en la *figura 2.3*.

2.4.1 Raspberry Pi 0 W

Como núcleo principal del lector Braille se mantiene la *Raspberry Pi Zero-W*, un microordenador que utiliza el sistema operativo Raspbian. Lo que diferencia a este modelo de la *Raspberry Zero* es que cuenta con conexión wifi y bluetooth, lo que hace que no sea necesario el uso de cables para la conexión a este pequeño ordenador. En su placa de circuito, *figura 2.4*, cuenta con un puerto USB, conexión mini HDMI y conexión micro USB para la alimentación. Además, incorpora una ranura microSD en la que se insertará la tarjeta con el sistema operativo instalado [6].

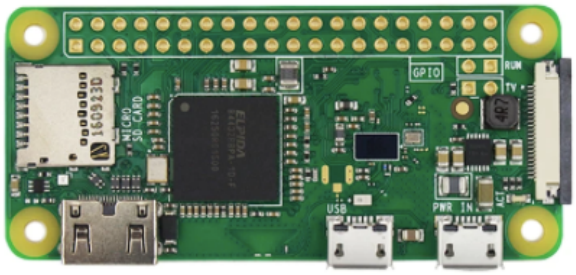


Figura 2.4: *Placa base de la Raspberry pi Zero W*

Algunas de las características más importantes de la *Raspberry Pi Zero W* son:

- LAN Inalámbrica que soporta los estándares 802.11 b/g/n.
- Bluetooth 4.1.
- CPU de solo un núcleo de 1GHz.
- Memoria RAM de 512Mb.
- Dimensiones: 65mm x 30mm x 5mm.

Por otro lado, cuenta con 40 pines que pueden desempeñar distintas funciones, entre las que se permite la comunicación mediante el bus SPI, GPIO o el protocolo I2C. La función de cada uno de ellos se describirá más adelante.

Un último aspecto que se podría destacar y que es otro de los motivos por los que se escogió este modelo es el hecho de ser el más pequeño y económico de la familia de las Raspberry Pi, algo clave para conseguir un producto final lo más competitivo posible.

2.4.2 Módulo de la celda Braille

Como el objetivo principal que se persigue es la optimización del tamaño, se buscará un diseño alternativo de la celda braille en el que se sustituyan los electroimanes por otro tipo de motores, más pequeños y menos ruidosos, que sean

capaces de adaptarse a los requerimientos de representar carácter a carácter en tiempo real. Como alternativa se presentan distintos modelos de motores a paso a paso.

Para controlar los motores paso a paso y permitir la comunicación entre estos y la raspberry usaremos dos puentes H doble para cada uno de ellos. A continuación, se explicará que es y para que sirve este tipo de circuito.

2.4.2.1 Motores paso a paso

Un motor paso a paso es un motor de corriente continua que realiza la rotación en un cierto número de pasos. Está formado por un cierto número de devanados, que son bobinas que al aplicar una corriente a través de ellas generan un campo magnético a su alrededor y hacen girar el rotor del motor, el principio de funcionamiento se puede ver resumido en la *figura 2.5*. En función de la dirección de la corriente a través de la bobina el giro será en un sentido u otro [7].

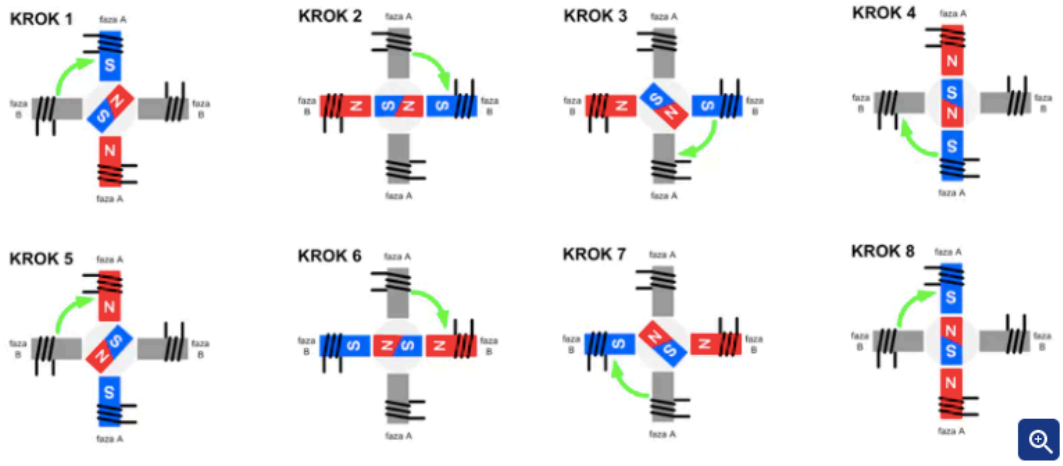


Figura 2.5: *Principio de funcionamiento de un motor paso a paso*

En este caso, no se ha escogido ningún motor en concreto, sino que se analizarán cuatro alternativas distintas para comparar las prestaciones y el consumo eléctrico que ofrecen cada uno de ellos. Las características comunes que presentan son que cuentan con 2 fases, 4 devanados y tienen polaridad; mientras que, lo que cambiará será la resistencia por devanado y el tamaño. Más adelante, se encuentra un apartado

donde se podrán ver los resultados de la caracterización de cada motor y se discutirán las ventajas e inconvenientes de cada uno de ellos.

Por el contrario, los electroimanes [8] consisten en una sola bobina que, al circular la corriente a través de ella, se activa el motor y cuando no circula vuelve a su estado de reposo inicial.

Los modelos de motores escogidos son: Mini 5mm Gear Stepper motor [9], 5MM Stepper Motor [10], Two-phase Four-wire 6MM Stepper Motor [11] y 3.3MM Stepper motor [12].

2.4.2.2 Puentes H

Un puente H consiste en un circuito formado por 4 interruptores conectados a una tensión de alimentación que permiten el giro del motor en un sentido u otro, dependiendo de la tensión que se le aplique. En las *figuras 2.6* y *2.7,* podemos ver la estructura básica de este circuito. Al cerrar S1 y S4, se aplicará una tensión positiva al motor y, por consiguiente, si se cierran S2 y S3, se aplicará una tensión negativa y se invertirá el sentido del giro [13].

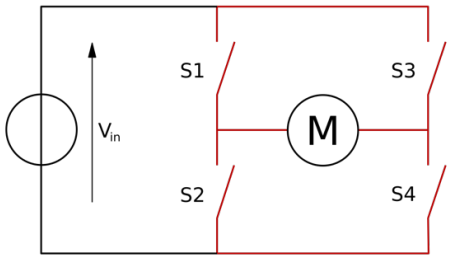


Figura 2.6: *Estructura de un puente H*

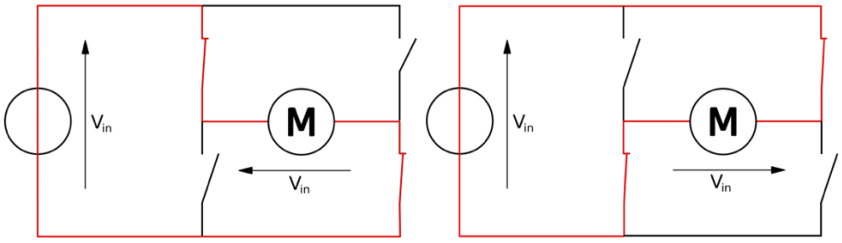


Figura 2.7: *Estados básicos de funcionamiento del puente H*

Como hemos comentado antes que se necesitan dos puentes H por cada motor, usaremos el chip DRV8847S [14] en el que vengan los dos incorporados y que sea la vía de conexión entre la raspberry y los motores, ya que este modelo es compatible con la interfaz I2C.

Otra característica es que soporta operaciones multi-esclavo, es decir, se pueden conectar al mismo bus varios chips. Por lo tanto, no será un problema usar un chip por cada motor y conectarlos a la raspberry.

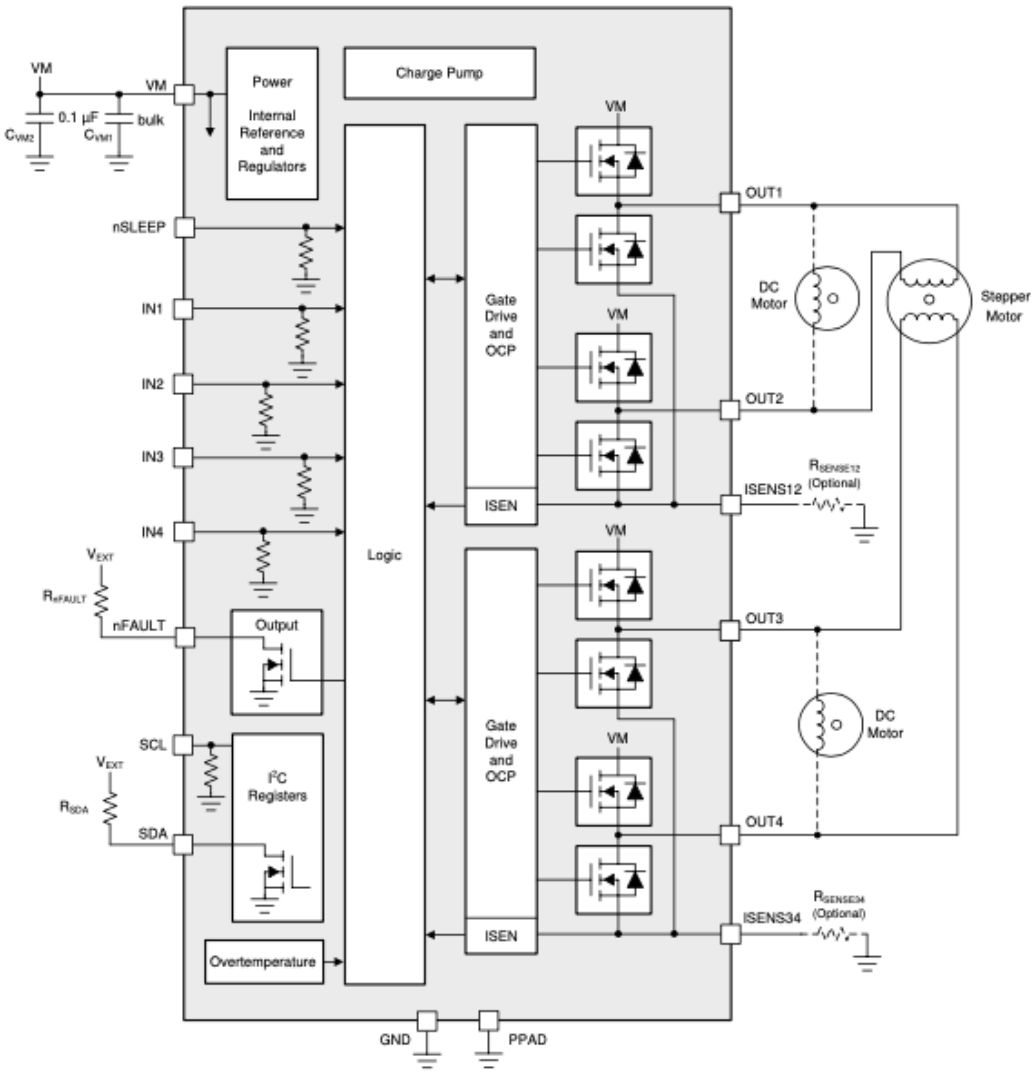


Figura 2.8: *Diagrama de bloques del DRV8847s*

2.4.3 Módulo Joystick

En este nuevo prototipo se ha cambiado el modelo del joystick por uno un poco más pequeño, similar al utilizado en la *playstation 4*, de la marca OTOTEC [15]. Este componente cuenta con 3 pines de datos que son los que indicarán la posición del joystick en cada momento: en el eje x, en el eje y y en el eje z, siendo este último pin digital y el resto analógicos. En la *figura 2.9*, se muestra el modelo de joystick de PS4 utilizado en el proyecto.

Recordamos que la raspberry pi solo cuenta con pines digitales, por ello, será necesario el uso de un convertidor analógico digital para la correcta comunicación.

Figura 2.9: *Joystick PS4*

2.4.3.1 CAD

Se escoge el chip MCP3004 [16] en lugar del MCP3008. Se ha tomado dicha decisión porque ambos tienen las mismas características y el software ya diseñado es completamente compatible con este nuevo chip. La única diferencia se encuentra en el número de canales analógicos de entrada con los que cuentan: 4 y 8 canales, respectivamente. Teniendo en cuenta que solo necesitaremos usar 3 de ellos, nos vale con usar el MCP3004. Este chip transformará las señales analógicas en señales digitales de 10 bits de precisión.

La comunicación con la raspberry es posible a través de una sencilla interfaz en serie que es compatible con el protocolo SPI y se pueden alcanzar tasas de conversión de hasta 200Ksps.

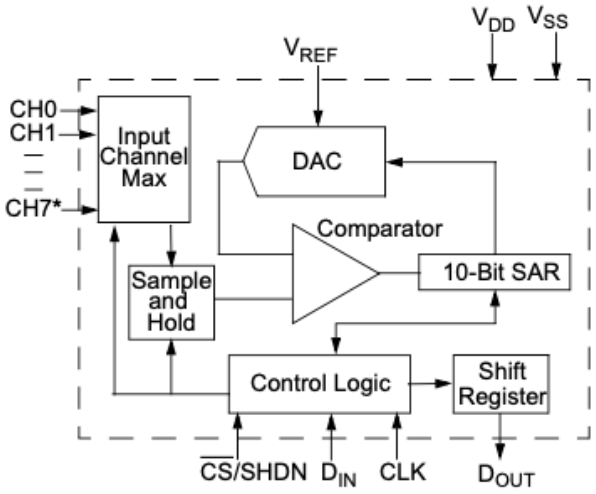


Figura 2.10: *Diagrama de bloques del MCP3004*

2.4.4 Módulo de Audio

El módulo de audio es una novedad en esta versión. Su implementación será posible usando señales PWM (Pulse-With Modulation), o de modulación por ancho de pulso, en las que, al modificar su ciclo de trabajo, podremos obtener audio. En la *figura 2.11* se muestra el circuito de audio al que se conectan los pines de audio de salida de la raspberry.

En la *Raspberry pi Zero W*, los pines PWM no están disponibles; sin embargo, realizando ciertas configuraciones podremos acceder al pin PWM0 y al PWM1.. El pin PWM0 se corresponde con el canal izquierdo y el PWM1 con el canal derecho [17].

Teniendo todo esto en cuenta, este bloque contará con los siguientes componentes:

- Un circuito formado por condensadores y resistencias, para filtrar la señal PWM de salida de la raspberry pi y juntar los dos canales, como paso previo a la amplificación.
- Un amplificador de audio, modelo LM4865MX/NOPB, al que se incorpora un potenciómetro para el control de volumen.

- Un altavoz, modelo KDMG28008, como salida de audio.
- Un conector Jack 3.5mm, modelo STX3000, también como salida de audio.

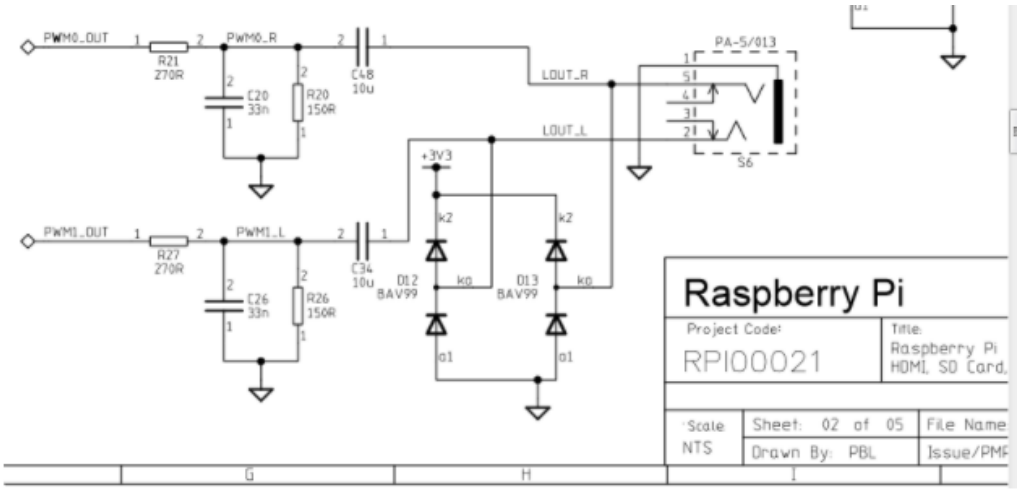


Figura 2.11: *Esquema Audio para Raspberry pi*

2.4.5 Alimentación

El núcleo de la alimentación seguirá siendo una batería, ya que es necesario mantener la característica de la autonomía del dispositivo. Como novedad frente a la versión anterior, se sustituye la batería externa modelo BOTKK Mini batería por una batería de litio de 2000mAh [18], *figura 2.12*.

