



# UNIVERSIDAD DE GRANADA

TRABAJO FIN DE GRADO  
INGENIERÍA INFORMÁTICA

## Diseño y Desarrollo de un Dispositivo de Interfaz Humana

---

ModernWood

**Autor**  
Carlos López Martínez

**Director**  
Jesús González Peñalver



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE  
TELECOMUNICACIÓN

Granada, 21 de junio de 2024









MODERN WOOD

---

Teclado ISO 105 Español

**Autor**

Carlos López Martínez

**Director**

Jesús González Peñalver



# Diseño y Desarrollo de un Dispositivo de Interfaz Humana: Teclado ISO 105 Español

Carlos López Martínez

**Palabras clave:** ISO, PlatformIO, Microcontrolador, ESP32-S3, Bluetooth, Wireless, USB, Windows, Linux

## Resumen

Creación de un dispositivo de interfaz humana siguiendo el estándar ISO 105 en español. Enfocado en la creación de un dispositivo de interfaz humana, este proyecto se propone diseñar y desarrollar un teclado conforme al estándar ISO 105 en español, ya que no existe gran variedad de estos en el mercado. Para alcanzar este objetivo, se empleará la plataforma "PlatformIO" junto con un microcontrolador ESP32-S3, aprovechando sus capacidades de conectividad y procesamiento. El dispositivo permitirá una interacción con SO Windows y Linux por USB (Cableado) y Bluetooth (Wireless)

Para todo el desarrollo del dispositivo se tendrá en cuenta el coste del mismo, abordando su reparabilidad, estética y en todo momento el rendimiento y funcionamiento, sin perder el objetivo de que sea un producto de alta gama. El dispositivo tiene que funcionar de forma correcta tanto de forma inalámbrica como cableada independientemente, pudiéndose bajar los costes si solo se desea una versión cableada.

El dispositivo dispondrá de una pantalla LCD a color para mostrar información y configuración del mismo, así como código ampliable y su asignación a teclas especiales.



# **Design and Development of a Human Interface Device: ISO 105 Spanish Keyboard**

Carlos López Martínez

**Palabras clave:** ISO, PlatformIO, microcontroller, ESP32-S3, Bluetooth, Wireless, USB, Windows, Linux

## **Abstract**

Creation of a human interface device following the ISO 105 standard in Spanish. Focused on the creation of a human interface device, this project aims to design and develop a keyboard in accordance with the ISO 105 standard in Spanish, as there is not a wide variety of these in the market. To achieve this goal, the "PlatformIO" platform will be used along with an ESP32-S3 microcontroller, taking advantage of its connectivity and processing capabilities. The device will allow interaction with Windows and Linux OS via USB (Wired) and Bluetooth (Wireless).

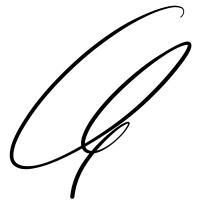
Throughout the development of the device, its cost will be taken into account, as it must be affordable, at least as a high-end product. Its reparability, aesthetics, and performance and operation will also be considered at all times. The device has to function correctly both wirelessly and wired independently. Costs can be reduced if only a wired version is desired.

The device will have a color LCD screen to display information and configuration of the same. As well as expandable code and its assignment to special keys.



---

Yo, **Carlos López Martínez**, alumno de la titulación INGENIERÍA INFORMÁTICA de la Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada, con DNI 20888530E, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.



Fdo: Carlos López Martínez

21 de junio de 2024



---

**D. Jesús González Peñalver**, Catedrático del departamento de Arquitectura y Tecnología de Computadores de la Universidad de Granada.

**Informa:**

Que el presente trabajo, titulado ***Diseño y Desarrollo de un Dispositivo de Interfaz Humana***, ha sido realizado bajo su supervisión por **Carlos López Martínez**, y autoriza la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expide y firma el presente informe en Granada a 21 de junio de 2024

**El director:**

**Jesús González Peñalver**



# Agradecimientos

A mi familia, por todo el apoyo que me han dado a lo largo de todos los años. A mis amigos por las ideas y momentos de inspiración. Y a todos los profesores que me han ayudado y han logrado hacer del Grado de Ingeniería Informática una gran experiencia.



# Índice general

<b>1. Introducción</b>	<b>5</b>
1.1. Motivación . . . . .	5
1.2. Estado actual de las alternativas . . . . .	7
1.2.1. Conectividad . . . . .	7
1.2.2. Formatos . . . . .	9
1.2.3. Precios . . . . .	9
1.2.4. Otros teclados personalizados . . . . .	10
<b>2. Especificación del Sistema</b>	<b>11</b>
2.1. Requisitos . . . . .	11
2.1.1. Requisitos funcionales . . . . .	11
2.1.2. Requisitos no funcionales . . . . .	12
2.2. Especificaciones . . . . .	13
2.2.1. Especificación Hardware . . . . .	13
2.2.2. Especificación Software . . . . .	14
2.3. Planificación . . . . .	15
<b>3. Diseño Y Prototipado</b>	<b>17</b>
3.1. Teclados, entendiendo sus partes . . . . .	17
3.2. Investigación: Herramientas y Tecnologías . . . . .	26
3.3. Alternativas de diseño . . . . .	26
3.3.1. Elección de las herramientas . . . . .	27
3.4. Elección de Hardware . . . . .	29
3.4.1. Layout . . . . .	29
3.4.2. Componentes Electrónicos . . . . .	31
3.4.3. Materiales . . . . .	39
3.4.4. Partes del teclado . . . . .	39
3.5. Elección de Software . . . . .	40
3.5.1. Entorno de desarrollo . . . . .	40
3.5.2. Librerías . . . . .	42
3.6. Presupuesto . . . . .	42

## ÍNDICE GENERAL

<b>4. Circuitos</b>	<b>47</b>
4.1. Búsqueda de componentes . . . . .	47
4.1.1. Creación de componentes . . . . .	49
4.2. Diseño Esquemático . . . . .	50
<b>5. PCB</b>	<b>57</b>
5.1. Diseño físico . . . . .	57
5.1.1. Diseño de la distribución . . . . .	58
5.1.2. Medidas Físicas . . . . .	58
5.1.3. Creación de la placa en Eagle . . . . .	60
<b>6. Carcasa</b>	<b>69</b>
6.1. Diseño físico . . . . .	69
6.1.1. Medidas Físicas . . . . .	70
6.1.2. Ergonomía . . . . .	71
6.2. Fabricación . . . . .	72
<b>7. Programación</b>	<b>75</b>
7.1. Plataforma . . . . .	75
7.1.1. Compilación y entorno . . . . .	77
7.2. Interfaz . . . . .	78
7.2.1. Menú . . . . .	79
7.2.2. Barra de estado . . . . .	81
7.3. Funcionalidad . . . . .	81
7.3.1. Estados . . . . .	81
7.3.2. Conectividad . . . . .	84
7.3.3. Leds . . . . .	84
7.3.4. Macros . . . . .	84
7.4. Boot . . . . .	85
<b>8. Prototipos</b>	<b>87</b>
8.1. Versiones . . . . .	87
8.1.1. V1: Correcciones . . . . .	87
8.1.2. V2: Correcciones . . . . .	88
8.1.3. V3: Modificaciones . . . . .	88
8.2. Problemas y Errores . . . . .	88
8.2.1. Corriente inversa en el regulador de tensión . . . . .	88
8.2.2. Voltaje en la salida del regulador de tensión . . . . .	89
8.3. Montaje . . . . .	90
<b>9. Validación</b>	<b>93</b>
9.1. Pruebas Eléctricas . . . . .	93
9.2. Pruebas en <a href="#">Windows</a> . . . . .	94
9.3. Pruebas en <a href="#">Linux</a> . . . . .	95

---

**ÍNDICE GENERAL**

---

**ÍNDICE GENERAL**

<b>10.Mejoras</b>	<b>99</b>
10.1. Posibles mejoras . . . . .	99
10.1.1. Software . . . . .	99
10.1.2. Hardware . . . . .	100
10.1.3. Materiales . . . . .	100
10.2. Consideraciones Personales . . . . .	101
<b>11.Conclusiones</b>	<b>103</b>
11.1. Trabajo futuro . . . . .	104
<b>12.Apéndice</b>	<b>105</b>
12.1. Circuitos . . . . .	105
12.1.1. Esp32S3 con Agujero . . . . .	105
12.2. PCB . . . . .	107
12.2.1. Dimensiones físicas . . . . .	107
12.3. Programación . . . . .	109
12.3.1. Código fuente . . . . .	109
12.3.2. Estructuras . . . . .	109
12.3.3. Funciones . . . . .	111
12.3.4. PID y VID . . . . .	115
12.3.5. Leds . . . . .	116
12.3.6. Macros . . . . .	117
12.4. Pruebas . . . . .	119
12.5. Ensamblaje . . . . .	121
12.5.1. Errores . . . . .	121
12.5.2. Montaje Nuevo . . . . .	122
<b>Glosario</b>	<b>129</b>
<b>Indice de figuras</b>	<b>133</b>
<b>Bibliografia</b>	<b>138</b>



# Capítulo 1

## Introducción

### 1.1. Motivación

Cada vez es más común el uso de ordenadores y sistemas informáticos en todos los ámbitos de la vida, por lo que el uso de dispositivos para poder interactuar con estos también es igual de común. Desde hace muchísimo tiempo, el dispositivo más usado para interactuar con los ordenadores y máquinas a lo largo del mundo es y ha sido el teclado. Existen teclados de todo tipo, formas, distribuciones y tecnologías. Todos ellos con el mismo objetivo, introducir caracteres en el sistema.

Personalmente, desde que puedo recordar siempre he usado el ordenador de casa y he crecido con el teclado a mi lado. He pasado por muchos teclados a lo largo de mi vida y todos ellos han acabado más o menos de la misma forma, estropeados o guardados porque me cansaba del teclado y de como era la sensación al usarlo. Reparar estos teclados era igual de costoso que el propio problema que tenían.

Intenté buscar siempre dispositivos que satisficieran mis necesidades; robustos, fáciles de arreglar, duraderos, inalámbricos y elegantes. En el mercado existen varios de estos, pero era mi último requisito el que hacía que mi búsqueda no arrojase ningún resultado, que fuera en español. Al fin y al cabo, escribo en español y estoy acostumbrado a una distribución española, por lo que dado las exigencias que tengo como usuario y que no hay mercado para lo que busco, decidí embarcarme en este proyecto para llenar el hueco que hay en el mercado y satisfacer mis demandas.