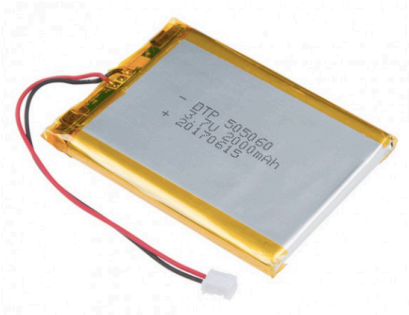


Figura 2.12: *Batería de litio 2000mAh*

Esta batería tiene una tensión nominal de 3,7V y, como la raspberry pi y los puentes H se alimentan con una tensión de 5V, será necesario un elevador de tensión

que nos permita obtenerla. Para ello, se selecciona el modelo RP401N501C-TR-FE [19], *figura 2.13,* un regulador lineal que proporciona a la salida una tensión fija, como en nuestro caso, o una tensión ajustable, en función del modelo elegido.

A la salida del regulador se obtiene una corriente máxima de 500mA, y como no es suficiente para hacer funcionar a los motores y a la raspberry pi se decidió añadir otro elevador de tensión. De este modo, se obtiene una tensión de 5V con uno de los reguladores para alimentar a la raspberry pi y otra tensión distinta también de 5V para alimentar al bloque de los motores y los puentes H.

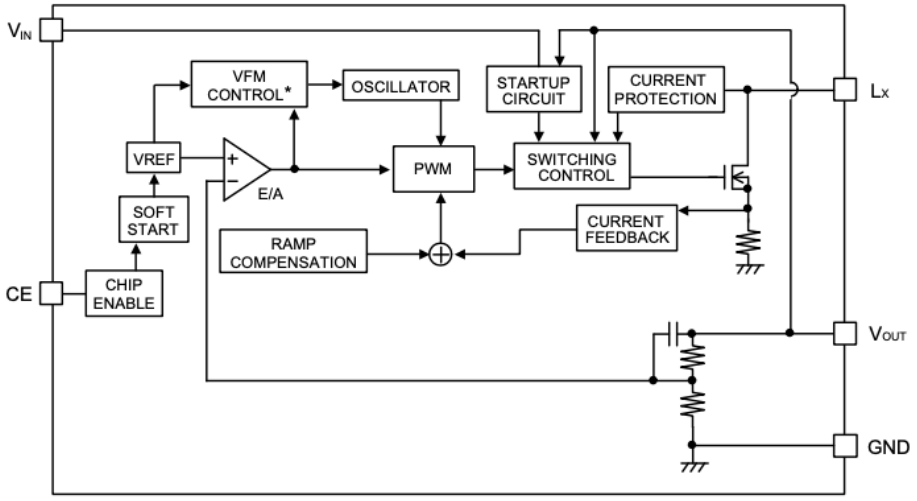


Figura 2.13: *Diagrama de bloques del RP401N501C-TR-FE*

En cuanto a la tensión de 3,3V, usada para alimentar el bloque de joystick y del audio, es proporcionada directamente de la raspberry pi, ya que esta cuenta con un regulador interno con una corriente máxima de 1A, suficiente para abastecer a todo el circuito.

A mayores, esta nueva versión cuenta con un cargador para la batería, en concreto se ha escogido el modelo MCP73831 [20], *figura 2.14*, válido para aplicaciones que cuentan con puerto USB para cargarse. En el caso concreto de esta aplicación, cuenta con un puerto mini USB tipo B hembra.

Finalmente, se incluye un led que cuando está encendido significa que el lector Braille está conectado a la alimentación y la batería se está cargando.

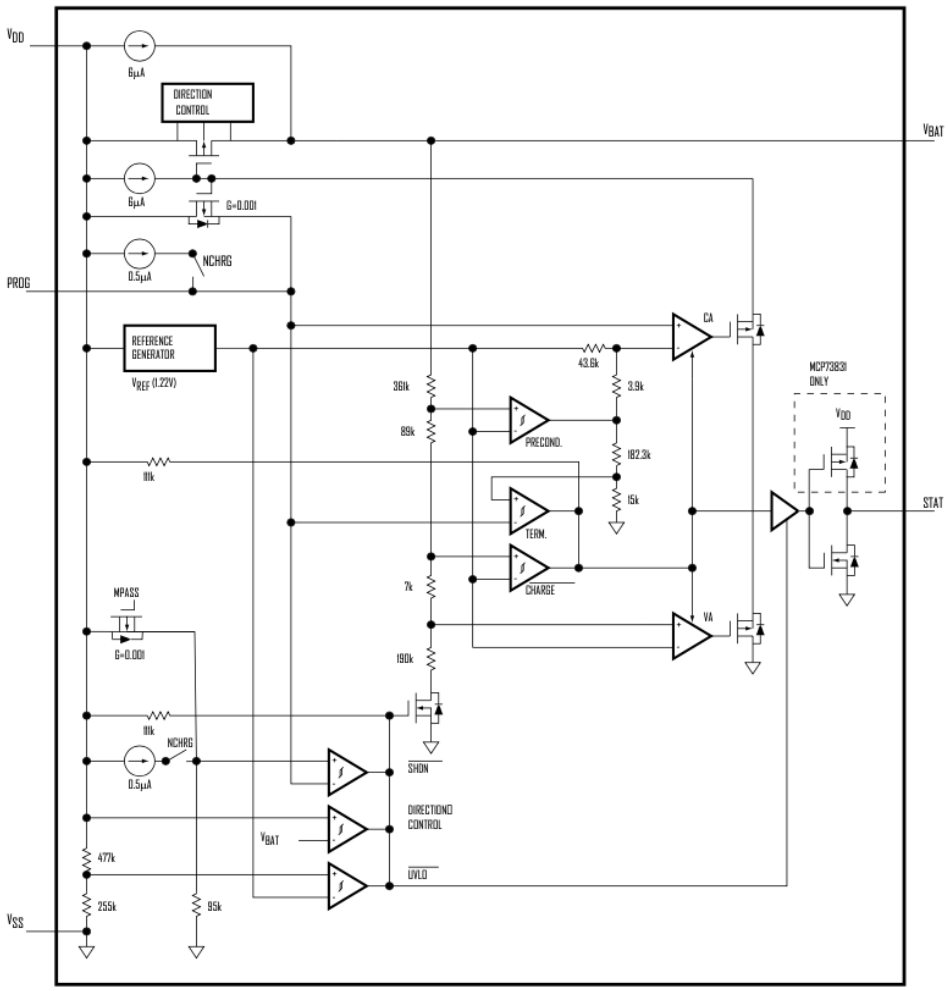


Figura 2.14: *Diagrama de bloques del MCP73831*

2.4.6 Componentes pasivos

Además de todos estos componentes mencionados, en cada uno de estos módulos se incluyen numerosos componentes pasivos como son las resistencias, condensadores, bobinas y diodos. Todos ellos se han usado para distintas tareas: condensadores de acoplo y de desacoplo, resistencias de *pull-up* y *pull-down* o los diodos para controlar el paso de corriente.

3. Herramientas utilizadas

3.1 Proteus 8

Proteus [21], *figura 3.1*, es un programa de diseño electrónico que nos permite realizar esquemáticos, simulaciones y el rutado de la placa de circuito impreso, entre otras cosas. En nuestro proyecto, aprovecharemos estas funcionalidades que ofrece para realizar un esquema completo con las conexiones entre los distintos componentes con Proteus ISIS. Una vez acabado y validado, se pasará a diseñar la PCB, donde se colocarán las huellas asociadas a cada componente y se realizará el rutado y se generarán los ficheros necesarios para su posterior fabricación, con Proteus ARES.

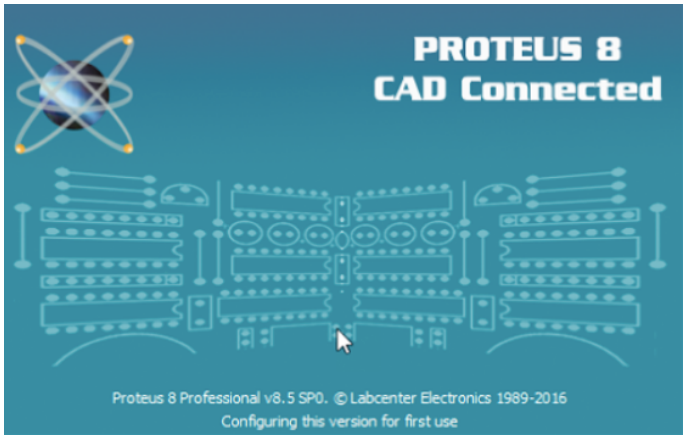


Figura 3.1: *Página de inicio de Proteus 8*

Todo esto es posible gracias a que este software cuenta con distintas librerías en las que se pueden encontrar multitud de componentes con sus huellas, y, además, permite la creación de nuestras propias bibliotecas para añadir aquellos componentes y huellas que no estén disponibles en la aplicación.

A parte de todo lo mencionado, también permite la generación de una lista de materiales en la que se encontrará información sobre cada uno de los componentes: el modelo, la cantidad, el código de referencia, la página donde podemos comprarlo, el coste, etc. Esta lista es una funcionalidad muy útil porque nos permite obtener el precio final del producto que se desarrolla.

3.2 VNC Viewer

VNC Viewer [22] es un programa con el que se controlará de forma remota la raspberry pi. Se podrá acceder a su escritorio a través de la IP que tenga asociada y se podrá controlar desde nuestro portátil. Será un programa de gran ayuda para desarrollar, ejecutar y probar el software del dispositivo y para realizar las configuraciones necesarias para poner a punto la raspberry pi para su uso.

Como paso previo a comentar para poder acceder a la raspberry pi, será necesario tener instalado el sistema operativo *Raspbian*, los pasos para su descarga e instalación se puede encontrar en la propia página de raspberry pi de forma totalmente gratuita [23]. Lo único que se necesita es una tarjeta microSD donde guardar toda la información e insertarla en la ranura con la que cuenta la *Raspberry pi Zero W.*

3.3 Tinkercad

Es un programa en línea gratuito que permite realizar modelos 3D y con el que podremos rediseñar la nueva carcasa del lector Braille digital [24].

Se ha escogido este programa frente a otros de su competencia porque es muy simple e intuitivo y no es necesario un conocimiento previo de diseño 3D para obtener un resultado final apropiado.

Una vez se tiene el modelo realizado, se permite exportar el diseño en formato STL e imprimirlo, en nuestro caso, con una impresora 3D.

3.4 Instrumentación electrónica

Por un lado, se ha usado un osciloscopio digital para visualizar las señales más relevantes y asegurarnos de que la forma de onda de las señales es la correcta.

Por otro lado, se ha utilizado un multímetro para medir la resistencia de los devanados de los motores y la tensión que tienen en funcionamiento o, también, para comprobar que se obtiene la tensión de alimentación correcta en los distintos puntos del circuito.

Para facilitar las mediciones, se colocaron en la PCB distintos puntos de test en los pines más importantes, ya que al tener todo un tamaño tan reducido puede resultar complicado colocar los terminales de estos aparatos en el lugar correcto. De este modo, se garantiza que estamos midiendo de forma adecuada y se evitan cortocircuitos derivados de tocar varias patillas a la vez.

4. Actualización del hardware del lector Braille digital

4.1 Captura esquemática del diagrama de Bloques

Se creará el diagrama de bloques utilizando Proteus 8.12, la versión más reciente hasta la fecha de este programa, y en las *figuras 4.1, 4.2, 4.3, 4,4 y 4.5* se recogen los resultados. Como ya se ha comentado, algunos de los componentes no se encuentran en las librerías y tendremos que diseñarlos manualmente, asociándoles una huella que se necesitará para diseñar la PCB. Para ello, utilizaremos como referencia las hojas de datos disponibles de cada uno de esos componentes para diseñarlos con las características requeridas.

En el caso del diseño esquemático, se realizará un bloque con el mismo número de pines con los que cuenta el componente a crear, siguiendo la enumeración que se indica en sus especificaciones.

En el caso de la huella, se diseñará con sus dimensiones reales y habrá que prestar especial atención a la colocación correcta de los pads, ya que un error de este tipo ocasionará problemas con las conexiones eléctricas y el dispositivo no funcionará.

En las siguientes páginas podemos ver el resultado final de nuestro diseño y a continuación se explicará cada uno de los bloques principales con los que cuenta este esquemático.

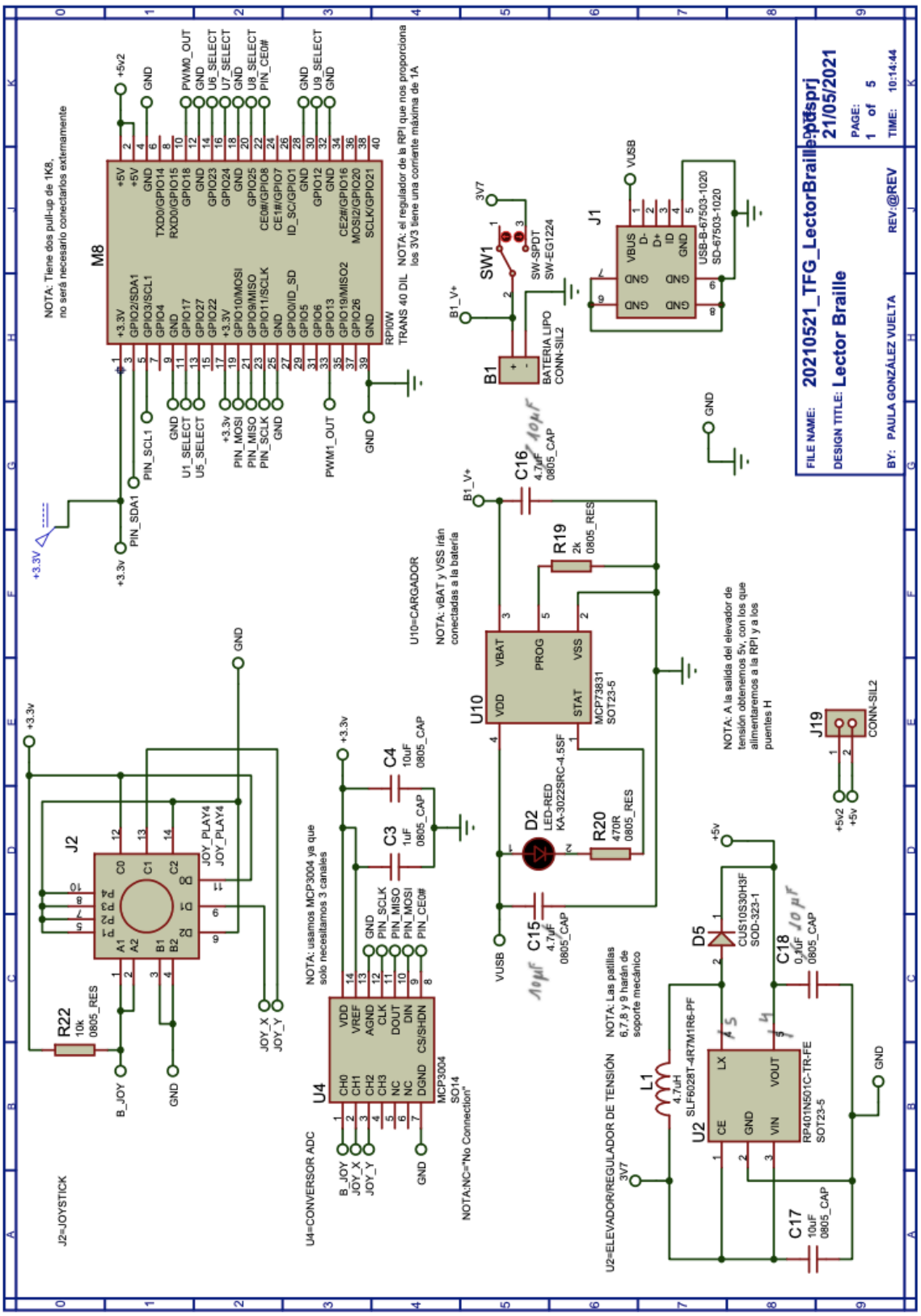


Figura 4.1: *Hoja 1 del esquemático, conexionado raspberry pi, bloque de la alimentación y bloque del joystick*