Mi desafío no solo radica en encontrar el teclado perfecto para mí, sino también en transformar la concepción convencional de diseño y fabricación de teclados. Para abordar este problema, mi enfoque va más allá de simplemente crear otro dispositivo; mi objetivo es redefinir lo que se considera estándar mediante la implementación de nuevas técnicas y conceptos inno-

vadores.

En lugar de adherirme a las técnicas de fabricación y ensamblaje convencionales, que a menudo resultan en dispositivos difíciles de reparar y poco robustos, planeo adoptar un enfoque disruptivo. La fabricación de la placa de circuito impreso (**PCB**) será un elemento central de esta estrategia. Al diseñar una **PCB** que priorice la facilidad de reparación y la durabilidad, se espera superar las limitaciones de los teclados tradicionales.

La estética del teclado también experimentará un cambio significativo en mi proyecto. En lugar de centrarme únicamente en el aspecto visual, mi enfoque se orientará hacia la funcionalidad y la simplicidad elegante. Esto implica la eliminación de partes estéticas innecesarias para priorizar la funcionalidad y la durabilidad. La forma seguirá a la función, y mi objetivo es crear un teclado que no solo sea una herramienta eficiente, sino también un testimonio de la belleza en la simplicidad.

La modularidad será una característica clave de mi diseño. La capacidad de desmontar y reemplazar fácilmente componentes permitirá a los usuarios personalizar su experiencia según sus necesidades específicas. Desde interruptores hasta placas de circuito, cada elemento será diseñado para ser modular, facilitando tanto la personalización como el mantenimiento.

En términos de conectividad, mi teclado ofrecerá versatilidad. No solo será inalámbrico, aprovechando la comodidad de la tecnología sin cables, sino que también contará con una opción por cable para situaciones donde la estabilidad de la conexión sea prioritaria. Esta dualidad busca ofrecer a los usuarios la flexibilidad necesaria para adaptarse a diferentes entornos y preferencias.

En resumen, mi proyecto busca romper las convenciones establecidas en el diseño y la fabricación de teclados. A través de la implementación de varias tecnologías, un enfoque renovado en la estética, así como la incorporación de características como la modularidad y el diseño simple. Aspiro a crear un teclado que no solo satisfaga mis necesidades y pueda llegar a llenar ese vacío en el mercado existente y crear, de una manera u otra, un producto de buena calidad.

## 1.2. Estado actual de las alternativas

Actualmente, existen muchos teclados, formas, distribuciones, materiales y precios. [14] Dado que el nuestro, aunque pueda ser fabricado en grandes cantidades, va a ser fabricado como prototipo una única vez. Vamos a mirar las opciones que se conocen como **DIY** o personalizados, que son las correspondientes a este tipo de mercado donde se fabrican bajo un precio menos ajustado y en menos cantidades.

En el ámbito de los teclados **DIY** o personalizados, se encuentran diversas opciones que permiten a los usuarios crear su propio dispositivo según sus preferencias y necesidades. Estas alternativas suelen destacar por ofrecer un mayor nivel de personalización en términos de diseño, disposición de teclas, interruptores y retro-iluminación, en comparación con los teclados convencionales o fabricados en masa. [25]

Una de las opciones más populares para este tipo de teclados son las placas base personalizables. Estas placas permiten a los usuarios seleccionar y soldar sus propios interruptores y estabilizadores, lo que brinda una libertad total en la elección de la disposición de las teclas. Además, suelen admitir la programación de macros y asignación de funciones a través de software.

También existe la posibilidad de elegir interruptores mecánicos específicos, siendo una característica clave. Existen diversos tipos de interruptores, como los Cherry MX, Gateron, Kailh, entre otros, cada uno con características únicas en términos de tacto, recorrido y sonoridad. Además, acompañando a estos interruptores, siempre hay lo que se denomina como **Keycaps**, que son los componentes que cubren los interruptores y que al fin y al cabo es lo que el usuario acaba presionando a la hora de escribir en el teclado.

Los teclados personalizados, aunque siempre ofrecen más opciones en cuanto a preferencias o más opciones en su diseño, también suele conllevar un mayor gasto y por eso es necesario tener en cuenta qué se quiere y como pueden cambiar los precios de estos.

### 1.2.1. Conectividad

En cuanto a conectividad, no hay tanta variedad, ya que esto se refiere a la forma de conectarse a un computador, y por el momento el grueso del mercado y de casi todos los teclados que se usan de forma comercial usan 4 tipos de conexión.

- Por Cable [42]

- [USB](#)

Esta forma de conectividad representa el grueso predominante en la actualidad, ya que prácticamente todos los dispositivos recurren a esta interfaz para establecer conexión con la computadora. Se trata de un estándar en constante evolución, actualizado mediante diversas versiones a lo largo del tiempo. ([USB 1.0](#), [USB 2.0](#), [USB 3.0](#), [USB 3.1](#), [USB 3.2](#) ...)

- [PS2](#)

La conexión [PS2](#) fue ampliamente utilizada en la década de 1990 y principios de la década de 2000 como un estándar para conectar periféricos a computadoras, especialmente dispositivos de entrada como teclados y ratones. Aunque ha sido superada en popularidad por el [USB](#), el [PS2](#) todavía se encuentra en algunos dispositivos más antiguos. Esta interfaz se caracteriza por su conector redondo con pines y ha experimentado varias revisiones a lo largo del tiempo, como el [PS2](#) estándar y el [PS2](#) Mini-DIN de 6 pines. A medida que la tecnología ha avanzado, el [PS2](#) ha quedado en gran medida relegado en favor de interfaces más modernas, pero su legado persiste en sistemas heredados y dispositivos retro.

- Por radiofrecuencia [15]

- [Bluetooth](#)

La conectividad por [bluetooth](#) ha ganado amplia aceptación en el ámbito inalámbrico, facilitando la comunicación entre dispositivos a corta distancia. Este estándar ha demostrado ser especialmente útil para la conexión de periféricos como auriculares, teclados y ratones de forma inalámbrica. Este también ha ido sufriendo actualizaciones que lo han mejorado en todos sus aspectos. Actualmente casi todos los teclados que son inalámbricos disponen de [bluetooth](#).

- [Dongle](#) receptor

El [Dongle](#) receptor, también conocido como adaptador, desempeña un papel crucial al habilitar la conectividad por radiofrecuencia en dispositivos que no cuentan nativamente con la capacidad [bluetooth](#). Al conectar este pequeño dispositivo, se amplía la gama de dispositivos compatibles y se permite la comunicación inalámbrica. Este se trata de una antena encapsulada en un conector que añadido al dispositivo y conectado a un [USB](#) hace de receptor, normalmente usa un protocolo que no es [bluetooth](#) y es más específico a la aplicación, por lo que hace que su [latencia](#) y alcance mejoren considerablemente.

**1.2.2. Formatos**

A lo largo de la evolución de los teclados, las configuraciones y disposiciones de las teclas han experimentado cambios significativos para adaptarse a las necesidades cambiantes de los usuarios y a los avances tecnológicos.

Inicialmente, los teclados adoptaron el diseño **QWERTY**, popularizado por las máquinas de escribir y posteriormente estandarizado por IBM. Este diseño se mantiene como el más común en la actualidad.

A lo largo de los años, surgieron alternativas, como el diseño Dvorak, que redistribuye las letras según su frecuencia de uso para aumentar la eficiencia de la escritura.

Para abordar preocupaciones ergonómicas, los teclados divididos surgieron con el objetivo de reducir la tensión al separar el teclado en dos secciones, ya sea físicamente o mediante un diseño ergonómico que coloca las secciones en ángulos más naturales para las manos.

En términos de tamaño y portabilidad, los teclados completos (100 %) ofrecen la disposición estándar, mientras que los teclados Tenkeyless (TKL) eliminan el teclado numérico para reducir el tamaño. Los teclados 75 % reducen aún más el tamaño al ajustar la disposición sin sacrificar funciones esenciales. Luego en el mundo de **DIY** existen variaciones extremas de esto, pudiendo encontrar teclados de 50 %, 45 % y hasta un 35 % que solo disponen de las teclas alfabéticas.

La elección de un formato de teclado se basa en las preferencias y necesidades individuales del usuario, considerando aspectos como la ergonomía, la portabilidad y el uso previsto, ya sea para trabajo, juegos u otros propósitos específicos. [21]

**1.2.3. Precios**

Cuando se trata de precios, la diversidad en el mercado de teclados refleja una amplia gama de opciones que se adaptan a diferentes presupuestos y necesidades. Desde opciones más asequibles hasta modelos de alta gama, los precios de los teclados varían según diversos factores.

Los teclados estándar con configuraciones tradicionales y cableado suelen ser más asequibles, brindando una opción económica para aquellos con presupuestos más ajustados. Estos teclados son ideales para usuarios que no necesitan características avanzadas o diseños especializados.

En el extremo superior del espectro, los teclados de gama alta ofrecen características avanzadas como retro-iluminación personalizable, interruptores mecánicos de alta calidad, y construcciones premium. Estos modelos suelen apuntar a entusiastas de los juegos, profesionales creativos o usuarios

que buscan una experiencia de escritura excepcional.

Además, la introducción de teclados inalámbricos, ergonómicos o compactos también afecta los precios. Los teclados ergonómicos diseñados para reducir la fatiga y mejorar la comodidad pueden tener un coste ligeramente superior, mientras que los modelos inalámbricos ofrecen la ventaja de la movilidad a un precio adicional.

En resumen, la amplia variedad de teclados disponibles en el mercado garantiza que haya opciones para todos los presupuestos. La clave está en identificar las características prioritarias y el uso previsto para encontrar el equilibrio perfecto entre funcionalidad y precio. [41]

#### **1.2.4. Otros teclados personalizados**

Siempre puede haber otra razón para buscar un teclado personalizado, entre ellas podemos encontrar motivos como diseños retro y [vintage](#). Teclados Temáticos o de Edición Limitada. También se hace si se quiere conseguir como objetivo principal lograr una funcionalidad extra que un teclado normal no va a proveer, o de forma extraordinaria una forma de presumir de las capacidades de fabricación y diseño del autor.