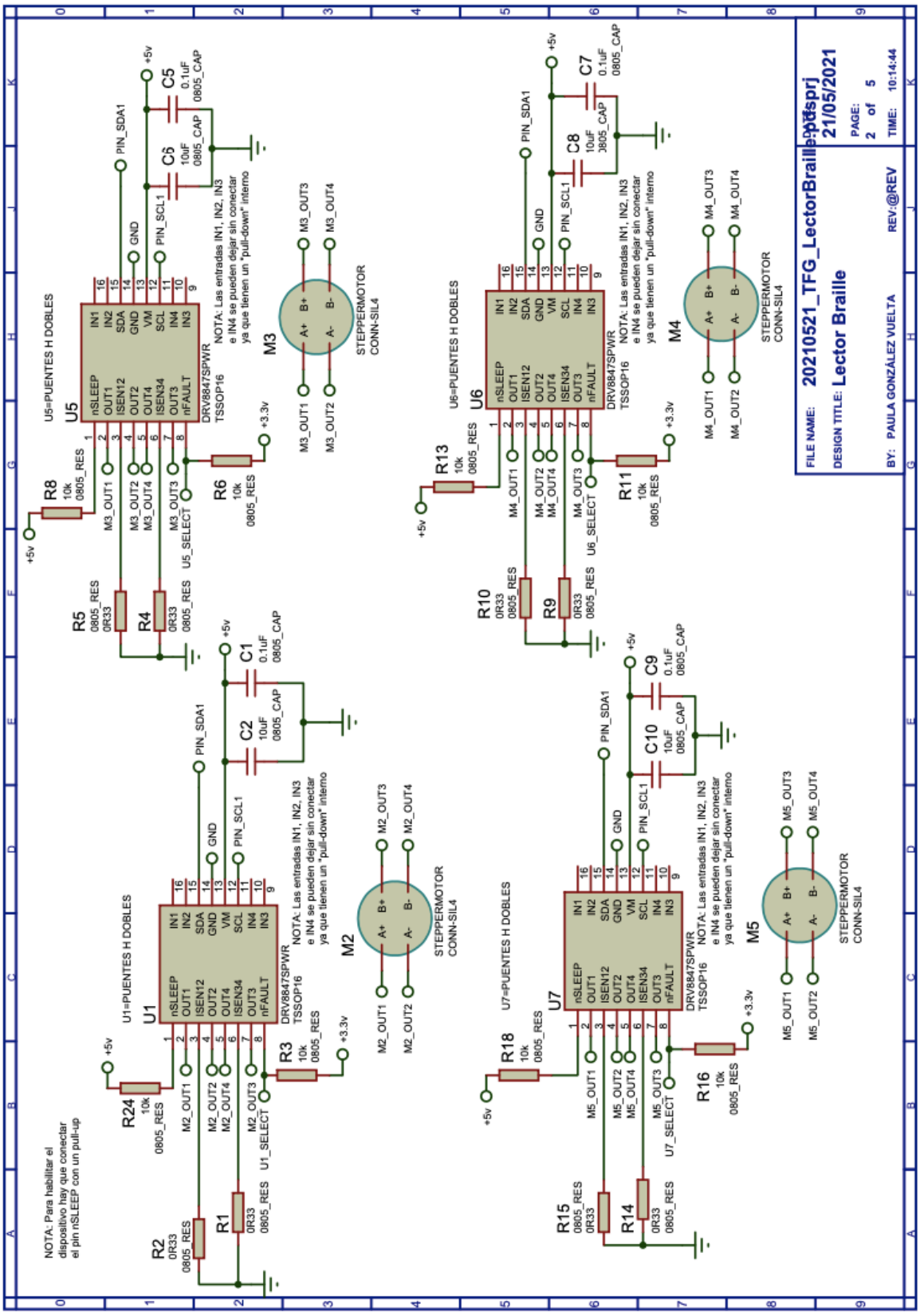


Figura 4.2: *Hoja 2 del esquemático, conexionado de puentes H y motores*

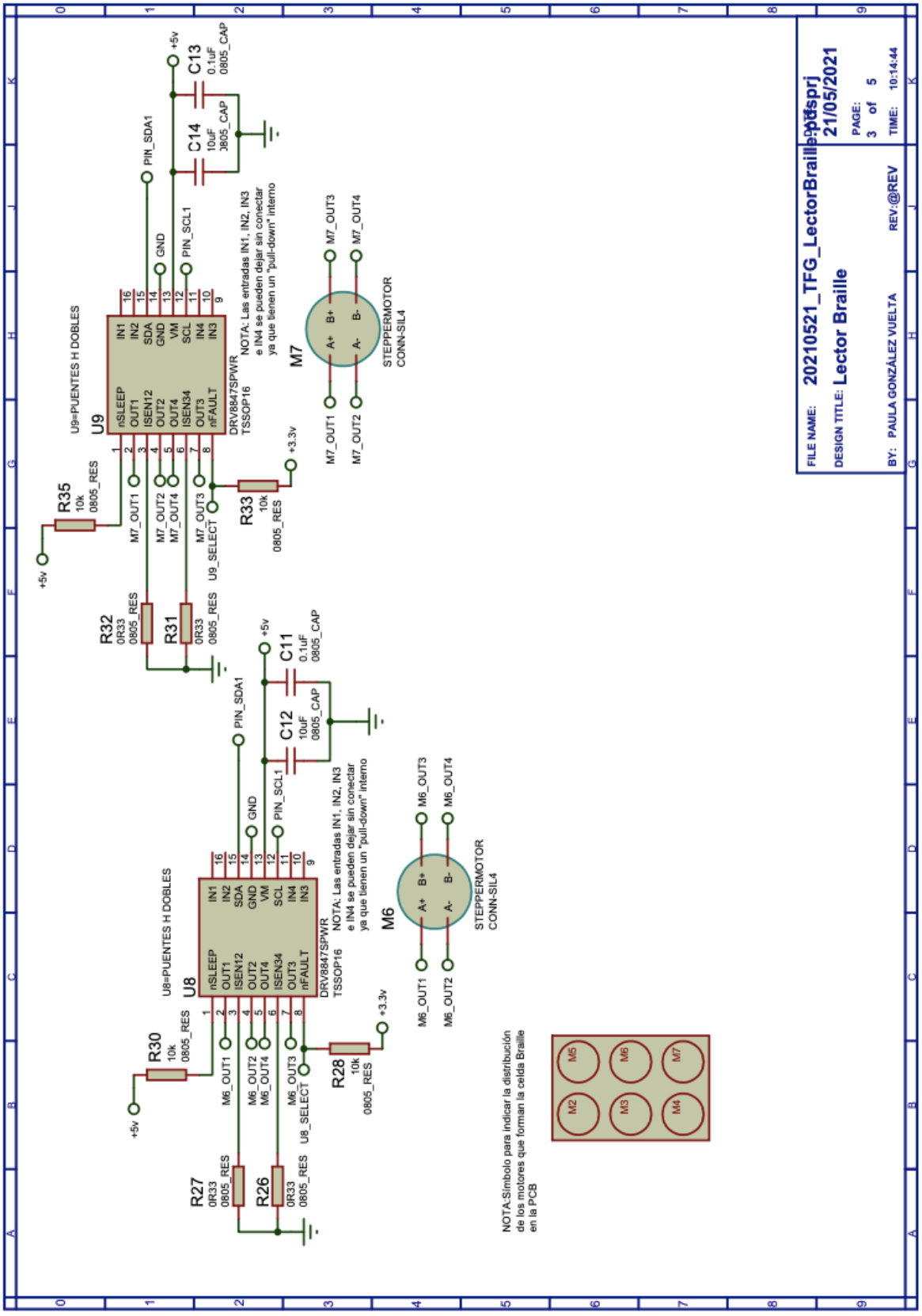


Figura 4.3: *Hoja 3 del esquemático, conexionado de puentes H y motores*

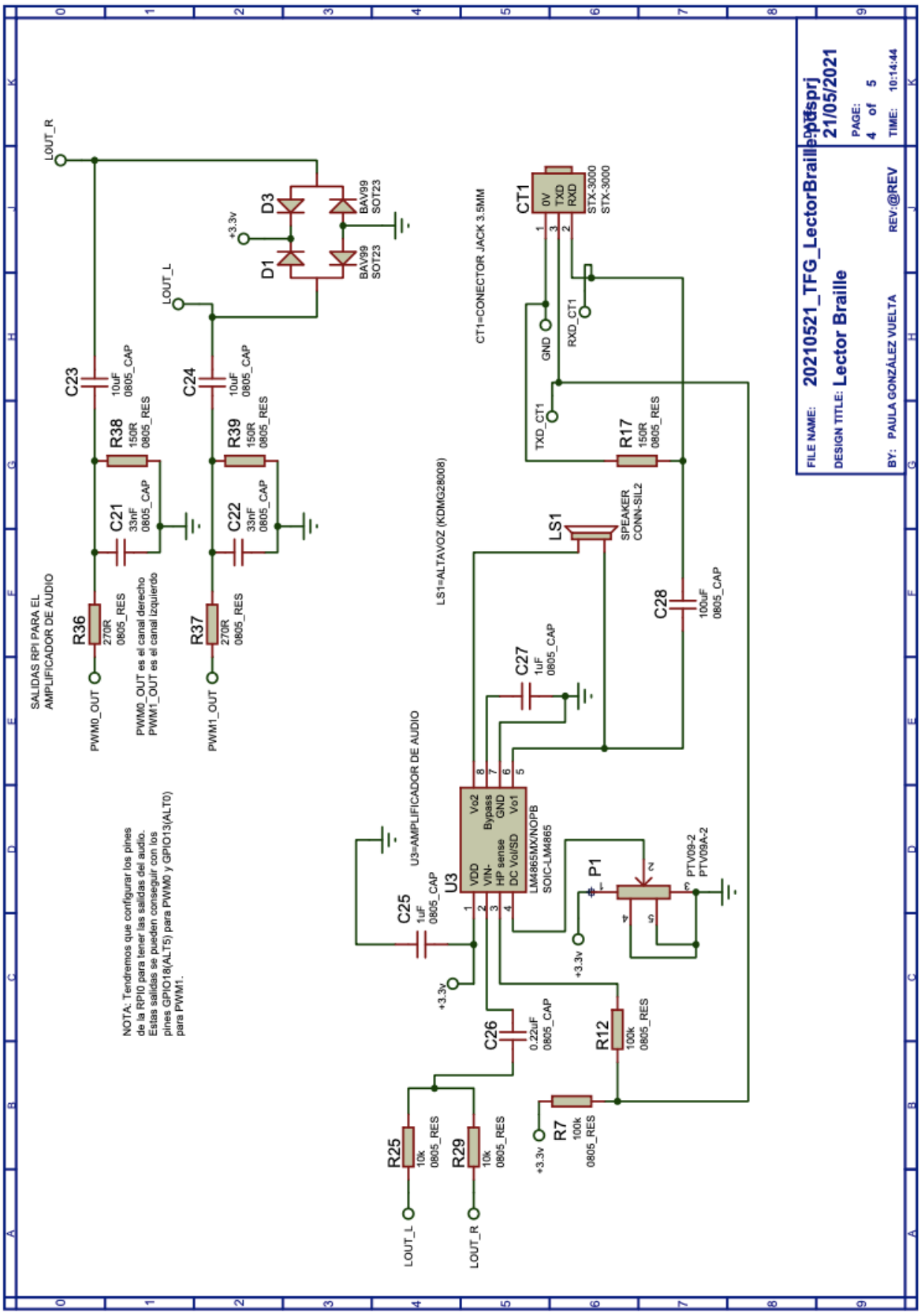


Figura 4.4: *Hoja 4 del esquemático, conexionado del audio*

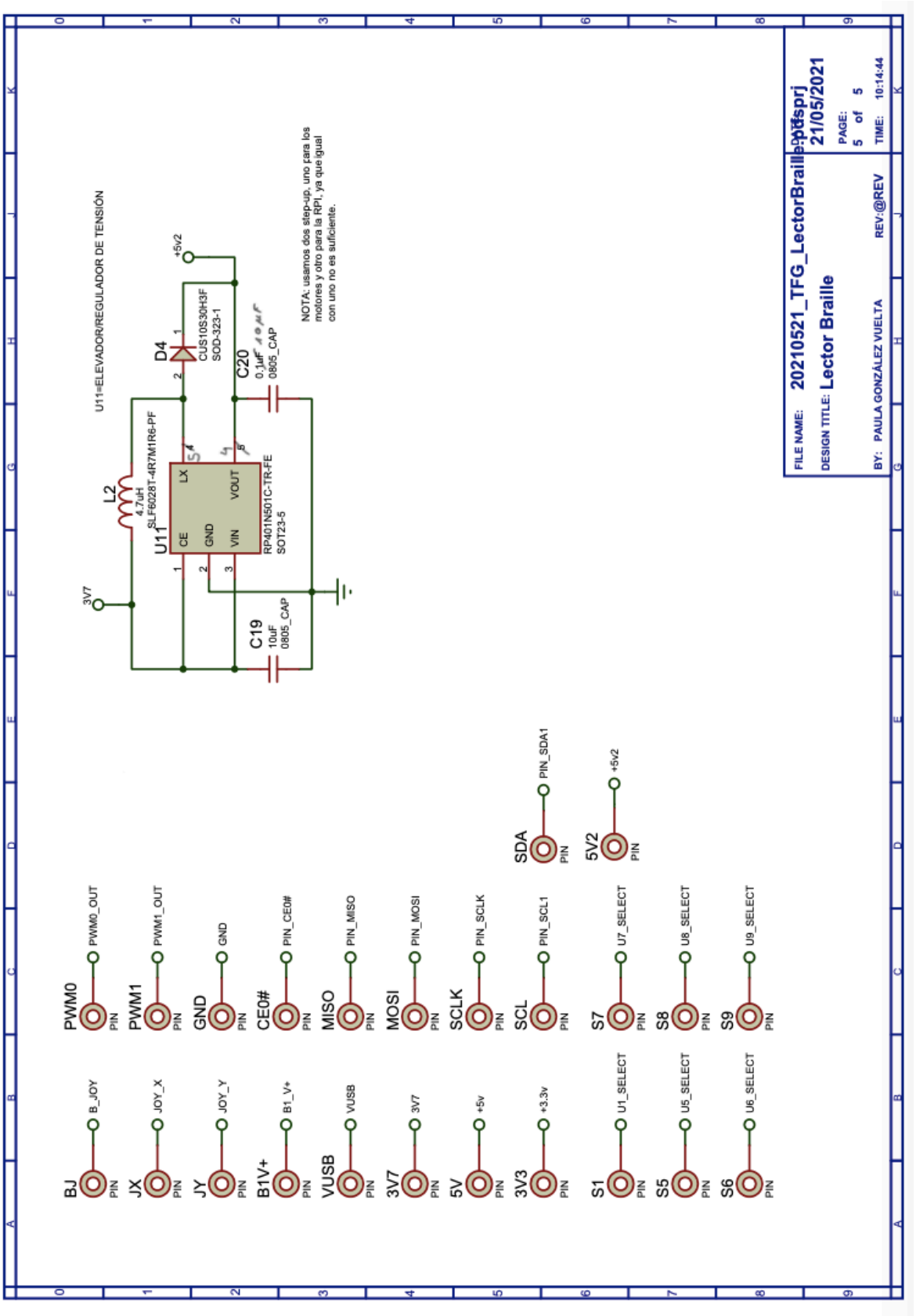


Figura 4.5: *Hoja 5 del esquemático, regulador extra y puntos de test*

4.1.1 Conexionado de la Raspberry pi Zero W

Se aprovecharán las distintas funciones que ofrecen los pines de la raspberry pi, *figura 4.6*, para conectarlos al resto del circuito. Algunos de los pines pueden funcionar como pines GPIO y además permitir la comunicación SPI o I2C. Por ello, hay que reservar estos pines para usarlos en estos casos especiales, necesarios en la aplicación [25].