## Capítulo 2

# Especificación del Sistema

### 2.1. Requisitos

#### 2.1.1. Requisitos funcionales

##### Conectividad por cable

- **RF-1:** El teclado deberá poder conectarse mediante un cable **USB** para garantizar la compatibilidad con dispositivos sin capacidad **Bluetooth**.
- **RF-2:** Se deberá permitir la conexión y desconexión en caliente (**hot-plugging**) a través del cable **USB** sin afectar el rendimiento del teclado.

##### Conectividad Bluetooth

- **RF-3:** El teclado deberá ser capaz de establecer una conexión **Bluetooth** con dispositivos compatibles.
- **RF-4:** Deberá admitir el emparejamiento seguro con al menos un dispositivo a la vez.
- **RF-5:** Se permitirá la conexión **Bluetooth** mientras el dispositivo esté conectado por cable.
- **RF-6:** Se permitirá la opción de apagar completamente el **Bluetooth** del dispositivo.

##### Batería/Energía y Alimentación

- **RF-7:** El teclado contará con una batería recargable que proporcionará una autonomía mientras no esté conectado con cable.

- **RF-8:** Se deberá incorporar un sistema de carga eficiente que permita recargar la batería mientras el teclado está conectado por cable.
- **RF-9:** El teclado será capaz de funcionar mientras se carga para garantizar un uso continuo.
- **RNF-10:** El teclado entrará en modo de suspensión automáticamente después de un período de inactividad para conservar energía.

### Distribución de Teclas

- **RF-11:** El diseño del teclado seguirá la distribución ISO de 105 teclas para cumplir con los estándares internacionales.

### Compatibilidad y Configuración

- **RF-12:** El teclado será compatible con los principales sistemas operativos, como Windows y Linux.
- **RF-13:** Deberá ser posible configurar las funciones y asignaciones de las teclas mediante firmware, además la configuración del usuario se deberá guardar entre usos y apagados del teclado.

### 2.1.2. Requisitos no funcionales

#### Estética y Diseño

- **RNF-1:** El diseño del teclado deberá ser estéticamente agradable.
- **RNF-2:** La calidad de los materiales utilizados en la fabricación garantizará durabilidad y resistencia al desgaste.
- **RNF-3:** El producto será sencillo de montar y desmontar para su limpieza. El mantenimiento debe ser sencillo.

#### Seguridad

- **RNF-4:** El sistema de emparejamiento Bluetooth deberá seguir protocolos de seguridad estándar para evitar accesos no autorizados.

#### Rendimiento y Latencia

- **RNF-5:** El teclado garantizará un rendimiento sin demoras perceptibles, manteniendo una baja latencia durante la escritura y la conexión inalámbrica.

- **RNF-6:** La respuesta de las teclas será consistente y proporcionará una experiencia de escritura fluida, independientemente de la conexión utilizada ([Bluetooth](#) o por cable).

### Energía y Eficiencia

- **RNF-7:** Se implementarán medidas de ahorro de energía para optimizar la duración de la batería.

### Interfaz de Usuario

- **RNF-8:** La interfaz de usuario debe ser simple de entender y usar.

## 2.2. Especificaciones

### 2.2.1. Especificación Hardware

#### Conectividad por cable

- La conexión por cable se realizará a través de un conector XS12 de aviación por su robustez y estética para asegurar una conexión estable y un menor desgaste.
- El conector atornillado a la carcasa tendrá otro cable Micro JST de 4mm para poder separar la placa base de la carcasa de forma sencilla sin tener que desoldar nada.

#### Conectividad [Bluetooth](#)

- El teclado estará equipado con un módulo [Bluetooth](#) que permita garantizar una conexión estable.
- El dispositivo [Bluetooth](#), debe poder establecer una conexión de al menos de 5 metros.

#### Batería y Alimentación

- La batería recargable será de [ion de litio](#) de alta capacidad que se ubicara en la parte inferior de la placa base.
- La batería tendrá un conector para poder desconectarla de la placa base.

**Distribución de Teclas**

- Las teclas seguirán el estándar [ISO](#) con disposición [QWERTY](#) para adaptarse a los usuarios de habla hispana.
- Se utilizarán interruptores mecánicos de alta calidad para garantizar una respuesta táctil precisa y duradera. La elección del tipo será del usuario por gusto personal.

**2.2.2. Especificación Software****Configuración y Personalización**

- El [firmware](#) del teclado admitirá perfiles personalizados que podrán almacenarse en la memoria interna del dispositivo.
- El dispositivo permitirá una opción de crear programas completos que se podrán ejecutar en una tecla. Estos se programarán por [firmware](#).

**Compatibilidad con Sistemas Operativos**

- El teclado será compatible con [Windows](#) y [Linux](#), garantizando una experiencia uniforme en diferentes plataformas.
- En todo momento será [Plug-and-Play](#) para facilitar la instalación y uso sin necesidad de [controladores](#) adicionales.

**Actualizaciones de [firmware](#)**

- Se diseñará el [firmware](#) del teclado con la capacidad de recibir actualizaciones, permitiendo mejorar la funcionalidad y corregir posibles vulnerabilidades de seguridad.
- Las actualizaciones podrán aplicarse de manera sencilla mediante conector para conectar un [USB A TTL](#).

**Indicadores [LED](#)**

- Se incluirán 10 [LED](#)s para uso personalizado del usuario, tanto estético como funcional, para indicar estados de batería o conexión.
- Los indicadores [LED](#) serán configurables, permitiendo a los usuarios personalizar la apariencia y comportamiento de los mismos.

## 2.3. Planificación

Esta sección detalla el plan de desarrollo del teclado ([HID](#)) con conectividad [Bluetooth](#), batería y conexión por cable. La planificación aborda las fases clave del proyecto, asignación de recursos y plazos de entrega. Para ello se ha realizado un gráfico de Gantt para poder organizar las diferentes fases del proyecto en hitos a cumplir.

El gráfico de Gantt realizado podemos verlo en la figura 2.1 que señala de manera clara y concisa la planificación temporal del proyecto. Esta planificación se ha realizado para poder estimar mejor los tiempos de trabajo y qué aspectos serían necesarios tener para poder progresar. Esto nos ayudará a saber en qué punto del proyecto nos encontramos y qué aspectos debemos mejorar para poder cumplir con los plazos establecidos. Además, nos ayudará a saber qué recursos necesitamos, tanto para estimar el coste de este como para saber qué aspecto es necesario terminar antes para poder avanzar.

## 2.3. Planificación

## Especificación del Sistema

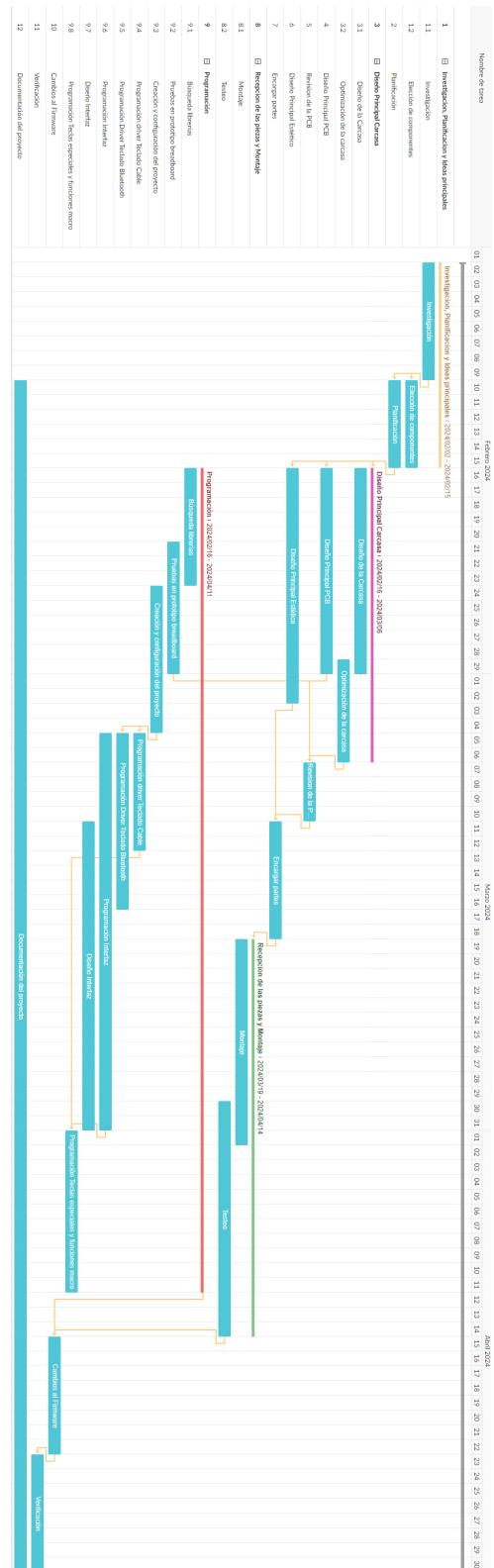


Figura 2.1: Planificación del proyecto con diagrama Gantt.

# Capítulo 3

## Diseño Y Prototipado

### 3.1. Teclados, entendiendo sus partes

En esta primera sección vamos a ver todas las partes de un teclado y que funciones tienen cada una de ellas. Así como interactúan entre sí y qué opciones hay. [43] La estructura general de un teclado es la que se muestra en la figura 3.1 que podemos, además, ver a continuación. Esta figura muestra las diferentes partes que un teclado podría tener. Aunque es cierto que aquí se muestran muchas más de las necesarias y de las que se usan normalmente. Más adelante se expondrán las partes que se van a usar en el teclado del proyecto.

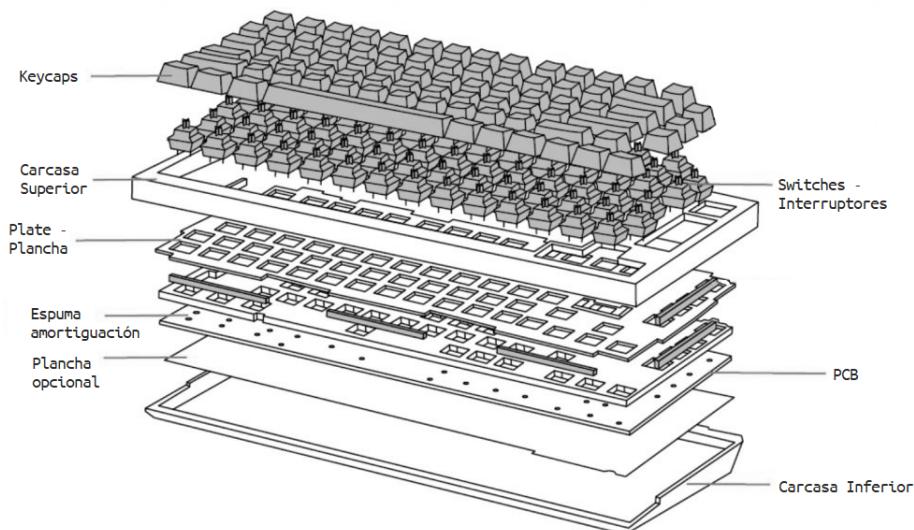


Figura 3.1: Estructura física de un teclado [43]

### Keycaps

Los **keycaps** son las tapas individuales que cubren las teclas en un teclado. Estas tapas suelen estar hechas de plástico y están diseñadas para permitir que los usuarios escriban y presionen las teclas de manera cómoda y precisa. Podemos ver un ejemplo en la figura 3.2. Los **keycaps** pueden variar en diseño, color, material y perfil. Algunos teclados personalizados incluso permiten a los usuarios intercambiar los **keycaps** para personalizar la apariencia o mejorar la sensación táctil del teclado. Algunos entusiastas de los teclados también coleccionan **keycaps** únicos y personalizados para crear conjuntos únicos y estéticamente atractivos. Para poder hacernos una idea de cómo son los **keycaps** se muestra la figura 3.2.

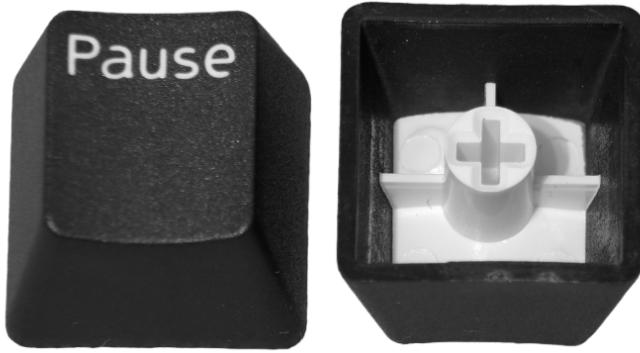


Figura 3.2: **Keycaps** [22]

### Switches o Interruptores

Los **switches** son los mecanismos que hay debajo de cada tecla. Permiten detectar cuándo se presiona una tecla y envían la señal al dispositivo electrónico al que está conectado el teclado. Estos **switches** pueden variar significativamente en términos de sensación táctil, fuerza de actuación y ruido producido al presionar la tecla.

- **Switches de membrana:** Estos son los más comunes y se utilizan en muchos teclados estándar. Consisten en una membrana de goma debajo de las teclas que se comprime cuando se presiona una tecla, cerrando un circuito eléctrico y enviando una señal al dispositivo.
- **Switches de tijera:** Estos **switches** utilizan una estructura de tijera debajo de las teclas para proporcionar una sensación de escritura más estable y una mejor respuesta táctil que los **switches** de membrana estándar.

- **Switches mecánicos:** Son switches individuales que utilizan un sistema mecánico para registrar la pulsación de una tecla. Hay varios tipos de switches mecánicos, incluyendo los populares switches Cherry MX, que vienen en variantes como los switches lineales, táctiles y clicky, que ofrecen diferentes sensaciones táctiles y niveles de ruido. Para hacernos una idea de cómo serían estos switches se muestra la figura 3.3.
- **Switches ópticos:** En lugar de utilizar contactos eléctricos, los switches ópticos utilizan luz infrarroja para detectar cuando se presiona una tecla. Ofrecen una mayor durabilidad y una respuesta más rápida en comparación con los switches mecánicos tradicionales.

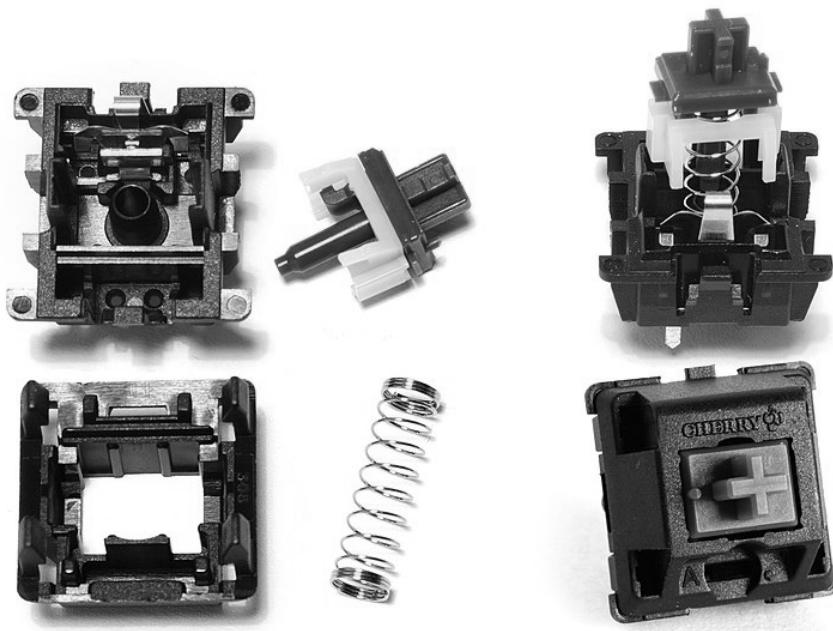


Figura 3.3: Switches o Interruptores mecánicos [24]

Los switches son un componente clave en la experiencia de escritura de un teclado y pueden afectar a la velocidad, la comodidad y la precisión al escribir. Los entusiastas de los teclados a menudo tienen preferencias personales sobre el tipo de switches que prefieren, y algunos incluso personalizan sus teclados con switches específicos para adaptarse a sus necesidades y preferencias individuales.

**Carcasa superior**

La carcasa superior de un teclado es la parte que cubre la parte de las teclas y proporciona la estructura externa. Esta carcasa puede variar en diseño y material según el tipo de teclado. En los teclados estándar, la carcasa superior suele estar hecha de plástico, mientras que en teclados de alta gama puede estar hecha de materiales más duraderos como aluminio o acero. La carcasa superior también puede incluir características adicionales, como reposamuñecas integrados o iluminación **LED**, dependiendo del diseño y la funcionalidad del teclado. En nuestro caso vamos a combinar esta junto con la inferior para formar una combinada que proporciona soporte y estructura sin que tenga que ser otra pieza nueva. Un ejemplo de una carcasa de plástico se puede ver en la figura 3.5 y la respectiva carcasa volteada en la figura 3.4.



Figura 3.4: Carcasa Superior [38]



Figura 3.5: Carcasa Superior Reverso [39]

### Plate o Plancha

El **plate**, también conocido como placa o plancha en español, es una pieza metálica o plástica que se coloca debajo de los **switches** en un teclado mecánico. Su principal función es proporcionar soporte estructural a los **switches** y distribuir uniformemente la fuerza ejercida al presionar las teclas sobre la superficie del teclado. Además, el **plate** también influye en la sensación táctil y el sonido de las pulsaciones de las teclas.

La elección del material y diseño del **plate** puede afectar la experiencia de escritura del usuario. Por ejemplo, los **plates** de acero ofrecen una mayor rigidez y pueden producir un sonido más nítido al escribir, mientras que los **plate** de aluminio pueden proporcionar una sensación más suave y amortiguada. Algunos teclados personalizados permiten a los usuarios elegir entre diferentes materiales y grosorres de **plate** para adaptarse a sus preferencias individuales.

- **Montaje en carcasa inferior:** Atornillada la **PCB** a la carcasa. Véase la figura 3.6.

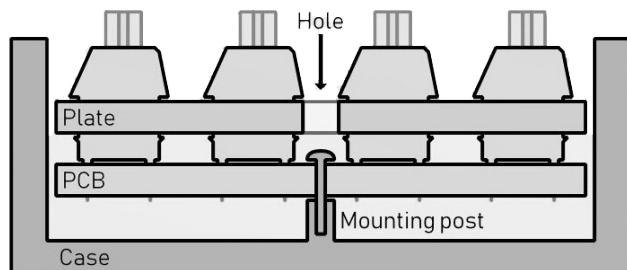


Figura 3.6: Montaje en carcasa inferior [20]

- **Montaje en carcasa superior:** Atornillada a la parte superior de la carcasa que luego se monta sobre la inferior. Véase la figura 3.7.

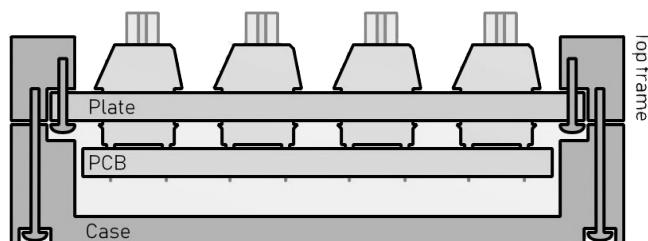


Figura 3.7: Montaje en carcasa superior [20]

- **Montaje en carcasa inferior escondida:** Atornillada a la carcasa inferior. Véase la figura 3.8.

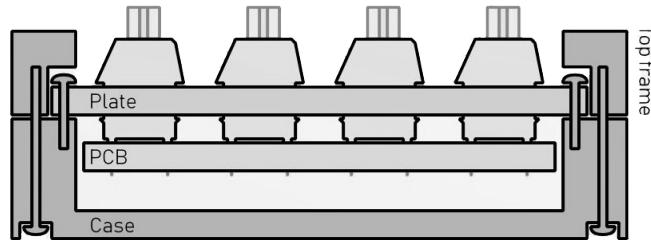


Figura 3.8: Montaje en carcasa [20]

- **Montaje en carcasa sandwich:** Atornillada siendo atravesada por la carcasa inferior desde la parte de abajo hasta la carcasa superior. Véase la figura 3.9.

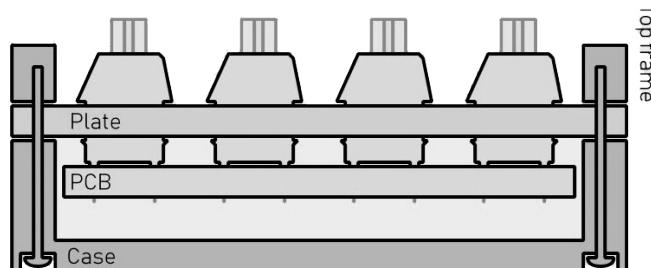


Figura 3.9: Montaje en carcasa sandwich [20]

- **Plate como carcasa superior:** La plate toma el rol de ser también la tapa o carcasa del teclado. Véase la figura 3.10.