Sabiendo esto, como se puede ver en la *figura 4.1* los pines que han sido utilizados son los descritos a continuación:

- Pines de alimentación: La *Raspberry Pi Zero W* cuenta con varios pines de los que podemos obtener tensión de 3.3V. Para funcionar, necesita una tensión de 5V que se obtiene a la salida del elevador de tensión y puede proporcionar una tensión de 3.3V adicional a través del regulador interno que tiene incorporado, y que se usará como tensión de referencia para aquellos componentes que no requieran de una tensión de 5V.
- Pines GND: Para conseguir la tierra del circuito.
- Pines GPIO: Pines de entrada y salida de propósito general, que irán conectados a los puentes H. Se usarán el GPIO17, GPIO27, GPIO23, GPIO24, GPIO25 y GPIO12.
- Pines SPI: Son los pines MISO, MOSI, SCLK, CE0 y CE1, estos dos últimos activos en nivel bajo. Permiten la comunicación a través del bus serie, con una topología maestro-esclavo. Se usarán para la comunicación del joystick con la raspberry a través de CAD. Gracias al pin SCLK se puede obtener una señal de reloj para los procesos síncronos.
- Pines I2C: Son los pines SDA, como línea de datos, y SCL, para el reloj. Compatibles con el protocolo I2C en serie. Se usarán para programar la celda Braille, permitirán a las raspberry mandar las órdenes necesarias para subir y bajar los motores, asignando una dirección distinta a cada uno de ellos.

Un caso especial es el uso de los pines GPIO13 para PWM0 y GPIO18 para PWM, para obtener las señales de audio.

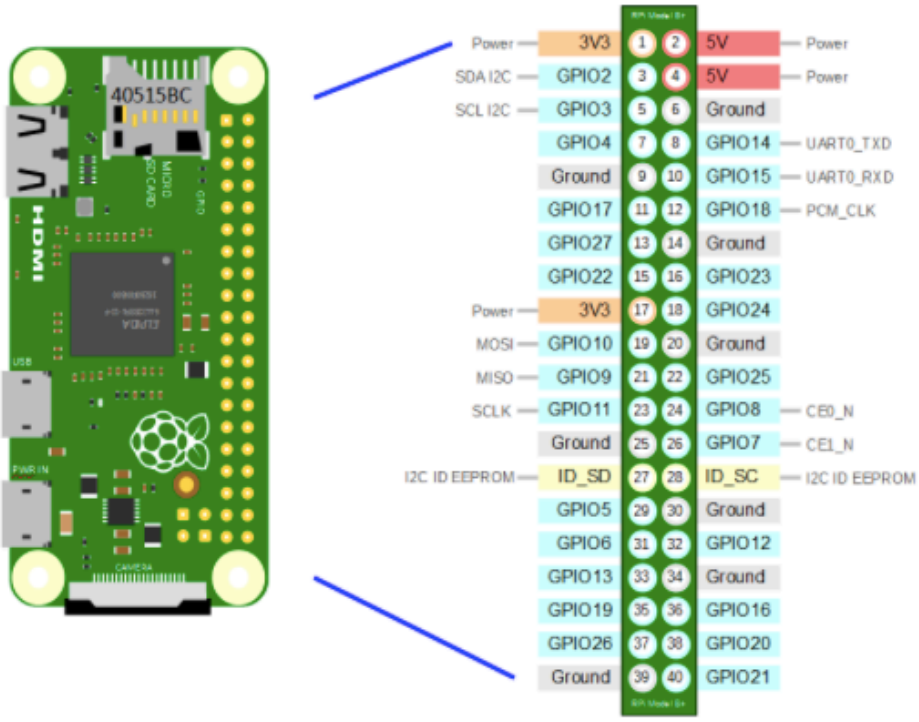


Figura 4.6: *Pines disponibles en la Raspberry pi 0 W*

4.1.2 Conexionado del Joystick y el CAD

Arriba a la izquierda de la *figura 4.1* se pueden ver las conexiones del módulo del joystick, formado por el propio joystick más un convertidor analógico-digital, cuyas salidas se conectan a la raspberry pi.

Ambos componentes se alimentarán con 3.3V, ya que si nos fijamos en la hoja de datos del MCP3004 se puede ver que admite un rango de tensiones entre 2.7V y 5V. Por lo tanto, usamos la tensión de alimentación lo más pequeña posible para tener un menor consumo eléctrico y de potencia.

Las salidas del joystick que indican la posición en cada uno de los ejes se conectan a los tres primeros canales del CAD.

Mediante el bus SPI se comunican la raspberry pi y el CAD, y la función de los pines es la siguiente (*figura 4.7*):

- Pin CS (*Chip Select*): Se usa para iniciar la comunicación y realizar la conversión analógico-digital cuando está en baja. Entre conversiones es necesario ponerlo en alta. Se conecta con el pin CE0 de la raspberry pi.
- Pin DIN (*Serial Data Input*): Es el pin de entrada de datos del bus SPI y se utiliza para cargar los datos de configuración del canal en el dispositivo. Se conecta con el pin MOSI de la raspberry pi.
- Pin DOUT (*Serial Data Output*): Es el pin de salida de datos del bus SPI, que se encarga de enviar los resultados de la conversión analógico-digital a la raspberry pi, conectado con el pin MISO.

MCP3004	MCP3008	Symbol	Description
PDIP, SOIC, TSSOP	PDIP, SOIC		
1	1	CH0	Analog Input
2	2	CH1	Analog Input
3	3	CH2	Analog Input
4	4	CH3	Analog Input
–	5	CH4	Analog Input
–	6	CH5	Analog Input
–	7	CH6	Analog Input
–	8	CH7	Analog Input
7	9	DGND	Digital Ground
8	10	$\overline{CS}$/SHDN	Chip Select/Shutdown Input
9	11	D_{IN}	Serial Data In
10	12	D_{OUT}	Serial Data Out
11	13	CLK	Serial Clock
12	14	AGND	Analog Ground
13	15	V_{REF}	Reference Voltage Input
14	16	V_{DD}	+2.7V to 5.5V Power Supply
5,6	–	NC	No Connection

Figura 4.7: *Funcionalidad de los pines del MCP3004*

4.1.3 Conexionado de la Alimentación

En las *figuras 4.1* y *4.5* encontramos el bloque de la alimentación, del que se obtienen los 5V. El proceso para obtener la tensión de alimentación de referencia es el siguiente: Mediante el micro usb tipo B nos conectamos a la corriente para cargar la batería a través del cargador (U10). A continuación, la tensión de la batería de 3.7V se eleva a 5V a través de los elevadores de tensión U2 y U11, para alimentar los puentes H y la raspberry pi, respectivamente.

4.1.4 Conexionado de los motores y los puentes H

En las *figuras 4.2 y 4.3* se han colocado las conexiones entre los puentes H y los motores. Este bloque estará alimentado con una tensión de 5V Como ya se comentó, tenemos un componente que actúa como puente H doble por cada uno de los motores.

El conexionado para todos ellos es equivalente (*figura 4.8*):

- las salidas del puente H OUT1, OUT2, OUT3 y OUT4 se conectan a los devanados del motor.
- Los pines SCL y SDA de cada puente H tienen una conexión común a los pines del mismo nombre en la raspberry pi.
- El pin nFAULT se conecta a la raspberry pi para habilitar o no la comunicación entre esta y el puente H que corresponda. Tendremos una conexión distinta por cada puente H.
- Los pines IN1, IN2, IN3 e IN4 de entrada se pueden dejar flotantes ya que cuentan con un *pull-down* interno, y servirán para controlar el avance y retroceso de los motores.

PIN				TYPE(1)	DESCRIPTION
NAME	DRV8847		DRV8847S		
	TSSOP HTSSOP	WQFN	TSSOP		
GND	13	11	13	PWR	Device ground. Recommended to connect the GND pin and device thermal pad (HTSSOP and WQFN packages) to ground
IN1	16	14	16	I	Half-bridge input 1
IN2	15	13	15	I	Half-bridge input 2
IN3	9	7	9	I	Half-bridge input 3
IN4	10	8	10	I	Half-bridge input 4
ISEN12	3	1	3	O	Full-bridge-12 sense. Connect this pin to the current sense resistor for full-bridge-12. Connect this pin to the GND pin if current regulation is not required.
ISEN34	6	4	6	O	Full-bridge-34 sense. Connect this pin to the to current sense resistor for full-bridge-34. Connect this pin to the GND pin if current regulation is not required.
MODE	14	12	—	I	Tri-state pin for selection of driver operating mode
nFAULT	8	6	8	OD / I	Fault indication pin. This pin is pulled logic low with a fault condition. This open-drain output requires an external pullup resistor. This pin is also used as an input pin for the DRV8847S device for releasing the I^2C bus.
nSLEEP	1	15	1	I	Sleep mode input. Set this pin to logic high to enable the device. Set this pin to logic low to go to low-power sleep mode
OUT1	2	16	2	O	Half-bridge output 1
OUT2	4	2	4	O	Half-bridge output 2
OUT3	7	5	7	O	Half-bridge output 3
OUT4	5	3	5	O	Half-bridge output 4
SCL	—	—	11	I	I^2C clock signal.
SDA	—	—	14	OD	I^2C data signal. The SDA pin requires a pullup resistor.
TRQ	11	9	—	I	Torque current scalar
VM	12	10	12	PWR	Power supply. Connect the VM pin to the motor power supply. Bypass this pin to ground with a VM-rated 0.1-µF (ceramic) and 10-µF (minimum) capacitor.

Figura 4.8: *Funcionalidad de los pines del DRV8847s*

Cada chip se comunicará con la raspberry pi mediante el protocolo I2C, para ello cuenta con una serie de registros de 8 bits, nosotros usaremos tres de ellos. La función de dichos registros es la que se describe a continuación [14]:

- Registro *Slave Address*: (*Figura 4.9*) Se accede al registro mediante la dirección 0x00. Como su propio nombre indica, sirve para fijar la dirección del dispositivo esclavo en el bus I2C, la cual es por defecto 0x60. Como tenemos 6 dispositivos esclavos (drivers), será necesario acceder a este registro para cambiar su valor por defecto y poder controlar con la raspberry pi los seis chips por separado. Más adelante se explicará el software necesario de configuración para realizar dicha tarea.

Bit	Field	Type	Reset	Description
7	RSVD	R	0b	Reserved
6-0	SLAVE_ADDR	R/W	1100000b	Slave address (8 bit) The default value is 0x60

Figura 4.9: *Descripción de los campos del registro Slave Address*

- Registro de control IC1: (*figura 4.10*) Se accede a este registro mediante la dirección 0x01 y por defecto todos sus bits están a 0. Mediante el software se jugará con los valores de los bits 3-6, que harán girar los motores en un sentido u otro, por lo que el bit 2 se configurará para usar los bits INx, en lugar de los pines físicos que recordamos que se han dejado flotantes, y los bits 0 y 1 se configurarán para usar el modo *4-pin interface*.

Bit	Field	Type	Reset	Description
7	TRQ	R/W	0b	0b = Torque scalar set to 100% 1b = Torque scalar set to 50%
6	IN4	R/W	0b	The INx bits are used to control the bridge operation.
5	IN3	R/W	0b	The INx bits are used to control the bridge operation.
4	IN2	R/W	0b	The INx bits are used to control the bridge operation.
3	IN1	R/W	0b	The INx bits are used to control the bridge operation.
2	I2CBC	R/W	0b	0b = Bridge control configured by using the INx pins 1b = Bridge control configured by using the INx bits
1-0	MODE	R/W	00b	00b = 4-pin interface 01b = 2-pin interface 10b = Parallel interface 11b = Independent mode

Figura 4.10: *Descripción de los campos del registro de control IC1*

- Registro de control IC2 (*figura 4.11*): Se accede a él mediante la dirección 0x02. El bit que nos servirá de utilidad será el bit 6, DISFLT, que sirve para habilitar o deshabilitar el pin nFAULT del chip. Será importante para configurar las direcciones de cada dispositivo esclavo.

Bit	Field	Type	Reset	Description
7	CLRFLT	R/W	0b	Set this bit to issue a clear FAULT command. This command clears all FAULT bits other than the OLD and OLDx bits. This bit reset to 0b after clearing all the faults. 0b = No clear FAULT command issued 1b = Clear FAULT command issued
6	DISFLT	R/W	0b	0b = nFAULT pin not disable 1b = nFAULT pin is disabled
5	RSVD	R	0b	Reserved
4	DECAY	R/W	0b	0b = 25% fast decay 1b = 100% slow decay
3	OCPR	R/W	0b	0b = OCP auto retry mode 1b = OCP latch mode
2	OLDOD	R/W	0b	0b = Idle 1b = OLD on-demand is activated
1	OLDFD	R/W	0b	0b = Fault signaling on OLD 1b = No fault signaling on OLD
0	OLDBO	R/W	0b	0b = Bridge operating on OLD 1b = Bridge Hi-Z on OLD

Figura 4.11: *Descripción de los campos del registro de control IC2*

El DRV8847S tiene distintos modos de funcionamiento para controlar motores, nosotros usaremos el modo *4-pin interface full-step*, como ya se ha mencionado, configurado para conducir un motor paso a paso o dos motores DC. Para configurar este modo de operación habrá que conectar a tierra el pin MODE y usar los pines IN1, IN2, IN3 e IN4 para controlar los puentes H. Esto permitirá el movimiento del motor en sentido de avance, para subir, y en retroceso, para bajar, con un cambio de fase de 90º entre devanados [6]. La secuencia de funcionamiento la podemos ver en la *figura 4.12*, consiste en ir activando los pines INx de dos en dos, para activar dos bobinas y que el motor avance un paso completo.

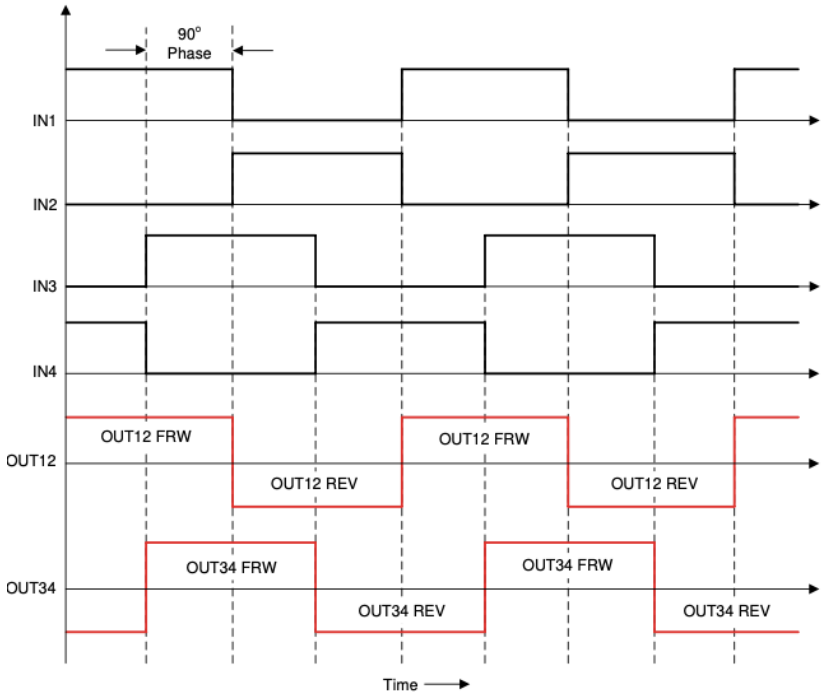


Figura 4.12: *Secuencia del modo 4-pin interface full-step*

4.1.5 Conexionado del Audio

Por último, en la *figura 4.4* se muestra el circuito necesario para conseguir audio. Las salidas PWM0_OUT y PWM1_OUT son los canales izquierdo y derecho, respectivamente.

Las resistencias R36 y R38 del canal izquierdo y R37 yR39 del derecho, actúan como un divisor de tensión que reduce la tensión a 1.1V aproximadamente.

A su vez, R36 y R37 forman un filtro paso bajo junto los condensadores C21 y C22, para filtrar las altas frecuencias presentes en la señal PWM de salida de la raspberry pi. A continuación, podemos encontrar dos condensadores de acoplo, C23 y C24, que evitan el paso de la corriente continua hacia el amplificador. A mayores, se colocan diodos de protección frente a descargas electroestáticas.