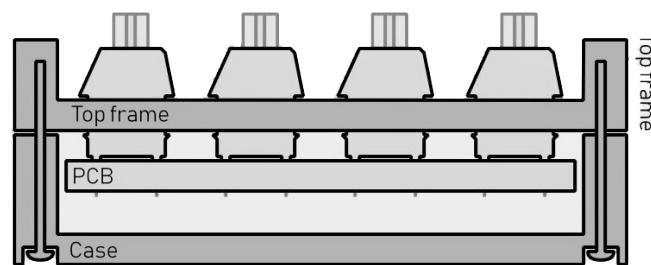


Figura 3.10: Montaje en forma de carcasa superior [20]

- **Montaje en empaquetado:** Al igual que el sandwich solo que esta queda bloqueada por unos pasadores además de ser atornillada. Véase la figura 3.11.

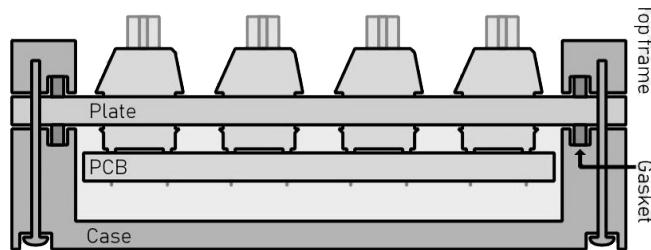


Figura 3.11: Montaje en empaquetado [20]

- **Montaje sin plate:** Este modo no usa **plate** a la hora de construir el teclado. Véase la figura 3.12.

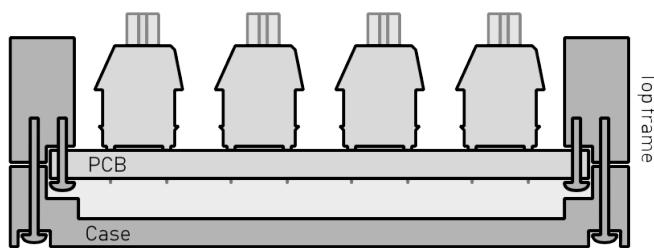


Figura 3.12: Montaje sin **plate** [20]

La opción sin **plate**, figura 3.12, es la opción que se ha escogido para nuestro teclado. Ya que facilitara el diseño, abaratara el presupuesto y simplificara el montaje.

Para hacernos una idea esta sería una **plate** convencional de un teclado 60% son sus tornillos correspondientes, tenemos la figura 3.13 con la que podemos ver cómo sería una **plate** de un teclado convencional del tamaño mencionado.



Figura 3.13: Plate [1]

## PCB

La **PCB**, o Placa de Circuito Impreso, es una parte fundamental de un teclado mecánico. Es una placa de material aislante, generalmente fibra de vidrio o resina epoxi, sobre la cual están montados los **switches** y otros componentes electrónicos del teclado. La **PCB** proporciona la estructura física para el ensamblaje de los componentes del teclado y permite la conexión eléctrica entre ellos.

Los circuitos impresos en la **PCB** dirigen las señales eléctricas de las teclas presionadas a través de los **switches** hacia el controlador del teclado, que luego las envía al dispositivo al que está conectado éste, como una computadora o una consola de juegos. Además, la **PCB** puede contener componentes adicionales, como diodos para la función anti-fantasma y **LEDs** para la retroiluminación.

La calidad y el diseño de la **PCB** pueden afectar a la durabilidad, la capacidad de personalización y la eficiencia energética del teclado. En teclados personalizados, la **PCB** a menudo es una parte importante del diseño y puede ser programable para permitir la personalización de las funciones de las teclas.

### Espuma

La espuma de amortiguación de sonido es un componente opcional que se puede agregar a un teclado mecánico para reducir el ruido producido por las pulsaciones de las teclas. Esta espuma se coloca generalmente entre la placa **PCB** y la carcasa inferior del teclado, ayudando a absorber las vibraciones y el ruido generado por las pulsaciones de las teclas.

Aunque la espuma de amortiguación de sonido puede mejorar la experiencia auditiva al usar el teclado, su inclusión es opcional y depende de las preferencias personales del usuario. Algunas personas prefieren el sonido más nítido y distintivo de un teclado sin espuma de amortiguación, mientras que otras prefieren un teclado más silencioso y amortiguado.

La calidad y el tipo de espuma utilizada pueden afectar la eficacia de la amortiguación de sonido. Algunos teclados personalizados vienen con espumas específicamente diseñadas para reducir el ruido, mientras que otros pueden permitir que los usuarios añadan su propia espuma según sus necesidades y preferencias.

### Carcasa Inferior

La carcasa inferior de un teclado es la parte que cubre la parte inferior del teclado y proporciona soporte estructural. Esta carcasa puede estar hecha de plástico u otros materiales resistentes y generalmente contiene la placa **PCB** y otros componentes internos del teclado. Esta proporciona una base sólida para los componentes internos y ayuda a protegerlos de daños externos. Si vemos la figura 3.14 podemos hacernos una idea de cómo sería una carcasa inferior de un teclado 60 % realizada en metal.

La carcasa inferior puede tener características adicionales, como patas ajustables para elevar la inclinación del teclado, o canales de enrutamiento para cables, que mejoran la comodidad y la organización al usar el teclado.



Figura 3.14: Carcasa Inferior de Metal de teclado 60 %

### 3.2. Investigación: Herramientas y Tecnologías

La investigación y elección de herramientas y tecnologías son elementos fundamentales en el desarrollo de cualquier proyecto. En esta etapa, nos centraremos en identificar y evaluar diversas opciones disponibles en el panorama tecnológico actual, prestando especial atención a criterios clave como accesibilidad, robustez, comunidad de soporte y facilidad de integración. Abordaremos las necesidades específicas de cada fase del proyecto, desde el desarrollo de hardware hasta el software, con el objetivo de seleccionar herramientas y tecnologías que maximicen la eficiencia y la sinergia entre los componentes del sistema.

Aunque a lo largo de esta etapa se han llegado a encontrar alternativas mejores, estas supondrían un coste/beneficio demasiado alto para su desarrollo en el cuadro de tiempo dado. Ya que estas suponen una curva de aprendizaje o coste de compra/uso demasiado alto para ser abordables. Aunque estas se mencionarán en cada sección y aparecerán descartadas por falta de tiempo o recursos. Aun así para alguien con el tiempo suficiente y la preparación necesaria serían sin duda la mejor elección.

En todas las secciones próximas se expondrán las elecciones realizadas en cada ámbito. Y también todas y cada una de las alternativas que se hayan llegado a considerar junto a la explicación de porque han sido descartadas. Finalmente, se mostrará una tabla con los pros y contras para que cada uno pueda tomar sus decisiones a cerca de estas elecciones.

### 3.3. Alternativas de diseño

Aquí vamos a contemplar las diferentes opciones que tenemos a la hora de hacer un teclado, ya que como se ha mencionado antes, hay muchos formatos y tipos. Personalmente, sé cuál va a ser el diseño elegido, ya que ando buscando un tipo de teclado que no he podido llegar a encontrar en decenas de páginas [online](#) que he visitado y que cumplen las características siguientes:

- **ISO**

La distribución será la [ISO](#), ya que mi idioma principal es el castellano.

- **100 % o 105**

El teclado quiero que tenga todas las teclas que puede tener un teclado convencional, incluyendo el teclado numérico y teclas de función. En este caso, al ser [ISO](#) la cantidad de teclas para llegar al 100 % son 105.

- **Bluetooth**

Conectividad inalámbrica.

- Conectividad por cable  
Para poder usar el teclado en un modo de baja [latencia](#).
- Interruptores Mecánicos.  
Para asegurar todo tipo de sensaciones a la hora de escribir. En mi caso Interruptores lineales.

### 3.3.1. Elección de las herramientas

Vamos a empezar por las herramientas de diseño, la primera de todas la que hemos usado para crear los esquemas y planos del teclado, tomar medidas y planificar todo el diseño.

- Planos  
Las herramientas que se han encontrado para la creación de los planos son básicamente dos, una de pago y otra opción de software libre. Estas son AutoCAD y QCad. Dado que poseo una licencia de AutoCAD y personalmente tengo experiencia en esta herramienta he decidido usar ésta para la creación de los planos del teclado. Para así poder ubicar todos los elementos de forma sencilla y sus dimensiones.
- Diseño de [PCB](#)  
Existen varias opciones disponibles en el mercado para el diseño de [PCB](#). Algunas de las más populares son Altium Designer, KiCad, Eagle. Todos ellos son bastante parecidos en cuanto a las funcionalidades que se van a usar para el proyecto. Como se ha mencionado con las herramientas para los planos, también tenemos de pago y de software libre. Realmente aquí se ha escogido la herramienta por su conocimiento previo gracias a materias del grado como Diseño de Circuitos Impresos, donde se trabajó con Eagle. Por lo que el teclado está realizado con la herramienta Eagle.
- Editor de Texto/Programación  
Aquí disponemos de muchas opciones, entre las más famosas tenemos VSCode, Vim, [Arduino](#) Editor (Si se usa [Arduino](#)) o Sublime. Por conveniencia y uso a lo largo del tiempo, se va a usar VSCode, ya que se dispone del framework [PlatformIO](#), que es el que se va a utilizar a posteriori para programar nuestro controlador y diseñar el programa/[firmware](#).
- Software 3D  
Aquí tenemos las opciones más sencillas de diseño 3D, FreeCAD y Fusion360. Como ambas permiten las mismas opciones, se ha elegido FreeCAD como software de edición 3D.

- **CNC Corte**

En la búsqueda de software para crear los archivos necesarios para hacer funcionar la glsCNC para así hacer la carcasa se han encontrado diversos programas. Aunque la elección es clara, FreeCAD, ya que tenemos los archivos 3D ya en la aplicación y no tenemos que hacer nada más.

### 3.4. Elección de Hardware

En esta sección se van a tratar las diferentes opciones para los materiales. Los componentes electrónicos, las diferentes partes del teclado y el porqué de estas.