Una vez se han filtrado las señales de audio, se mezclan ambos canales para introducir una única señal de entrada en el amplificador, cuyas salidas se conectarán a los altavoces y al conector Jack 3.5mm.

El amplificador es estéreo, de 750mW de potencia, entrada analógica y de clase AB con control de volumen DC y compatible con salida de audio por altavoz y por auriculares.

4.2 Diseño de la carcasa 3D

Como ya se comentó anteriormente, para el diseño de la carcasa 3D se ha utilizado el programa Tinkercad. Se han diseñado la parte superior e inferior de la carcasa, como podemos ver en la *figura 4.13* teniendo en cuenta las conexiones exteriores que se necesitan en el diseño.

Para el diseño de la carcasa se plantearon dos objetivos principales:

- Conseguir un diseño ergonómico.
- Colocar las conexiones exteriores de forma estratégica para que el manejo del dispositivo fuera lo más fácil posible.

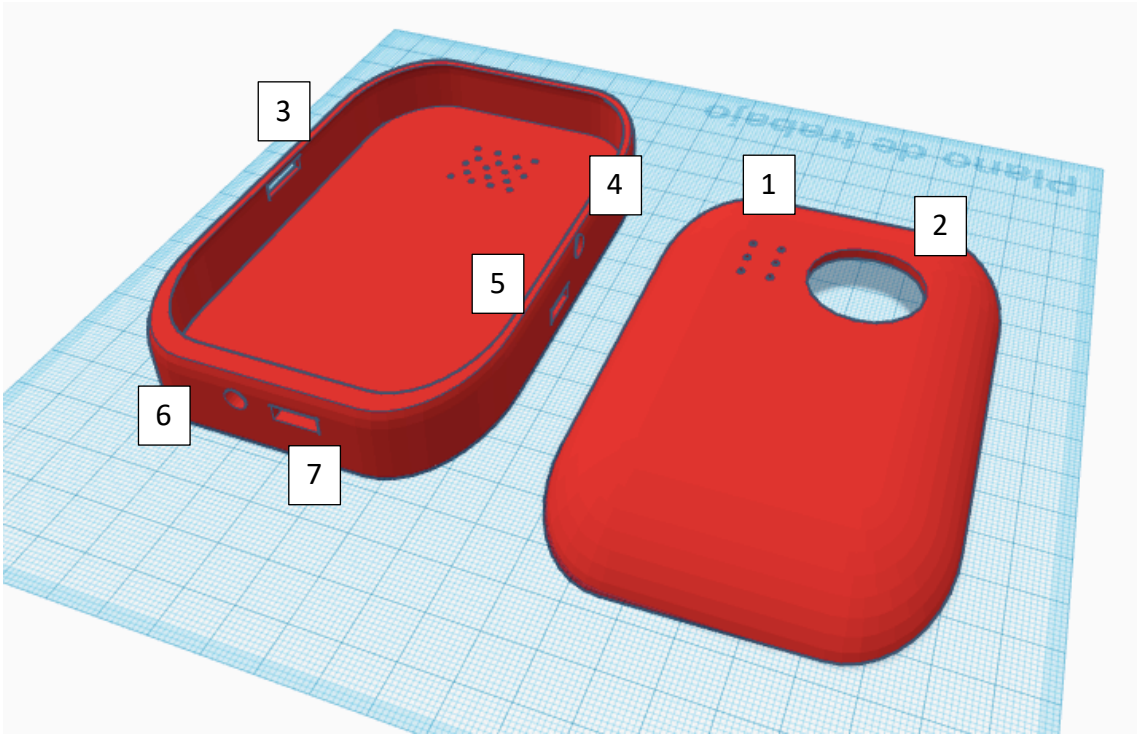


Figura 4.13: *Parte inferior y superior de la carcasa, de izquierda a derecha*

Para conseguir un diseño ergonómico se intentó dimensionar la carcasa para que el tamaño y la forma se asemejaran a las de un ratón, con la parte superior un poco redondeada para que fuera más cómodo sostener el dispositivo con una sola mano. Las dimensiones se minimizaron lo máximo posible teniendo en cuenta las limitaciones físicas que marcaban los componentes.

En cuanto a la colocación de los componentes exteriores se siguieron las reglas siguientes, según la numeración que se muestra en la *figura 4.13*:

1) La celda Braille se situó en la esquina superior izquierda para que el usuario siguiera la lectura con el dedo índice.
2) El joystick, como debido a su tamaño era difícil situarlo en un lateral, se colocó en la otra esquina superior, al considerarse el lugar donde no entorpecería la sujeción del dispositivo con una mano.
3) El interruptor se situó en el lado izquierdo de forma que se pudiera controlar con el dedo pulgar.

4) En el lateral derecho se situó el potenciómetro para controlar el volumen.
5) En el lateral derecho se situó la conexión micro USB B para insertar la memoria con los libros.
6) En el lateral inferior se situó la conexión Jack.
7) El micro USB B para la carga del dispositivo también está situado en el lateral inferior.

4.3 Diseño de la PCB

Como novedad, en este nuevo diseño, el hardware del dispositivo irá integrado sobre una placa de circuito impreso. Una placa de circuito impreso, o PCB, sirve de soporte e interconexión para los componentes electrónicos que forman el sistema. Este paso se realiza una vez se ha completado el diseño del esquemático y se ha verificado que todo esté conectado de forma correcta. Como ya se ha comentado, seguiremos utilizando el programa de *Proteus 8.12* para la realización de estos apartados.

La principal incógnita que surge es como colocar los componentes sobre la PCB para conseguir un diseño ergonómico y lo más pequeño posible, que se adapte a la carcasa previamente diseñada, algo que, como ya se dijo, es clave en la toma de decisiones.

Sin embargo, antes de decidir la colocación de los componentes es importante comprobar que cada uno de ellos tiene una huella asociada. En caso afirmativo, comprobaremos que esa huella coincide con la que aparece en la hoja de especificaciones; y en caso de que no la tenga, buscaremos en las librerías si tenemos alguna que se ajuste a las medidas requeridas, y sino, se creará de cero.

La huella representa donde se colocan los componentes, y está formada por los pads y la serigrafía. Los pads pueden ser de tipo montaje superficial (smd) o de agujero pasante, en función del encapsulado del componente. La serigrafía es informativa y será útil para marcar las dimensiones que tiene el componente y saber

el espacio que va a ocupar en la PCB. En nuestro caso, además, se ha utilizado la serigrafía para marcar el pin 1 de cada componente como referencia, lo que agiliza el proceso de montaje y evita errores a la hora de colocarlos al revés. Es importante prestar especial atención a este punto, ya que errores derivados de una huella mal diseñada pueden hacer que se tenga que volver a fabricar la PCB como, por ejemplo, en el caso de no enumerar los pads de forma correcta o que esté mal dimensionada.

En este caso, las huellas que se han tenido que diseñar de cero son la del joystick, el micro USB tipo B, el interruptor, el amplificador de audio, el potenciómetro, el conector de audio, las bobina y el led rojo, componentes que son algunos de agujero pasante y otros de montaje superficial, y otros tienen ambos tipos de pads.

Para componentes mecánicos, como los motores, se ha usado la serigrafía para dibujar la planta y saber el espacio necesario para colocar los 6 motores que representan la celda Braille, y que se tendrá en cuenta para colocar el resto de los componentes.

Una vez que se han diseñado todas las huellas y la serigrafía necesaria, es cuando se comienza a colocar los componentes sobre la PCB y trazar las pistas que interconecten los componentes entre sí. Para ello, se deben tener en cuenta una serie de reglas de diseño relacionadas con la compatibilidad electromagnética. Entre ellas:

- Los condensadores de acoplo se colocarán lo más cerca posible de la alimentación, es decir, se colocarán lo más pegados posibles al componente.
- A la hora de trazar las pistas que unen los pads, cuanto más cortas sean mejor y entre pistas deberá haber una distancia mínima para prevenir acoplamientos capacitivos o por impedancia común.
- Si las pistas ocupan un área grande se pueden producir bucles de corriente, lo que generará acoplamiento magnético.
- Además, todas aquellas pistas relacionadas con la alimentación se trazarán más anchas, para disminuir la resistencia de la pista.

- También, se intentará rutar el mayor número de pistas por la cara superior.

El tamaño máximo que puede tener la PCB será de 10x10cm, ya que es la dimensión estándar que nos proporciona el proveedor que utilizaremos para fabricarla y la más barata. De todos modos, en nuestro caso, el tamaño final lo delimita la carcasa diseñada y será menor. También se permite el trazo de pistas por 2 caras.

Con las huellas diseñadas y las reglas de diseño establecidas, los primeros componentes que se colocarán son el joystick y los motores, uno al lado del otro, porque nos determinarán la anchura de la PCB al ir colocado uno al lado del otro y ser los componentes que más volumen ocupan.

El siguiente paso es colocar todos los componentes que tienen conexión al exterior en el borde de la placa. Estos componentes son: la raspberry pi, el potenciómetro, el micro USB tipo B, el conector Jack y el interruptor.

A continuación, se colocan el resto de los componentes, procurando agruparlos por bloques para que los que estén interconectados entre sí estén lo más cerca posible, y así respetar las reglas de diseño. Los puentes H y sus componentes externos se han colocado debajo de los motores, los reguladores y la batería entre el interruptor y el micro USB tipo B y el bloque del audio en el espacio que quedaba disponible al ser independiente del resto de módulos y solo tener conexión con las raspberry pi. A mayores, se han añadido bastantes puntos de test para medir las señales más relevantes y poder verificar el funcionamiento de forma más eficiente, al lado de los pines correspondientes a cada uno.

A la hora de realizar el rutado de las pistas el programa Proteus cuenta con una herramienta de autorrutado que puede simplificar mucho el trabajo. Sin embargo, hay que tener cuidado si se decide utilizar en lugar de optar por un rutado manual, ya que las pistas que traza no son demasiado óptimas y correctas. Esto se debe a que no sigue ningún patrón y traza las pistas de forma muy aleatoria. Aun así, debido a la dificultad que se presentaba al tener los componentes tan juntos entre sí, resultaba

muy complicado el rutado totalmente manual. Por tanto, se optó por realizar un primer autorutado y a partir de este, revisar y modificar aquellas pistas que fueran problemáticas manualmente, además de aumentar la anchura de las pistas de la alimentación, como ya se comentó antes. En las *figuras 4.14 y 4.15* se recogen los resultados de las dos caras de la PCB diseñada.

Finalmente, se buscó algún hueco para añadir la serigrafía con el nombre del alumno que realiza este trabajo de fin de grado, el escudo de la ETSIT, el escudo de la Universidad de Valladolid y unas siglas "LLEB" que representan el nombre del producto: Lector de Libros electrónicos en Braille.

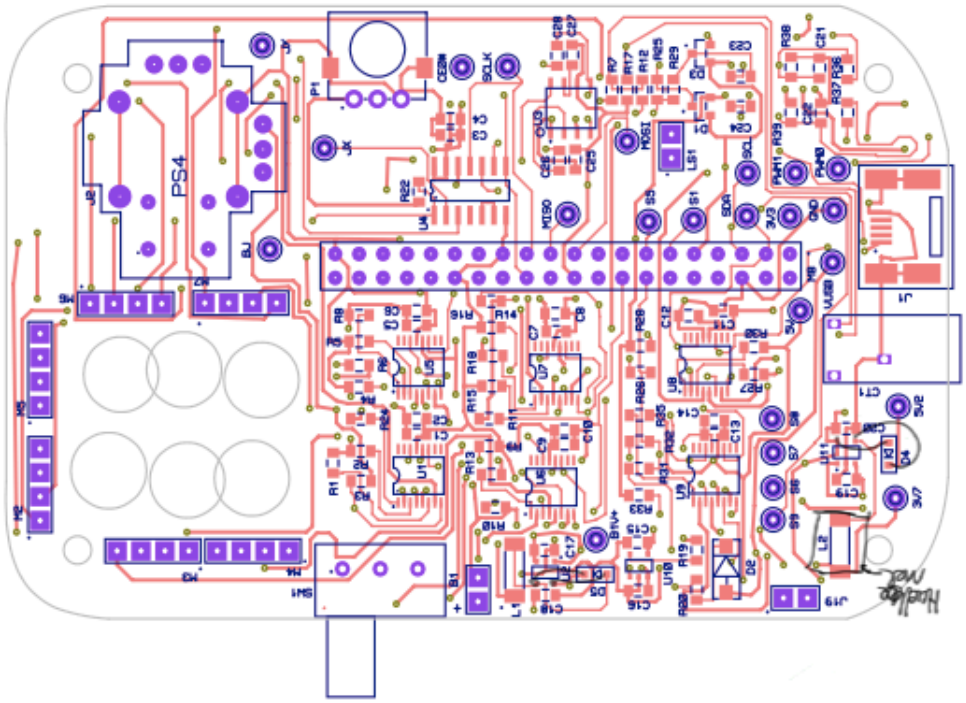


Figura 4.14: Cara superior de la PCB

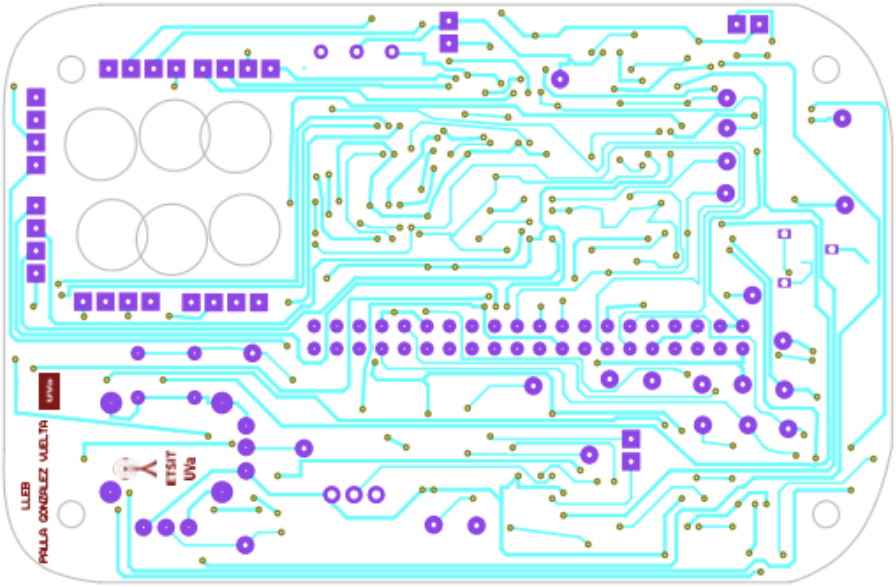


Figura 4.15: *Cara inferior de la PCB*

4.4 Fabricación de la PCB

El proceso de fabricación de la placa se ha llevado a cabo a través de la página *PCBway* [26]. Esta empresa de fabricación se encuentra en la ciudad de Shenzhen, en China, y nos ofrece una opción económica, con una calidad adecuada y en muy poco tiempo. El proceso para pedir las PCB es muy sencillo:

- Lo primero de todo es generar desde el Proteus los ficheros gerber que será necesario adjuntar en la solicitud.
- A continuación, se selecciona en la web las características que queremos que tenga nuestra PCB (grosor de la placa, color, material, etc) así como la cantidad y el tamaño. En nuestro caso, se ha escogido una placa de 100x100mm, 5 unidades y de color azul. El color por defecto es verde, pero seleccionar ese color no aumentaba el precio y se decidió ese porque la serigrafía destaca más.

- Por último, una vez se realiza el pago, el equipo de PCBway revisa los ficheros y en caso de que haya algún error se ponen en contacto para solucionarlo.

Todo este proceso es muy rápido y, a pesar de la lejanía, en tan solo 1 semana están las placas en nuestras manos y no supone ralentizar mucho el tiempo de desarrollo del proyecto.

4.5 Montaje de componentes

El montaje de la placa se realizó en el laboratorio de la ETSIT, en la Universidad de Valladolid, donde se cuenta con numerosas herramientas necesarias para el montaje y análisis de PCBs. Entre las necesarias para nuestro montaje se encuentra un soldador con elevada temperatura de fusión, para fijar de forma sólida con estaño los componentes a la PCB, microscopio, pinzas de precisión o rotuladores flux.