#### 3.4.1. Layout

Lo primero que se necesitó es conocer las dimensiones de la distribución. Así que buscando entre varias páginas encontré una herramienta que se llama “Keyboard-Layout-Editor“ [19], que te permite personalizar la distribución directamente. La única modificación que sufrió la distribución ISO fue que bajé las teclas especiales de la sección Print-Screen al bloque de Pause/Delete. Esto fue pensado a futuro para poder colocar la electrónica en un sitio accesible. Esta modificación se puede ver en la figura 3.16, tenemos de referencia la distribución ISO en la figura 3.15 que sería la distribución original del teclado.

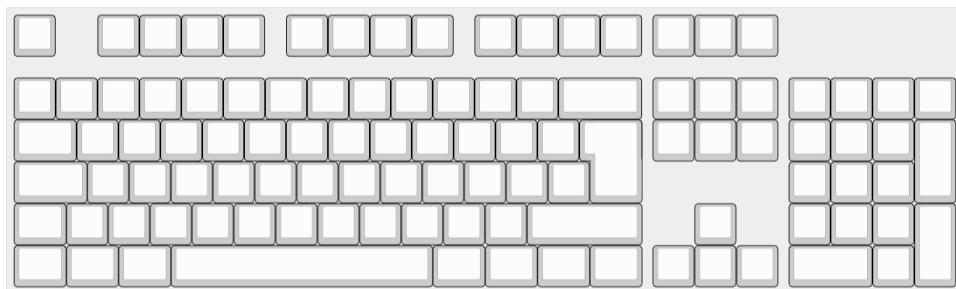


Figura 3.15: ISO 105 sin modificar

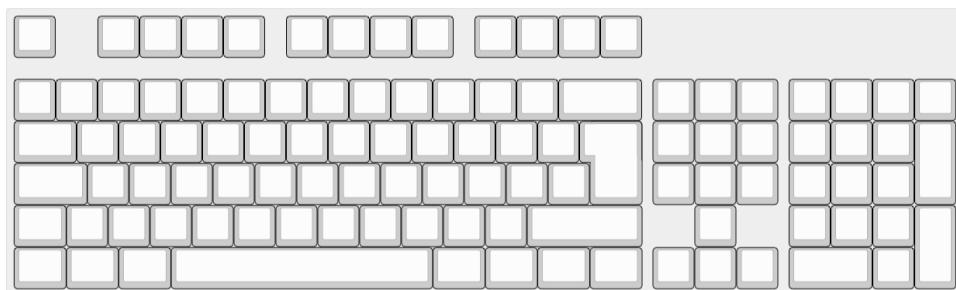


Figura 3.16: ISO 105 Modificado para el teclado ModernWood

Esta herramienta permite exportar el esquema del teclado en un .JSON que posteriormente será necesario para usarlo en la página de “Plate & Case

Builder“ [40]. Esta herramienta nos da un archivo .DWG que es un formato de AutoCAD para planos. El plano que nos ofrece es la plancha del teclado. Aunque nuestro teclado no la va a utilizar, esta nos servirá posteriormente para diseñar la [PCB](#) y la carcasa, ya que nos dirá donde va cada interruptor de forma muy precisa.

### 3.4.2. Componentes Electrónicos

A lo largo de toda la investigación he ido encontrando muchas alternativas para realizar el teclado, diferentes sistemas, diferentes microcontroladores y diferentes tipos de circuitos.

#### Microcontrolador

Este es el cerebro del teclado, va a ser el encargado de la comunicación, de registrar las teclas, de construir los paquetes de datos, de mostrar los datos correspondientes y de mantener toda la iluminación.

Por tanto es la pieza sobre lo que gira todo el proyecto y la parte más importante, también hay que considerar la facilidad para programarlos y la cantidad de recursos que tiene, tanto de librerías como de documentación. Cuanto más extendido sea el uso, más sencillo será lograr que funcione.

A lo largo del periodo de investigación y diseño se han encontrado varias alternativas. Estas tienen que cumplir que tenga un hardware dedicado a la conectividad [USB](#) llamado “[USB OTG](#)” ya que se ideó así en la sección 2.1.1.

Los principales y más destacables son:

- **Atmega32u4:** Es un microcontrolador de la familia AVR de Atmel. Es ampliamente utilizado en proyectos de electrónica, especialmente en entornos [Arduino](#). Ofrece una buena cantidad de recursos y es fácil de programar.
- **STM32F103C8T6:** Es un microcontrolador de la familia STM32 de STMicroelectronics. Ofrece un buen equilibrio entre potencia de procesamiento y recursos. Es popular en proyectos de electrónica debido a su amplia disponibilidad y comunidad activa de usuarios.
- **nrf52840:** Es un microcontrolador de Nordic Semiconductor, diseñado específicamente para aplicaciones de conectividad inalámbrica de baja energía. Ofrece capacidades avanzadas de [Bluetooth](#) y un bajo consumo de energía, lo que lo hace adecuado para dispositivos portátiles y [wearables](#).
- **ESP32S3:** Es un microcontrolador de Espressif Systems, conocido por su conectividad [WiFi](#) y [Bluetooth](#) integrada. Ofrece una potencia de procesamiento considerable y una amplia gama de recursos.

Cada uno de estos microcontroladores ofrece pros y contras al proyecto.

**Atmega32u4**

<b>Pros</b>	<b>Contras</b>
<ul style="list-style-type: none"> <li>- Ampliamente utilizado y documentado en proyectos de teclado y dispositivos de entrada.</li> <li>- Facilidad de programación con el ecosistema <a href="#">Arduino</a>.</li> <li>- Integración de puertos <a href="#">USB</a> nativos, lo que facilita la comunicación.</li> <li>- Fácil de soldar</li> <li>- Librerías para <a href="#">PCB</a></li> <li>- Muy barato</li> <li>- 5V</li> </ul>	<ul style="list-style-type: none"> <li>- Requiere hardware adicional y configuración personalizada para habilitar la conectividad <a href="#">Bluetooth</a>.</li> <li>- Limitaciones en términos de potencia de procesamiento y recursos en comparación con otros microcontroladores más orientados a la conectividad.</li> </ul>

Tabla 3.1: Ventajas y desventajas de Atmega32u4

**STM32F103C8T6**

<b>Pros</b>	<b>Contras</b>
<ul style="list-style-type: none"> <li>- Soporte para una amplia variedad de librerías y documentación, incluidas las relacionadas con <a href="#">Bluetooth</a>.</li> <li>- Potente y versátil, lo que facilita la implementación de soluciones <a href="#">Bluetooth</a> personalizadas.</li> <li>- Ampliamente utilizado en aplicaciones de <a href="#">IoT</a> y sistemas embebidos, lo que puede ofrecer soluciones ya probadas para la conectividad <a href="#">Bluetooth</a>.</li> <li>- Librerías para <a href="#">PCB</a></li> <li>- Fácil de soldar</li> <li>- Barato</li> </ul>	<ul style="list-style-type: none"> <li>- Requiere herramientas de desarrollo específicas y conocimientos más avanzados en programación.</li> <li>- Curva de aprendizaje más pronunciada debido a su complejidad en comparación con los microcontroladores basados en <a href="#">Arduino</a>.</li> <li>- Posiblemente excesivamente difícil para aplicaciones simples de teclado.</li> <li>- Requiere hardware adicional y configuración personalizada para habilitar la conectividad <a href="#">Bluetooth</a>.</li> <li>- Requiere el uso de su framework específico</li> <li>- 3.3V</li> </ul>

Tabla 3.2: Ventajas y desventajas de STM32F103C8T6

**Nrf52840**

Pros	Contras
<ul style="list-style-type: none"> <li>- Diseñado específicamente para aplicaciones de baja potencia y conectividad inalámbrica, lo que lo hace ideal para la implementación de <b>Bluetooth</b> de baja energía.</li> <li>- Amplia gama de características de conectividad, incluido <b>Bluetooth</b> de baja energía.</li> <li>- Buen soporte de documentación y comunidad de desarrollo.</li> <li>- Modo <b>deep sleep</b> para ahorrar energía</li> </ul>	<ul style="list-style-type: none"> <li>- Sin librerías para <b>PCB</b> Eagle</li> <li>- Muy difícil de soldar</li> <li>- Menos común en aplicaciones de teclado en comparación con otros microcontroladores.</li> <li>- Requiere más experiencia en el desarrollo de sistemas inalámbricos y <b>Bluetooth</b>.</li> <li>- Requiere el uso de su framework específico.</li> <li>- Caro.</li> <li>- 3.3V</li> </ul>

Tabla 3.3: Ventajas y desventajas de Nrf52840

**ESP32S3**

Pros	Contras
<ul style="list-style-type: none"> <li>- Capacidades de conectividad <b>WiFi</b> y <b>Bluetooth</b> integradas, lo que facilita la implementación de soluciones <b>Bluetooth</b>.</li> <li>- Potente y versátil, con una amplia comunidad de desarrollo y soporte de documentación.</li> <li>- Ideal para proyectos que requieren conectividad inalámbrica y capacidades avanzadas de procesamiento.</li> <li>- Con todo tipo de librerías</li> <li>- Modo <b>deep sleep</b> para ahorrar energía</li> <li>- Muy barato</li> </ul>	<ul style="list-style-type: none"> <li>- Dificultad media para soldar.</li> <li>- Consumo alto de energía sin <b>deep sleep</b>.</li> <li>- 3.3V</li> </ul>

Tabla 3.4: Ventajas y desventajas de ESP32S3

La elección final se va a volver a hacer en cuanto a estética y funcionalidad. Cómo buscar nuevo hardware para la conectividad [bluetooth](#) iba a ser otra tarea costosa en cuanto a tiempo y recursos, los microcontroladores que necesitasen un hardware específico para la conectividad han sido descartados.

Lo que nos deja solo con ESP32S3 y Nrf52840. Entre estos dos, dada la dificultad de aprendizaje, la dificultad muy alta para soldarlo a la [PCB](#) y que hay menos documentación, nos vamos a quedar con el microcontrolador **ESP32S3**. Una vez que ya sabemos qué microcontrolador vamos a usar podemos atender a las diferentes secciones.

## Sistema de Batería

Como se ideó en la sección de diseño 2.2.1 y dado que el hueco para albergar la batería es pequeño y bastante plano, ya que tiene que ir entre la placa **PCB** y la carcasa no hay muchas opciones. Esta será una batería Li-ion (Ion Litio) y dado que el microcontrolador funciona a 3.3V tendremos que buscar una batería típica de 3.7V.