El montaje de los componentes es la etapa más decisiva, ya que es en este punto cuando podremos verificar el funcionamiento del dispositivo y detectar posibles errores derivados del diseño. Por ello, se definen una serie de pasos para soldar bloque por bloque e ir comprobando poco a poco la funcionalidad. Nos apoyaremos en la lista de materiales, el esquemático y el layout para llevar un cierto orden y organización.

Para evitar errores en el montaje hay que prestar especial atención a la orientación de los componentes y asegurarnos de que cada pin se conecta a su pad. Todos los encapsulados suelen tener una marca que indica donde está situado el pin 1, ya que este se usa como pin de referencia, al igual que hemos hecho en la PCB. También hay que asegurarse de que el pin queda bien soldado en su lugar para evitar problemas de conexión por poco contacto con el pad, o en otro caso, que al soldar haya algún cortocircuito al unir dos pads o pistas. Por eso, al trabajar con componentes tan pequeños, resulta tan útil el uso de un microscopio, para detectar estos fallos a tiempo

y corregirlos antes de alimentar el dispositivo, porque se pueden estropear los componentes y tocaría sustituirlos por otros nuevos.

Teniendo todo esto en cuenta, se procedió al montaje de componentes, *figura 4.16*, colocando en primer lugar todo el bloque de la alimentación, verificando que se obtenían las tensiones correctas en cada punto del circuito. A continuación, se colocaron los puentes H, el CAD y el amplificador, es decir, el resto de encapsulados SMD. Después se colocaron los componentes pasivos antes de colocar la raspberry pi, el joystick, el potenciómetro, el interruptor, etc, componentes que ocupan un mayor volumen.

Finalmente, con todo el hardware verificado, se realizó el montaje de los motores para su caracterización. Algunos de ellos, debido a su pequeño tamaño, resultaron muy difíciles de montar con las herramientas disponibles y será algo a tener en cuenta a la hora de tomar una decisión acerca del motor más adecuado.

A mayores, como se puede ver en la *figura 4.17,* la raspberry pi se ha colocado un nivel por debajo de la PCB, ya que era la única forma de adaptarse a las dimensiones de la carcasa y evitar hacer un dispositivo muy ancho. Para unirlo con la PCB se ha colocado un conector hembra de 40 pines en la raspberry pi y un conector macho en la raspberry pi.

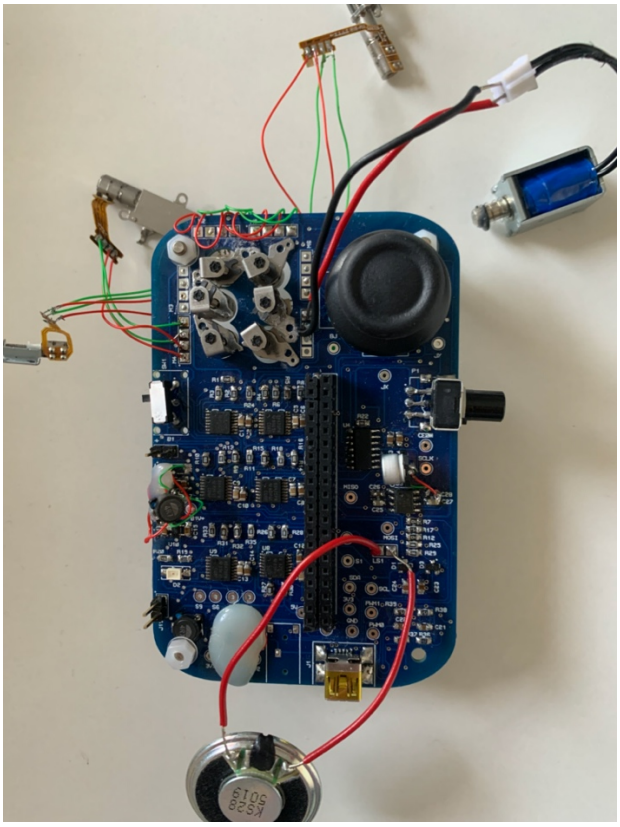


Figura 4.16: *Vista superior de la PCB*

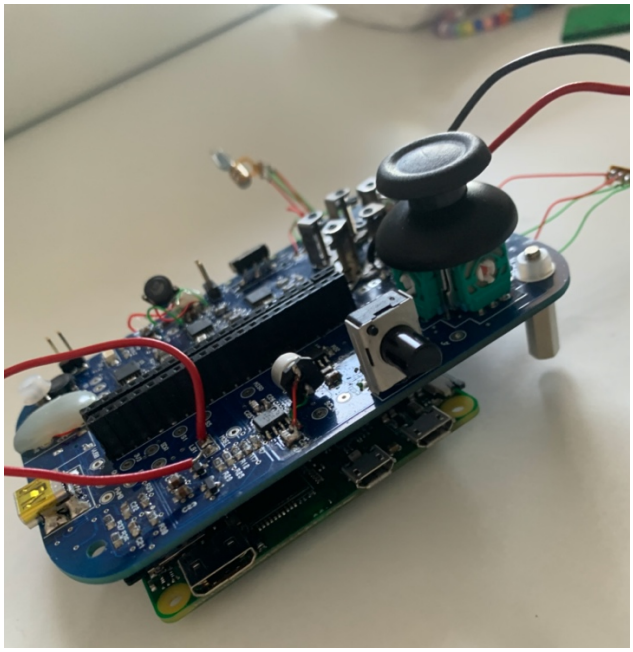


Figura 4.17: Vista lateral de la PCB

4.6 Caracterización de los motores

En este apartado, se va a llevar a cabo una caracterización de distintas alternativas que pueden resultar válidas para emular la celda Braille. En concreto, se analizarán 4 motores paso a paso y el propio electroimán, usado en la versión anterior. Lo que se pretende conseguir es un motor que sea capaz de representar el mayor número de caracteres posibles en el menor tiempo posible. En primer lugar, analizaremos los motores en cuanto al consumo eléctrico que tienen cuando están en movimiento y en cuanto a la facilidad de montaje sobre la PCB. A continuación, se descartarán los modelos que no cumplan nuestros requisitos y finalmente, en caso de que algún motor sea válido, se estudiará la cantidad de caracteres por segundo que es capaz de representar. En la *figura 4.18* tenemos los 4 modelos que vamos a caracterizar y las dimensiones de cada uno de ellos, excepto las del motor 2, que tiene unas dimensiones de 4,5mm de alto y 3.3mm de diámetro.

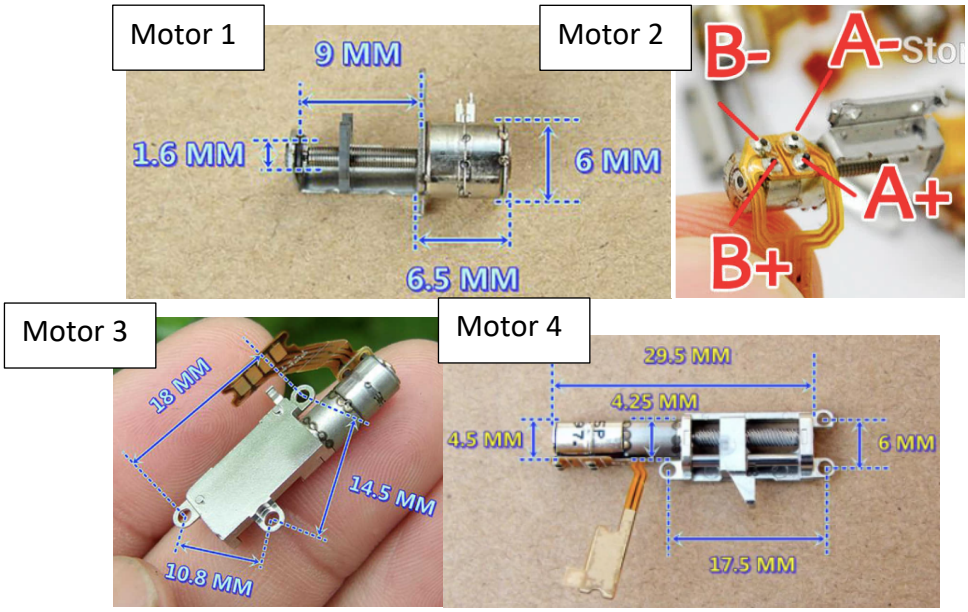


Figura 4.18: *Distintos modelos de motores paso a paso*

Para medir el consumo necesitaremos medir la tensión en los pines ISEN12 o ISEN34 del DRV8847s, y así podremos obtener la corriente requerida por cada uno de ellos aplicando la ley de Ohm: V/R = I. En la *tabla 4.1* se recogen los resultados obtenidos.

Para medir la tensión hay que tener en cuenta que en el caso de los motores paso a paso la resistencia conectada en los dos pines mencionados es la misma: 0,33Ω. Por lo tanto, ISEN12 = ISEN34 y bastará con medir la tensión en uno de los puntos. Por el contrario, para el electroimán solo tiene un devanado y mediremos la tensión en el pin ISEN12.

Para tomar una decisión sobre qué motor podría ser el más adecuado, no solo se tendrá en cuenta el consumo eléctrico de cada motor, aunque sí que es cierto que este factor tendrá un peso importante. Como ya se ha comentado, será necesario ver la dificultad que supone el montaje del motor sobre la placa, que como ahora veremos, nos va a condicionar en la elección del más adecuado.

Se caracterizará dicha dificultad con 3 niveles, teniendo en cuenta que hay que montar seis motores:

- Dificultad Alta: Si el montaje de los 6 motores con los medios disponibles en el laboratorio se volvería muy complicado debido al tamaño del motor.
- Dificultad Media: Si a pesar del pequeño tamaño de los motores, el modelo es posible soldarlo a la PCB si se tiene algo de precisión.
- Dificultad baja: Si no supone apenas dificultad la integración de los motores en la PCB.

	Rdev (Ω)	**ISEN12 (mV)**	**Idev (mA)**	**Dificultad montaje**
Motor paso a paso 1 (M2)	52	28	82	Alta
Motor paso a paso 2 (M3)	25	42	128	Media
Motor paso a paso 3 (M4)	14	60	182	Media
Motor paso a paso 4 (M6)	15	83	252	Media
Electroimán (M7)		81	246	Baja

Tabla 4.1: *Características de los motores*

Comentando los resultados, se puede ver que cuanto mayor es la resistencia por devanado del motor, menor es la tensión en el punto medido y, por consiguiente, menor es la corriente consumida por el motor. Si solo tuviéramos en cuenta las características eléctricas escogeríamos, sin duda, el primer motor (M2), que era la

primera opción desde el principio. Sin embargo, cuando se llevó a cabo el montaje sobre la PCB se encontró una dificultad muy elevada a la hora de soldar los cables en el motor, llegando incluso a fundir las conexiones en alguno de los casos. Por lo tanto, M2 se descartará. En cuanto a los motores M3, M4 y M6, la dificultad de montaje es media en los 3 casos, ya que todos ellos cuentan con unas pistas a las cuales se pueden soldar los cables con más facilidad y podrían montarse en la PCB. El motor M6 tiene un consumo similar al del electroimán (M7), por lo que también se descartará. Como era de esperar, los electroimanes se descartarán (M7), ya que a parte de que su consumo es el más elevado, el tamaño es demasiado grande y el objetivo que se persigue en este trabajo es sustituirlos. Por lo tanto, de todos los motores que hemos analizado solamente podrían valer el M3 y el M4. A priori, el motor M3 sería la opción más idónea: es el que menos consume y es el más pequeño. Sin embargo, debido a ser el de menores dimensiones, es probable que no soporte la presión del dedo y no se pueda simular la subida y bajada de los puntos braille, un factor importante que en un diseño posterior habría que analizar.

Finalmente, se realiza un análisis experimental del movimiento de los motores M3 y M4. El motor M3 no fue capaz de soportar las pruebas a una velocidad tan elevada, y dejó de funcionar porque se sobrecalentó demasiado y por este motivo no se pudieron realizar las pruebas y quedará pendiente de análisis, porque debido a la falta de tiempo no ha sido posible cambiarlo por uno nuevo. Por el contrario, el motor M4 si que soportó las pruebas realizadas, y es capaz de representar unos 8 caracteres por segundo desplazándose 1mm arriba y abajo. Por lo tanto, este motor tiene unas prestaciones buenas en cuanto a su desplazamiento y es el único que será válido para un futuro diseño.

4.7 Análisis del consumo eléctrico

A la hora de diseñar un dispositivo, es importante saber el consumo eléctrico que vamos a tener. Para realizar este cálculo hay que fijarse en las hojas de datos de cada componente, para obtener la corriente máxima, así como en las tensiones de referencia utilizadas.

3,3V		5V	
CAD	0,55mA	RPI 0W	230mA
Amplificador	7mA	Cargador	1,5mA
Potenciómetro	0.33mA	Motores (x6)	182mA/motor
		Regulador (x2)	0,19mA/regulador
TOTAL (3,3V)	7,88mA	TOTAL (5V)	1,32A

Tabla 4.2: *Estimación del consumo eléctrico del dispositivo con el motor M4*

Atendiendo a los resultados de la *tabla 4.2*, y teniendo en cuenta que cada regulador tiene una corriente máxima de salida de 500mA, vemos que de cualquier forma se supera la corriente total que se puede conseguir con ambos reguladores si se escoge el motor M4, como se había adelantado en el apartado anterior.

En realidad, con el montaje actual del dispositivo solo sería compatible el motor M2, porque un solo regulador se encarga de alimentar a todos los motores, es decir, la corriente de los 6 motores no puede superar los 500mA, y para eso cada motor debería consumir menos de 83mA.

A continuación, se plantean alternativas que podrían llevarse a cabo para aumentar las opciones a la hora de elegir un motor:

- La opción más asequible en estos momentos sería mover los motores de forma secuencial, ya que de este modo el hardware actual sería compatible con cualquiera de los motores y no necesitaríamos modificar el diseño, simplemente adaptar el software a esta necesidad.
- No alimentar a todos los motores con el mismo regulador: Como el diseño cuenta con 2 reguladores y la raspberry pi consume una corriente de 230mA, podríamos alimentar 1 o 2 motores con el otro regulador. De este modo, podríamos escoger un motor de hasta 120mA.
- Además de la solución anterior, sustituir un regulador por otro que proporcione una corriente de 1A: En total tendríamos una corriente de 1,5A para alimentar a la raspberry pi y a los motores, y podríamos escoger

un motor de hasta 200mA. El problema es que habría que rediseñar la PCB y escoger ese nuevo regulador. Esta sería la solución más adecuada ya que nos abre un rango de posibilidad mayor a la hora de escoger un motor.

4.8 Corrección de errores

A la hora de realizar el montaje de los componentes se detectaron algunos errores de diseño que, por suerte, ha sido posible corregir sin necesidad de volver a fabricar una nueva PCB. Esos errores son los siguientes:

- El primer error que se detectó fue en el elevador de tensión: A la hora de crear el componente para situarlo sobre el esquemático los pines 4 y 5 se habían intercambiado. Como los dos pines estaban del mismo lado, se pudo solucionar levantando esas patillas para que no hicieran contacto con la PCB y tirando cables para conectarlos a la pista correcta.
- El segundo error detectado fue en la huella de la bobina, ya que era más pequeña que las dimensiones reales y no había espacio para ponerla sobre la PCB. Se solucionó con cables, dejando la bobina un nivel por encima del resto de componentes.
- El tercer error se detectó en la huella del potenciómetro, que estaba girada 180 grados.
- El último error se encontró en la huella del conector Jack, donde la huella estaba en espejo. Esto se solucionó soldando el componente por la cara de debajo de la PCB.

Sin embargo, también se detectaron errores en el bloque de audio que no ha sido posible solucionar. Por un lado, se visualizaron las señales PWM de salida de la raspberry en el osciloscopio y se observó que no se generaba ninguna señal en ese punto. Por otro lado, se introdujo una señal PWM externa para comprobar si el diseño electrónico funcionaba, y sería entonces un problema interno de configuración de la raspberry pi, pero también se obtuvo un resultado negativo y no se reproducía ningún

sonido ni a través del altavoz ni de los auriculares. Como este bloque es independiente del resto del circuito, y no afectaba al funcionamiento general, se decidió continuar sin esta parte, porque la única solución era fabricar de nuevo la PCB y el tiempo era limitado.

4.9 Lista de materiales

Otra de las funcionalidades que se había comentado acerca del programa Proteus era que nos permitía obtener una lista de materiales con distintas características sobre cada uno de los componentes utilizados, lo que el propio programa denomina como BOM (*Bill Of Materials*). Podemos añadir la información que consideremos necesaria, ya que se permite la creación de nuevas columnas. En la figura adjunta (*figura 4.19, figura 4.20*), se puede ver que tenemos 7 columnas en las que se recogen los datos de la cantidad, las referencias en el esquemático y que también están serigrafiadas en la PCB, el modelo, la huella, el código de referencia Farnell y el precio por unidad.

Gracias a esta funcionalidad, se obtiene el coste de todos los componentes de manera automática, el cual asciende a 81,60 euros. Sin embargo, a esto habría que sumarle coste por mano de obra, coste de fabricación de la PCB, 10 unidades por 25 euros, y otros muchos factores. También hay que tener en cuenta que en una fabricación a gran escala el coste por unidad es menor; y que, a mayor cantidad de productos, los gastos de envío disminuyen y la rentabilidad es mucho mayor.

El coste de mano de obra de este proyecto se puede calcular en función de las horas que se han dedicado, un total de 300 horas, lo equivalente a 12 créditos. Si ponemos un valor de 10 euros la hora, este coste ascendería a 3000€, los cuales, sumado al coste de materiales y de la PCB, harían un total de en torno a 3100€. Si solo se fabrica 1 unidad el coste es elevado, pero esto disminuye notablemente en una producción a gran escala.

UVa iversidad de Valladolid

Bill Of Materials for Lector Braille

Design Title Lector Braille
Author PAULA GONZÁLEZ VUELTA
Document Number
Revision
Design Created martes, 3 de noviembre de 2020
Design Last Modified viernes, 21 de mayo de 2021
Total Parts In Design 121

7 Modules

Quantity	References	Value	PCB Package	Package Type	Stock Code Farnell	Cost
6	M2,M3,M4,M5,M6,M7	STEPPERMOTOR	CONN-SIL4			0.34
1	M8	RPI0W	TRANS 40 DIL		PIZEROW	10.44
Sub-totals:						12.48

28 Capacitors

Quantity	References	Value	PCB Package	Package Type	Stock Code Farnell	Cost
8	C1,C5,C7,C9,C11, C13,C18,C20	0.1uF	0805_CAP		2332733	0.09
11	C2,C4,C6,C8,C10, C12,C14,C17,C19, C23,C24	10uF	0805_CAP		1867958	0.33
3	C3,C25,C27	1uF	0805_CAP		2896563	0.17
2	C15,C16	4.7uF	0805_CAP		1833814	0.35
2	C21,C22	33nF	0805_CAP		2522471	0.37
1	C26	0.22uF	0805_CAP		1740667	0.22
1	C28	100uF	0805_CAP		2494476RL	1.19
Sub-totals:						7.69

36 Resistors

Quantity	References	Value	PCB Package	Package Type	Stock Code Farnell	Cost
12	R1,R2,R4,R5,R9, R10,R14,R15,R26, R27,R31,R32	0R33	0805_RES		603-PT0805FR-7W0R33L	0.41
15	R3,R6,R8,R11,R13, R16,R18,R22,R24, R25,R28,R29,R30, R33,R35	10k	0805_RES		1160203	0.43
2	R7,R12	100k	0805_RES		3473686	0.03
3	R17,R38,R39	150R	0805_RES		1469877RL	0.02
1	R19	2k	0805_RES		1469884	0.05
1	R20	470R	0805_RES		1469932	0.05
2	R36,R37	270R	0805_RES		1652970	0.05
Sub-totals:						11.62

11 Integrated Circuits

Quantity	References	Value	PCB Package	Package Type	Stock Code Farnell	Cost
6	U1,U5,U6,U7,U8,U9	DRV8847SPWR	TSSOP16		296-53467-6-ND	1.59
2	U2,U11	RP401N501C-TR-FE	SOT23-5		848-RP401N501CTRFE	1.25

Figura 4.19: *Hoja 1 lista de materiales*

1	U3	LM4865MX/NOPB	SOIC-LM4865		926-LM4865MX/NOPB	1.91
1	U4	MCP3004	SO14		1852016	1.90
1	U10	MCP73831	SOT23-5		2709764	0.54
Sub-totals:						16.39

5 Diodes

Quantity	References	Value	PCB Package	Package Type	Stock Code Farnell	Cost
2	D1,D3	BAV99	SOT23		1902422	0.17
1	D2	LED-RED	KA-3022SRC-4.5SF		1142617	0.59
2	D4,D5	CUS10S30H3F	SOD-323-1		757-CUS10S30H3F	0.30
Sub-totals:						1.53

34 Miscellaneous

Quantity	References	Value	PCB Package	Package Type	Stock Code Farnell	Cost
24	3V3,3V7,5V,5V2, B1V+,BJ,CE0#,GND, JX,JY,MISO,MOSI, PWM0,PWM1,S1,S5, S6,S7,S8,S9,SCL, SCLK,SDA,VUSB	PIN	PIN			
1	B1	BATERIA LIPO	CONN-SIL2		2786900	14.96
1	CT1	STX-3000	STX-3000		2839860	0.98
1	J1	USB-B-67503-1020	SD-67503-1020		2426381	0.51
1	J2	JOY_PLAY4	JOY_PLAY4			11.16
1	J19	CONN-SIL2	CONN-SIL2			
2	L1,L2	4.7uH	SLF6028T-4R7M1R6-PF		1670018	0.21
1	LS1	SPEAKER	CONN-SIL2		1502732	2.54
1	P1	PTV09-2	PTV09A-2		2519592	0.73
1	SW1	SW-SPDT	SW-EG1224		EG2585-ND	0.58
Sub-totals:						31.88

Totals: 81.60

viernes, 21 de mayo de 2021 10:46:13

Figura 4.20: *Hoja 2 lista de materiales*

5. Actualización del software del lector Braille digital

5.1 Introducción

En este capítulo se explicará como se ha llevado a cabo la implementación del software del dispositivo, una vez que se han montado todos los componentes de forma correcta.

En este caso, el objetivo es actualizar la funcionalidad ya desarrollada previamente en un trabajo anterior, en base a la modificación que se ha realizado del hardware del dispositivo. Se modificará la parte de los motores, integrando la funcionalidad de mover motores paso a paso a través de los puentes H. También se comprueba la compatibilidad del software con el joystick en este nuevo diseño.

5.2 Entorno de trabajo

El código ha sido desarrollado en *Python 3*, porque raspberry cuenta con numerosas librerías que incluyen funciones que serán de gran utilidad para el desarrollo de la funcionalidad requerida en el dispositivo. Para probar el código se utilizó VNC Viewer, software que nos permite acceder al escritorio Raspbian de forma remota mediante la dirección IP de la raspberry pi. Todas las instrucciones para el manejo y compilación del código que compone el programa se realizaran desde el terminal que nos ofrece Raspbian, *figura 5.1*. Por ejemplo, el comando de compilación utilizado es *python nombreFichero.py*. Para desarrollar el software se utilizó el programa *Anaconda* en nuestro ordenador local de trabajo, un editor de código compatible con el lenguaje de programación utilizado, por lo tanto, será necesaria una herramienta que nos permita transferir fácilmente los ficheros desde nuestro ordenador local a la raspberry pi, como veremos a continuación.

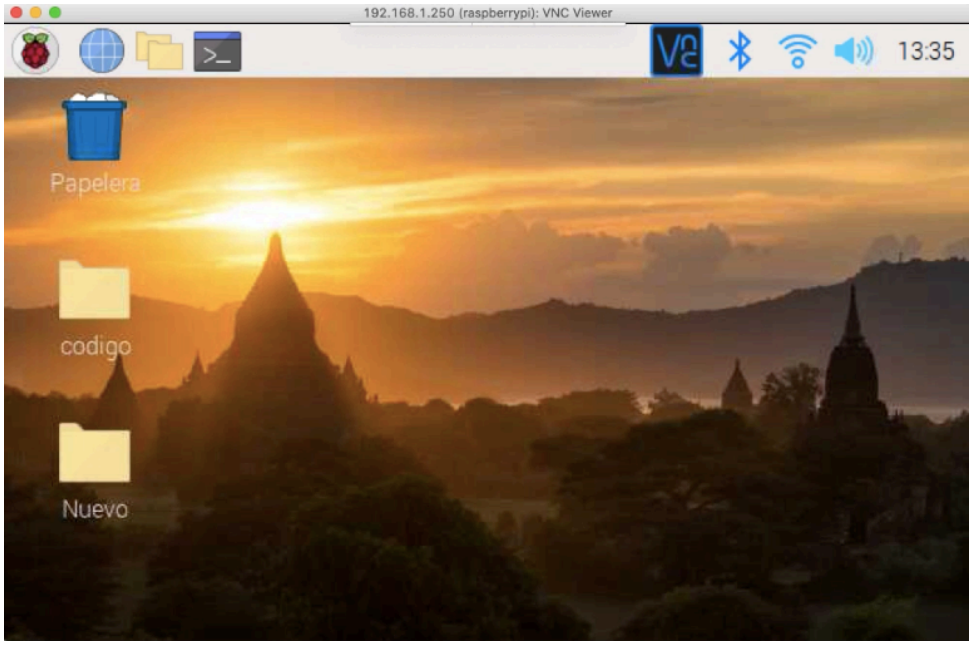


Figura 5.1: *Entorno de escritorio Raspbian*

VNC Viewer no es la única manera de manejar la raspberry pi. Se puede acceder y controlar la raspberry pi de dos formas: mediante una conexión remota a su escritorio, como ya se ha mencionado, o mediante el servidor ssh desde el terminal de nuestro ordenador local, escribiendo el comando *ssh pi@dirección-IP*. En este último caso, no tendremos acceso al escritorio, y se controlará todo desde el terminal. Además, gracias a esta última vía de conexión, tendremos la posibilidad de transferir los archivos desde nuestro escritorio a la raspberry pi. Para ello, nos situamos desde el terminal de nuestro ordenador en la carpeta origen del fichero a transferir y se escribe el siguiente comando en el terminal: *scp "nombreFichero" pi@"direccionIP": "ruta/carpeta/destino"*. En la *figura 5.2* podemos ver un ejemplo de uso de este comando.

Figura 5.2: *Ejemplo de transferencia de ficheros a la raspberry pi*

5.3 Configuración Raspberry pi 0 W

Para poder acceder a la raspberry pi y ejecutar el código que pone en marcha el dispositivo, es necesario realizar unas configuraciones previas en nuestra raspberry pi. A continuación, se detallan los pasos a seguir cuando se enciende el dispositivo la primera vez para configurar su conexión a internet.

En primer lugar, nos conectamos al escritorio de forma remota mediante un cable HDMI para conectarnos a nuestra red WIFI y desde el menú de configuración mostrado en la *figura 5.3* se asigna una dirección IP fija a la raspberry pi para acceder desde el VNC Viewer [27]: en nuestro caso escogemos la dirección 192.168.1.250.

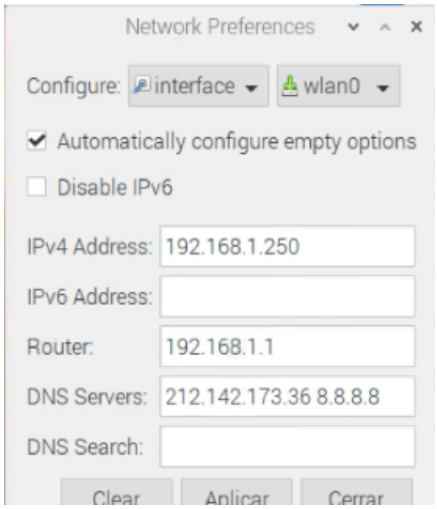


Figura 5.3: *Menú donde configurar IP Fija*

Una vez tenemos esta configuración realizada ya podremos conectarnos a nuestra raspberry desde el ordenador local y controlarla.

5.4 Actualización de la funcionalidad de los motores

Los motores paso a paso requieren de un software más complejo que los electroimanes para funcionar, al ser controlados por los puentes H mediante el bus I2C. En las versiones anteriores, dentro del software podíamos encontrar un método llamado *variarLED()* que controlaba el movimiento de los electroimanes, como

podemos ver en la *figura 5.4.*. Dicho método va a ser sustituido en esta nueva versión por otro que permita controlar el movimiento de los motores paso a paso y al que llamaremos *variarMotor()*. Como novedad, se creará a mayores un nuevo método para la configuración inicial de los puentes H y se llamará *initDRV8847s()*. A continuación, explicaremos de forma detallada estas dos funciones.

```
9
10  def variarLED(channel, entrada):
11
12      #escoger canal
13      if channel == 1:
14          canal=17
15      if channel == 2:
16          canal=27
17      if channel == 3:
18          canal=22
19      if channel == 4:
20          canal=5
21      if channel == 5:
22          canal=6
23      if channel == 6:
24          canal=13
25
26      #encender motor, cambiado para los motores
27      if entrada == "0":
28          GPIO.output(canal,GPIO.HIGH)
29      if entrada == "1":
30          GPIO.output(canal,GPIO.LOW)
31
```

Figura 5.4: *Método variarLED() de la versión anterior*

En la *figura 5.5* se representa el código del método *initDRV8847s()* que servirá para asignar a cada chip una dirección para comunicarnos a través del bus I2C. Para utilizar las instrucciones del protocolo I2C en raspberry es necesario instalar previamente la librería *smbus,* para ello, escribimos en el terminal *sudo apt-get install python-smbus* y *sudo apt-get install i2c-tools*. Para utilizarla la importaremos con el *comando import smbus as bus*. Entre las funciones que ofrece esta librería para controlar dispositivos compatibles con I2C están la lectura y escritura en los dispositivos [28].

```
import smbus as bus
import RPi.GPIO as GPIO
from moverMotores import variarMotor
I2Cbus = bus.SMBus(1) #creamos el bus I2C
#Asignamos a cada chip una dirección para comunicarnos a través del bus I2C
def initDRV8847s():
    GPIO.setmode(GPIO.BCM) #referencias GPIO para la numeración de los pines,
                           #en lugar de los pines físicos
    nFaultPins = [17,27,23,24,25,12]
    dirDevice = [0x12,0x13,0x14,0x15,0x16,0x17]
    i=0

    for pin in nFaultPins:
        GPIO.setup(pin,GPIO.OUT)
        GPIO.output(pin,True)
    #fijamos la dirección de cada dispositivo esclavo
    I2Cbus.write_byte_data(0x60,0x02,0x40) #bit DISFLT=1

    for pin in nFaultPins:
        GPIO.output(pin,False)

    for pin in nFaultPins:
        GPIO.output(pin,True)
        I2Cbus.write_byte_data(0x60,0x00,dirDevice[i])
        GPIO.output(pin,False)
        i+=1

    for pin in nFaultPins:
        GPIO.setup(pin,GPIO.OUT)
        GPIO.output(pin,True)

    for dirr in dirDevice:
        I2Cbus.write_byte_data(dirr,0x02,0x00) #bit DISFLT=0

    for pin in nFaultPins:
        GPIO.setup(pin,GPIO.IN)

    #ponemos todos los motores en su estado bajo de forma inicial
    for motor in range (0,5):
        variarMotor(motor,"0")

    GPIO.cleanup()
```

Figura 5.5: *Método initDRV8847s()*

Esta función no cuenta con argumentos de entrada y se llamará una sola vez al encender el dispositivo desde el método *menuOpciones(),* a modo de configuración inicial, porque cada vez que lo reseteemos los registros de los puentes H volverán a sus valores por defecto. Antes de nada, es necesario definir como variables una cadena con la dirección que vamos a dar a cada dispositivo y otra con el número de cada pin nFAULT en la raspberry. Hay que destacar que la numeración de los pines GPIO se puede configurar de dos formas diferentes [29]: según los pines externos, usando el comando *GPIO.setmode(GPIO.BOARD),* o según los pines internos de la raspberry, usando *GPIO.setmode(GPIOBCM)*, en nuestra función usaremos este último. En la *tabla 5.1* se resumen los datos más relevantes de está función.