Ahora el problema subyacente es cargar la batería y al mismo tiempo poder usar el teclado sin que esta se dañe. Buscando alternativas y modos de realizar esta tarea se ha encontrado un módulo para desarrollo llamado TP4056 que contiene un sistema de protección de la batería y un sistema de carga y consumo en paralelo manteniendo la vida de la batería, podemos ver este circuito en la figura 3.17. Como se quiere aun así mantener la estética, no se empleara el módulo directamente, sino que se buscara un esquemático del módulo y se implementará directamente sobre la [PCB](#). Se han encontrado varios archivos esquemáticos.

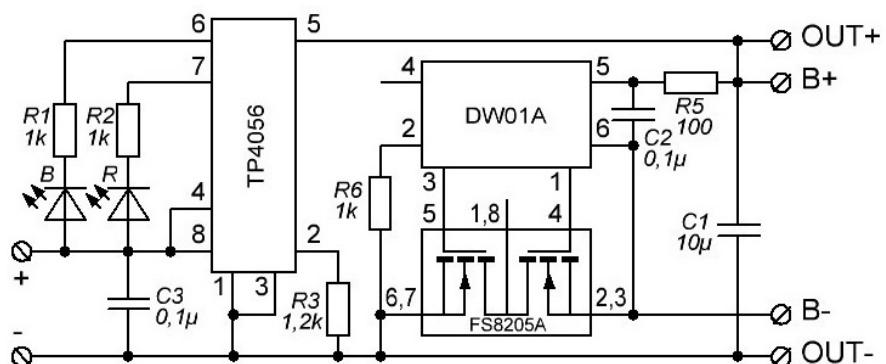


Figura 3.17: Esquemático del módulo TP4056 [23]

### Sistema de Pantalla

La idea es que este teclado dispusiera de una pantalla para mostrar información básica del estado del teclado, configuración y un menú de usuario. La búsqueda de una pantalla tampoco ha supuesto un gran reto. Solo tenía que cumplir varios requisitos. Que cupiese en el área designada para la electrónica y que fuera compatible con el microcontrolador. Hay dos pantallas que cupiesen en el área designada y no fueran unan **LCD** de poca resolución, estas son la **TFT 0.96"** y la **OLED 0.96"**.

#### **TFT 0.96"**

Pros	Contras
<ul style="list-style-type: none"> <li>- Pantalla a color que permite una representación más detallada.</li> <li>- Fácil de programar y utilizar con librerías disponibles.</li> <li>- Ideal para proyectos que requieren interfaces de usuario simples.</li> <li>- Amplia disponibilidad y variedad de opciones en el mercado.</li> </ul>	<ul style="list-style-type: none"> <li>- Consumo moderado de energía, pero más alto que las pantallas <b>OLED</b>.</li> </ul>

Tabla 3.5: Ventajas y desventajas de la pantalla TFT 0.96"

#### **OLED 0.96"**

Pros	Contras
<ul style="list-style-type: none"> <li>- Pantalla que ofrece un contraste superior a un solo color.</li> <li>- Consumo de energía muy bajo, especialmente al mostrar contenido estático.</li> <li>- Ángulos de visión más amplios y mejor visibilidad en condiciones de luz ambiental variable.</li> <li>- Ideal para aplicaciones de bajo consumo.</li> </ul>	<ul style="list-style-type: none"> <li>- Solo un color.</li> </ul>

Tabla 3.6: Ventajas y desventajas de la pantalla OLED 0.96"

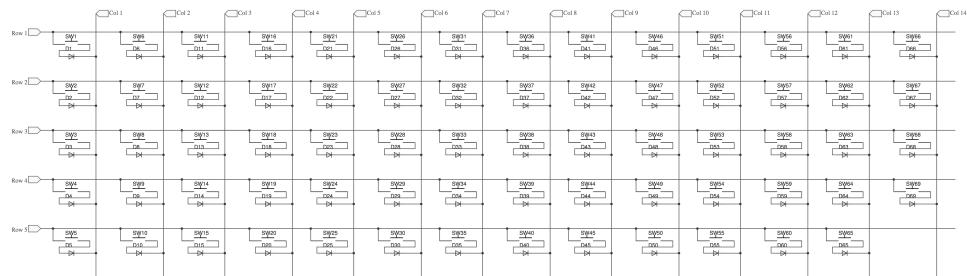
Se ha acabado escogiendo la pantalla **TFT 0.96"** solo por la cantidad de colores que puede mostrar. Esta usa 8 pines de nuestro microcontrolador.

### Sistema de polling

Dado que los microcontroladores no tienen la cantidad de pines necesarios para conectar la matriz de pines de 16x6, en total son necesarios 22 pines exclusivamente para todas las teclas. Revisando materias de electrónica, recordé la existencia de un dispositivo que permite multiplexar señales eléctricas. Dado que tenemos que ir recorriendo las secciones verticales una a una e ir leyendo las horizontales para saber qué tecla se ha pulsado, podemos multiplexar esas 16 líneas en tan solo 4 gracias al dispositivo **multiplexor**. Para hacernos una idea de como es la matriz de teclado podemos ver la figura 3.18.

El objetivo es encontrar un **multiplexor** que quepa y que nos ofrezca la función de 4 a 16, que valga a 3.3V y que pueda ser soldado con facilidad. Tan solo con una búsqueda aparece el **CD74HC154M96**, dado que directamente cumple todos los requisitos pedidos nos vamos a quedar con este.

Esto nos permitirá configurar el microcontrolador con 4 pines de salida para seleccionar la columna que queremos comprobar y leer los 6 pines configurados como entrada para saber qué tecla exacta está o no pulsada.



### Conexión USB

Como estamos usando un microcontrolador que dispone de [USB-OTG](#) no será necesario usar un elemento hardware externo que nos permita comunicarnos con el protocolo [USB](#). Este ya vendrá por defecto en el ESP32S3.

Eso sí, necesitaremos mirar qué pines están conectados al módulo de [USB](#) en el ESP32S3. Para la conexión al exterior usaremos simplemente un cable conectando la [PCB](#) al conector de aviación XS-12 que nos asegure la robustez que se ideó en la sección 2.1.2. El conector XS-12 se puede ver abajo en la figura 3.19.



Figura 3.19: Imagen del conector XS-12

Este conector irá atornillando a la carcasa y luego se fabricará un cable de este conector a [USB](#) tipo A. Usando un cable reforzado de 4 hilos internos y dos conectores más. Un conector de seguridad GX16 de 4 pines para darle robustez y un acabado más estético y un cabezal [USB](#) A chapado en oro y todo sellado con tubo [termoretráctil](#). El conector GX16 podemos verlo en la figura 3.20 y así hacernos una idea de cómo será el conector [USB](#) y el tipo A en la figura 3.21.



Figura 3.20: Imagen del conector GX-16



Figura 3.21: Imagen del conector [USB](#) tipo A

### Sistema de [LEDS](#)

Se han elegido los [LEDS](#) programables que se usan en las tiras comunes, ya que permiten muchas combinaciones y son muy baratos y sencillos de programar, además de que solo usan un pin del microcontrolador. La elección más sencilla ha sido para **WS2812B-B/W** que es el modelo más común y más barato. Podemos ver una imagen de este [LED](#) en la figura 3.22.



Figura 3.22: Imagen del [LED](#) WS2812B

### 3.4.3. Materiales

Desde un principio la elección clara era madera, ya que se quería conseguir una estética natural y tecnológica juntas. Las maderas serán a elección del usuario, para que pueda escoger tanto color, tipo de veta y sensación al tacto. Por lo tanto esta subsección se tratará de gustos personales. Entre las elecciones para este proyecto se han considerado **Jatoba, Sucupira y Mansonia**, además se añadirán detalles en metacrilato o acrílico para proteger los componentes que son visibles, ya que el teclado está diseñado para ser un teclado sin [plate](#).

### 3.4.4. Partes del teclado

Durante la fase de diseño 2.2.1 se han hecho elecciones de cómo va a ser el teclado, y de qué cosas dispondrá este. En esta subsección se hará un breve resumen. El teclado se compondrá de una carcasa en forma de cajón donde la [PCB](#) irá atornillada y no tendrá [plate](#). Los interruptores se colocarán soldados sobre la [PCB](#) con las [keycaps](#). De forma adicional se colocarán unas placas de acrílico sobre los componentes electrónicos para protegerlos y que aun así queden vistos. La tornillería se compondrá de elementos como separadores y de tornillos M3 negros.

La electrónica se compondrá de un ESP32S3 un [multiplexor](#) para el mapeo de las teclas, una pantalla [TFT](#) de 0.96”, una batería, un circuito específico para la batería (TP4056) y todo el cableado [USB](#).

## 3.5. Elección de Software

En esta sección se van a tratar las diferentes opciones para las elecciones que tenemos de herramientas para desarrollar el [firmware](#) para el ESP32S3 y qué librerías se van a usar para ello.

### 3.5.1. Entorno de desarrollo

Dado que vamos a trabajar con un ESP32S3 tenemos múltiples opciones, **Arduino IDE**, **PlatformIO** y **Espressif ESP-IDF**.

Estas alternativas están ordenadas de menor a mayor complejidad y funcionalidad.

#### **Arduino IDE**

[Arduino](#) IDE se compone un programa que permite la edición de un archivo .ino que posteriormente se compila y se sube al microcontrolador. Dado que es muy sencillo este entorno, está muy muy limitado para proyectos con varios archivos, librerías personalizadas y código C. Esta queda directamente descartada, ya que nuestro proyecto estará si o si dividido en varios archivos y será mucho más sencillo en las otras.

#### **PlatformIO**

[PlatformIO](#) es un entorno de desarrollo que reúne las herramientas necesarias para el desarrollo de proyectos de hardware, especialmente en microcontroladores como Atmega, ESP32, STM32, entre otros. Ofrece una solución más avanzada y versátil en comparación con el [Arduino](#) IDE.

Permite la creación y gestión de proyectos complejos que involucran múltiples archivos, bibliotecas personalizadas y configuraciones específicas para diferentes placas y microcontroladores.

Facilita la inclusión y gestión de bibliotecas externas a través de su sistema de gestión de librerías integrado. También ofrece herramientas avanzadas de compilación y depuración que facilitan el desarrollo y la corrección de errores en el código.