Dispositivo Esclavo	Pin nFAULT (GPIO.BCM)	Pin nFAULT (GPIO.BOARD)	Dirección por defecto	Dirección final
U1	17	11	0x60	0x12
U5	27	13	0x60	0x13
U6	23	16	0x60	0x14
U7	24	18	0x60	0x15
U8	25	22	0x60	0x16
U9	12	32	0x60	0x17

Tabla 5.1: *Resumen de los datos usados para el método initDRV8847s()*

Una vez definidas las variables, el procedimiento que hay que seguir para cambiar la dirección de cada dispositivo esclavo es la siguiente [14]:

- Primero se configuran todos los pins nFAULT como salidas, usando el comando *GPIO.setup(pin,GPIO.OUT)*.
- En segundo lugar, se accede al registro de control IC2 para poner el bit 6 (DISFLT) a "1" en todos los dispositivos esclavos conectados al bus. Para ello, se usa el método *I2Cbus.write_byte_data(0x60,0x02,0x40)*, donde el primer argumento de entrada es la dirección del chip en el bus, el segundo es la dirección del registro de control I2C, donde queremos escribir, y el último es el valor que queremos escribir en ese registro.
- En tercer lugar, se pondrá el pin nFAULT de todos los chips en baja con el comando *GPIO.output(pin,False)* y, de forma secuencial, se irá poniendo en alta solamente uno de ellos en cada iteración, de este modo, solamente un chip estará conectado al dispositivo maestro y podremos cambiar la dirección de ese dispositivo esclavo. Para escribir la dirección nueva se usa nuevamente el comando de escritura *I2Cbus.write_byte_data(0x60,0x00,dirDevice[i])*, con la diferencia de que ahora se escribe sobre el registro *Slave Address*.
- Una vez se hayan asignado las direcciones únicas a cada dispositivo, se volverá a poner el bit DISFLT a "0" usando *I2Cbus.write_byte_data(dirr,0x02,0x00)*, donde dirr será cada una de las direcciones del bus que hemos asignado previamente.

- Finalmente, establecemos de nuevo los pines nFAULT como entradas y ya podremos acceder a cada dispositivo esclavo de forma independiente para controlar los motores.

Para comprobar que el método se ha ejecutado correctamente y que todos los dispositivos están correctamente conectados a la raspberry escribimos en el terminal el comando *sudo i2cdetect -y 1*, y se mostrarán todas las direcciones conectadas al bus, como podemos ver en la *figura 5.6*.

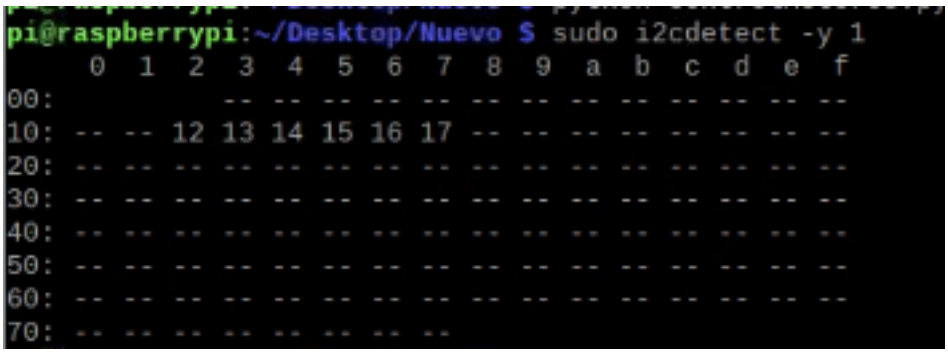


Figura 5.6: *Direcciones conectadas al bus I2C*

El otro método que se crea es el de *variarMotor()*, y como se muestra en la *figura 5.7*, cuenta con dos argumentos de entrada, el primero para indicar cual de los seis motores se va a mover y el segundo para indicar el estado de ese motor: “0” si queremos que el motor baje y “1” si queremos que el motor suba. De nuevo, se definen las variables con los valores de las direcciones de los motores y de los pines nFAULT y, a mayores, se definen otras dos cadenas: una con la secuencia que activará el modo avance del motor y otra con la secuencia que activará el modo retroceso, recordamos que esta secuencia ha sido explicada en un apartado anterior. Para controlar los motores será necesario acceder al registro de control IC1 de cada dispositivo esclavo.

```
import smbus as bus
import RPi.GPIO as GPIO
import time
I2Cbus = bus.SMBus(1) #creamos el bus I2C

def variarMotor(motor, estado):

    GPIO.setmode(GPIO.BCM)
    nFaultPins = [17,27,23,24,25,12]
    dirr = [0x12,0x13,0x14,0x15,0x16,0x17]
    i=0

    GPIO.setup(nFaultPins[motor],False)

    seqBajada = [0x09,0x05,0x06,0x0A]
    seqSubida = [0x0A,0x06,0x05,0x09]
    # seq1=0x09 #IN1, IN4 = 1
    # seq2=0x05 #IN1, IN3 = 1
    # seq3=0x06 #IN2, IN3 = 1
    # seq4=0x0A #IN2, IN4 = 1
    I2Cbus.write_byte_data(dirr[motor],0x01,0x04) #I2CBC=1 para usar
                                #los bits INx y MODE=00 (4-pins interface)
    a=I2Cbus.read_byte_data(dirr[motor],0x01)

    if estado == "0":
        #BAJADA
        while i<50:
            for s in seqBajada:
                I2Cbus.write_byte_data(dirr[motor],0x01,a|s<<3)
                time.sleep(0.001)
            i+=1
    if estado == "1":
        #SUBIDA
        while i<50:
            for s in seqSubida:
                I2Cbus.write_byte_data(dirr[motor],0x01,a|s<<3)
                time.sleep(0.001)
            i+=1

    I2Cbus.write_byte_data(dirr[motor],0x01,0x00)
```

Figura 5.8: *Método variarMotor()*

Usando el comando de *I2Cbus.write_byte_data(dirr[motor],0x01,0x04)* accedemos al registro IC1 del dispositivo esclavo correspondiente, según lo que indique el argumento de entrada *motor*, para poner el bit 2 (I2CBC) en "1" e indicar al dispositivo que usaremos los bits INx para controlar los motores. Seguidamente, leemos dicho registro con el comando *I2Cbus.read_byte_data(dirr[motor],0x01)* y se guarda el valor en la variable "a". Este paso es necesario porque para mover el motor volveremos a usar el comando de escritura para acceder al registro IC1, sobrescribiendo el valor que hasta ese momento tenga. Como es necesario mantener el bit 2 en "1" el valor que se escribirá en el registro será $a \mid s<<3$. Usando la función OR "|" nos aseguramos de que el bit 2 mantiene su valor y con la función de desplazamiento hacia la izquierda "<<" escribimos la secuencia entre los bits 3 y 6, que son los que corresponden con los bits IN1, IN2, IN3 e IN4, respectivamente. Para finalizar, se vuelven a poner todos los bits de ese registro a "0".

Por otro lado, se comprobó el bloque del joystick con el software del proyecto anterior [5], porque no era necesario realizar ningún cambio al haber realizado un hardware que era compatible. La prueba realizada consistía en comprobar que se detectaba el movimiento del joystick y se podía seleccionar cualquiera de las 3 opciones disponibles. El resultado obtenido fue correcto, y no se detectaron problemas de compatibilidad.

6. Conclusiones y líneas futuras

Para finalizar, vamos a comentar y analizar en que medida se han alcanzado los objetivos propuestos al inicio del trabajo, así como los problemas encontrados y posibles mejoras para tener en cuenta para una optimización futura de este dispositivo.

Si que es verdad que los objetivos que se han alcanzado no han sido del todo los esperados. Debido a distintos problemas, que ahora comentaremos, se ha ralentizado el proceso de diseño del lector y ha sido necesario encaminar el trabajo de otro modo. Sin embargo, esto es un aprendizaje positivo en el sentido de saber afrontar los problemas y saber adaptarse a cambios que puedan surgir inesperadamente. La idea inicial consistía en rediseñar el hardware integrando una nueva funcionalidad de audio y cambiando los electroimanes por los motores paso a paso, así como añadir el software necesario para que la nueva electrónica funcione. Sin embargo, el bloque de audio no ha podido ser integrado en el software al surgir problemas que no han podido ser solucionados y solamente se ha hecho una descripción del hardware implementado. Por otro lado, los motores que se habían escogido no han podido ser montados en la PCB por lo que se tomó la decisión de caracterizar distintos modelos y analizar ventajas y desventajas de cada uno de ellos.

A excepción del problema ya comentado del audio, el resto de los problemas surgidos han podido ser corregidos y no ha supuesto inconveniente en desarrollar el dispositivo como se debía.

Finalmente, y teniendo en cuenta los objetivos que se deseaban conseguir en este trabajo inicialmente, se plantean varios puntos futuros de mejora:

- En primer lugar, analizar el problema surgido con el audio: por un lado, es posible que la electrónica diseñada no sea del todo compatible con la raspberry pi y sea necesario realizar cambios en el esquemático, y, por otro lado, puede que este modelo de raspberry pi no sea capaz de generar audio, lo que supondría que habría que eliminar esta parte del diseño.
- Otro punto importante es tomar una decisión sobre el motor que se quiere montar en el dispositivo, teniendo en cuenta la caracterización que se ha llevado a cabo en este trabajo.
- Por último, seguir añadiendo nuevas funcionalidades al software del dispositivo, algo que en este trabajo no ha sido el foco principal de mejora.

7. Bibliografía y referencias

[1] BraiBook Braille. *BraiBook: el dispositivo electrónico de lectura para personas invidentes*. Available at: <https://braibook.com/braibook-el-dispositivo-electronico-de-lectura-para-personas-invidentes/>

[2] Once.es. *Braille en español, alfabeto, signos, aprendizaje - Web ONCE*. Available at: <https://www.once.es/servicios-sociales/braille>

[3] BraiBook Braille. *Lee en braille*. Available at: <https://braibook.com/lee-en-braille/>

[4] Sergio Pérez Callejo. (2020). *Trabajo de Fin de Grado: Diseño de prototipos en Raspberry e impresión 3D de carácter social*.

[5] David García Crespo (2020). *Trabajo de Fin de Grado: diseño de prototipos en Arduino orientados al Internet de las Cosas (IoT) de carácter social*

[6] Raspberrypi.org. Available at: <https://www.raspberrypi.org/products/raspberry-pi-zero-w/>

[7] Tme.eu. *Motor paso a paso – tipos y ejemplos del uso de motores paso a paso | Distribuidor de componentes electrónicos. Multisort Elektronik*. Available at: <https://www.tme.eu/es/news/library-articles/page/41861/Motor-paso-a-paso-tipos-y-ejemplos-del-uso-de-motores-paso-a-paso/>

[8] Electroimanes. *Solenoides y actuadores Solenoid - 5V (Small)* Available at: https://www.mouser.es/ProductDetail/SparkFun/ROB-11015?qs=WyAARYrbSnb40pNKVm6nzg==&mgh=1&vip=1&gclid=Cj0KCQjws4aKBhDPARIsAIWH0JW6w2ePXxzQbCh86py90TW0OTUsHwnGZ3-3l_1hEvIFpCJkgf0Vi_waAjzaEALw_wcB

[9] aliexpress.com. *Actuador lineal mini de 12mm, engranaje planetario de precisión, Motor paso a paso, ultrafino, 4mm, 2 fases, 4 cables.* Available at: <https://es.aliexpress.com/item/4000084515397.html?spm=a219c.12057483.0.0.6dad4deb00BoxA>

[10] Es.aliexpress.com. *Mini Motor paso a paso de 5MM, dos fases, cuatro cables, caja de cambios planetaria, deslizamiento de Metal, rodamientos de bolas de precisión.* Available at: <https://es.aliexpress.com/i/4000001735582.html?spm=a219c.12057483.0.0.29924bd92XyIqG>

[11] Es.aliexpress.com. *Motor paso a paso de cuatro cables de dos fases, con deslizador, varilla de tornillo de 9 MM, Motor de precisión para enfoque de cámara Digital, 6MM.* Available at: <https://es.aliexpress.com/i/4000172838008.html?spm=a219c.12057483.0.0.3a23153bVFyoBo>

[12] aliexpress.com. *Micromotor pequeño de 20 piezas, paso a paso con tornillo DC 5V D3.3mm * H4.5mm* Available at: <https://es.aliexpress.com/item/32222397990.html>

[13] Es.wikipedia.org. 2021. *Puente H (electrónica) - Wikipedia, la enciclopedia libre*. Available at: <https://es.wikipedia.org/wiki/Puente_H_(electr%C3%B3nica)#/media/Archivo:H_bridge_operating.svg>

[14] Texas Instruments. *DRV8847 Dual H-Bridge Motor Driver*. Available at: <https://www.ti.com/lit/ds/symlink/drv8847.pdf?ts=1631448283633&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FDRV8847>

[15] Amazon.es. Available at: <https://www.amazon.es/OTOTEC-Juego-Herramientas-Mando-Playstation/dp/B07M5ZZQRQ/ref=sr_1_5?__mk_es_ES=%C3%85M%C3%85%C5%BD%C3%95%C3%91&dchild=1&keywords=joystick+recambio+ps4&qid=1620298992&sr=8-5>

[16] Microchip Technology. *MCP3004-I/P, 2.7V 4-Channel/8-Channel 10-Bit A/D Converters with SPI Serial Interface*. Available at: <https://www.mouser.es/datasheet/2/268/MCHPS02710_1-2520919.pdf>

[17] Adafruit Learning System. *Adding Basic Audio Ouput to Raspberry Pi Zero*. Available at: <https://learn.adafruit.com/adding-basic-audio-ouput-to-raspberry-pi-zero>

[18] *Batería Recargable, 3.7 V, Polímero de Litio, 2 Ah, JST* https://es.farnell.com/mikroelektronika/mikroe-1120/bater-a-de-litio-pol-mero-3-7v/dp/2786900?ost=MIKROE1120+MIKROELEKTRONIKA&CMP=os_pdf-datasheet

[19] Ricoh Electronic Devices Company. *RP401N501C-TR-FE: RP401x SERIES, HIGH EFFICIENCY, SMALL PACKAGES, STEP-UP DC/DC CONVERTERS*. Available at: <https://www.mouser.es/datasheet/2/792/rp401-ea-1770437.pdf>

[20] Microchip. *MCP73831-2ACI/MC: Miniature Single-Cell, Fully Integrated Li-Ion, Li-Polymer Charge Management Controllers*. Available at: <https://4donline.ihs.com/images/VipMasterIC/IC/MCHP/MCHPS05673/MCHPS05673-1.pdf?hkey=6D3A4C79FDBF58556ACFDE234799DDF0>

[21] *Proteus Libraries*: https://componentsearchengine.com/library/proteus?gclid=Cj0KCQjws4aKBhDPARIsAIWH0JU17FsYghqlG9uKwnwGNgK3i8tvQBuqiY6NuvMolMs9uaWbVYHLrbEaAqK_EALw_wcB

[22] VNC Viewer. *Download*: https://www.realvnc.com/es/connect/download/viewer/

[23] Raspberrypi.org. Available at: https://www.raspberrypi.org/software/

[24] Tinkercad. *Tinkercad | From mind to design in minutes*. Available at: <https://www.tinkercad.com/>

[25] Es.pinout.xyz. *DPI Raspberry Pi GPIO pinout*. Available at: https://es.pinout.xyz/pinout/dpi

[26] Pcbway.com. *China PCB Prototype & Fabrication Manufacturer - PCB Prototype the Easy Way*. Available at: <https://www.pcbway.com/?gw1&campaignid=172480651&adgroupid=8787904531&feeditemid=&targetid=kwd-96217560494&loc_physical_ms=1005546&matchtype=p&network=g&device=c&devicemodel=&creative=377957049820&keyword=pcbway&placement=&target=&adposition=&gclid=Cj0KCQjws4aKBhDPARIsAIWH0JVlnrJdlpnQghXE55mGYAaUSHJWD0DPW-VqVsVLetyf3Eb9NqmjLx8aAj3FEALw_wcB>

[27] Asociación Programo Ergo Sum. *Direcciones IP en Raspberry Pi*. Available at: <https://www.programoergosum.es/tutoriales/direcciones-ip-en-raspberry-pi/>

[28] HETPRO/TUTORIALES. *Python: I2C, uso y configuración - HETPRO/TUTORIALES*. Available at: <https://hetprostore.com/TUTORIALES/python-i2c-uso-y-configuracion/>

[29] Prometec.net. *Usando los GPIO con Python | Tienda y Tutoriales Arduino*. Available at: <https://www.prometec.net/usando-los-gpio-con-python/>