Admite una amplia gama de microcontroladores y placas, lo que permite a los desarrolladores trabajar con diferentes dispositivos sin restricciones. Entre ellas el microcontrolador ESP32S3.

Además, [PlatformIO](#) se integra con entornos de desarrollo integrado (IDE) como Visual Studio Code, Atom y Eclipse, lo que proporciona a los desarrolladores una experiencia de desarrollo más completa y personalizable.

### Espressif ESP-IDF

El Espressif [IoT](#) Development Framework (ESP-IDF) es un conjunto de herramientas de desarrollo de software diseñado específicamente para trabajar con microcontroladores de la serie ESP32 de Espressif Systems. Este framework proporciona un entorno de desarrollo completo y robusto para la creación de aplicaciones y [firmware](#) personalizado para dispositivos basados en ESP32.

El ESP-IDF está diseñado para aprovechar al máximo las capacidades y características del ESP32, incluyendo su procesador de doble núcleo, conectividad [WiFi](#) y [bluetooth](#), y una amplia gama de periféricos integrados.

Viene con una amplia gama de bibliotecas y ejemplos de código que facilitan el desarrollo de aplicaciones para una variedad de propósitos, como la comunicación inalámbrica, la gestión de energía y la interfaz con sensores y actuadores.

Espressif proporciona una documentación exhaustiva y actualizada que cubre todos los aspectos del desarrollo con ESP-IDF, incluyendo guías de inicio rápido, tutoriales y referencias de [API](#).

En resumen, el Espressif ESP-IDF es un framework de desarrollo de software potente y versátil que ofrece a los desarrolladores todas las herramientas necesarias para crear aplicaciones personalizadas para dispositivos basados en ESP32, aprovechando al máximo las capacidades de estos microcontroladores. Pero dado que este es mucho más amplio, también es más complejo de usar.

### Elección final

Dado todo lo anterior y que nuestro proyecto no va a requerir un código muy complejo y un control de muy bajo nivel sobre el microcontrolador, vamos a elegir [PlatformIO](#) que nos permitirá tener un proyecto grande, compuesto de múltiples archivos que nos dará la suficiente libertad sin tener que invertir muchos recursos aprendiendo a usar ESP-IDF. Aun así, instalaremos ESP-IDF para compilar el código, ya que el compilador específico genera un mejor código.

### 3.5.2. Librerías

En el apartado Hardware se han listado varios componentes que van a necesitar comunicarse con el Microcontrolador ESP32S3, entre ellos tenemos los **LEDS**, la pantalla **TFT 0.96"**. También será necesario activar y poder usar el **bluetooth** como elemento **HID**. Por lo que haciendo el recuento tenemos que buscar estas 3 librerías que nos permitan hacer uso de estos dispositivos.

- **LEDS**

Para los **leds** con una simple búsqueda podemos encontrar que Adafruit (El fabricante de la pantalla) ya pone a disposición una librería compatible con nuestro Framework. Esta librería es la **Adafruit Neo-Pixel**.

- **Pantalla TFT 0.96"**

Al igual que para los **leds** disponemos de una librería que nos permite hacer todo tipo de cosas con la pantalla, mostrar gráficos, imágenes y texto de forma sencilla. Esta está realizada por un particular en este caso. La librería se llama **TFT\_eSPI [2]**.

- **Bluetooth HID**

Aquí volvemos a encontrar que disponemos de una librería particular que nos resuelve el problema de mostrar nuestro teclado como un dispositivo **HID** a otros dispositivos a través de **bluetooth**, este también nos permite enviar caracteres al dispositivo conectado, por lo que nos vendrá genial. La librería en este caso es la **NimBLE-Arduino [18]**.

## 3.6. Presupuesto

Esta sección presenta un listado detallado de los costos asociados al desarrollo y fabricación del producto. El presupuesto abarca diferentes aspectos del proyecto, desde la adquisición de materiales hasta los gastos relacionados con el personal y la investigación y desarrollo. También se proporcionará una lista de todos los componentes con su coste asociado y el precio de adquisición así como el lugar de donde se ha adquirido y el coste de envío.

### Componentes Electrónicos

En la tabla 3.7 se puede ver el coste de todos estos componentes, estos han sido elegidos en base al precio, funcionalidad y tipo.

El total de la suma de los componentes electrónicos supone: **67,76 €**

Tipo	Elemento	Cantidad	Proveedores	Precio
Multiplexor	CD74HC154M96 (SOIC-24-300mil)	1 (5)	lcsc.com	3.40
Condensador	293D105X9035A2TE3 1uF (1206)	5 (20)	lcsc.com	2.06
Resistencia	RT0805BRD071KL 1k (0805)	5 (20)	lcsc.com	0.68
Resistencia	RC0805FR-071ML 1M (0805)	3 (100)	lcsc.com	0.21
Resistencia	0805W8F2004T5E 2M (0805)	2 (100)	lcsc.com	0.16
Resistencia	RC0805FR-073M74L 3.7M (0805)	1 (50)	lcsc.com	0.18
Circuito Carga Batería	TP4056 (C725790)	1 (5)	lcsc.com	0.52
C. Protección Batería	DW01A (SOT23-6)	1 (20)	lcsc.com	0.41
C. Protección Batería	ME6211C33M5G (SOT-23-5)	1 (10)	lcsc.com	0.48
C. Protección Batería	FS8205A (TSSOP-8)	1 (10)	lcsc.com	0.62
Chip Principal	ESP32-S3 (SMD,18x25.5mm)	1 (1)	lcsc.com	4.46
Leds	WS2812B (C114586)	10 (25)	lcsc.com	2.18
Diodos	1N5819W (SOD123FL)	96 (250)	lcsc.com	2.08
PCB	Placa Base DIY	1 (5)	jlpcb.com	42.76
Gasto Envíos	Envio Piezas	1 (1)	lcsc.com	7.56
Total				67.76

Tabla 3.7: Componentes electrónicos

### Componentes de Montaje

En la tabla 3.8 se puede ver el coste de todos estas piezas y partes, estos han sido elegidos en base al precio, estética y funcionalidad.

El total de la suma de los componentes para montaje supone: **185,83 €**

Tipo	Elemento	Cantidad	Proveedores	Precio
Partes Cable	Paracord negro de 9 núcleos	1 (1)	Aliexpress Qiuhiike	3.16
Partes Cable	Cabeza <b>USB A</b> Chapada en Oro	1 (5)	Aliexpress Lingxun	4.54
Tornillería	Espaciadores hexagonales de latón M3x3,M3x4, M3x5	M3x3 25 (30) M3x4 25 (30) M3x5 25 (30)	Aliexpress HUJI	10.77
Tornillería	Tuerca hexagonal para inserción muebles	25 (30)	Aliexpress Electrician	3.05
Estética	Tubo de gel de difusor led negro	1 (1)	Aliexpress ANTVLED	6.92
Electronica	Pantalla <b>TFT LCD</b> 0.96" PCB Negra	1 (1)	Aliexpress All-goods	5.06
Conectores Electronica	Micro conector JST de 4 pines	2 (10)	Aliexpress JSAAHZ	5.48
Conectores Electrónica	Conecotor XS12 Aviación de 12mm	1 (1)	Aliexpress MannHwa	2.24
Batería	Batería LiPo 3.7V 706090 5000mha	1 (1)	Aliexpress EasyLander	12.13
Partes Cable Conector	Conecotor Gx16-4 Pin	1 (1)	Aliexpress Wire-conn	1.35
Estabilizadores	Everglide-tornillo PCB Estabilizadores teclado	1 Pack (1)	Aliexpress KRepublic	18.90
Interruptores	Gateron- 5 Pines	105 (110)	Aliexpress Lesozoh	30.41
<b>Keycaps</b>	JakeTsai, doble capa MX ISO ANSI, SA ABS	1 Pack (1)	Amazon JakeTsai	50.00
Carcasa	Tabla de Madera elección Sucupira/Jatoba/Mansonia	1 (1)	Contraveta	31.82
Total				185.83

Tabla 3.8: Componentes para montaje

### Servicios y Herramientas

A parte de todo esto, que son las piezas necesarias para montarlo y que compondrán un teclado, serán necesario muchas más cosas, entre ellas las herramientas necesarias para poder montarlo. También se tendrá en cuenta los servicios contratados, entre ellos el de **fresado** para la carcasa y a su vez las horas dedicadas al proyecto. Estos servicios y herramientas se pueden ver en la tabla 3.9.

El total de la suma de los servicios y herramientas supone: **23.049,3 €**

Tipo	Elemento	Cantidad	Proveedores	Precio
Servicios	Fresado Carcasa	1 (1)	CNC GRANADA	51.43
Servicios	Costes Ingeniero	1 (1)	CLM	22900.00
Herramientas	Soldador Estaño	1 (1)	Aliexpress Preferred Store	9.95
Herramientas	Soldador <b>SMD</b> Aire caliente	1 (1)	Amazon Faokze	48.95
Herramientas	Bobina Estaño	1 (1)	Amazon lumcov	7.00
Herramientas	Flux Soldadura	1 (1)	Amazon Cerioll	7.99
Herramientas	Estaño Pasta Soldadura <b>SMD</b>	1 (1)	Amazon Happy Finding	11.99
Herramientas	Kit Destornilladores	1 (1)	Aliexpress WoodPow	11.99
Total				23049.30

Tabla 3.9: Servicios Y herramientas

**Total Presupuesto Unidad**

También es interesante saber el coste asociado a una unidad, sin precio de las horas de trabajo de diseño ni de montaje solo de materiales. Para poder hacer un estimado a la hora de venderlo como producto. El valor de un teclado en materiales con los envíos y todo ajustado a la cantidad de material que consume un teclado asciende a (25,36 € en la parte la electrónica), (80,21 € para la parte obligatoria del montaje + 12,13 € con batería), (80,41 € de la elección de [Keycaps](#) e interruptores). Se debe tener en cuenta el servicio del fresado, ya que es obligatorio para el producto, por lo que son 51,43 €.

El total del producto en el caso de las elecciones hechas es de 198,11 € y de **249,54 €** con el servicio.

Como producto para la venta en internet de producto de lujo y [DIY](#) tendríamos que añadir los costes asociados a las herramientas y al tiempo que se tarda en montarlo. Este total es de **409,4 €**.

# Capítulo 4

## Circuitos

Como ya se decidió en la sección 3.3.1 se va a usar la herramienta de diseño Eagle. Usaremos la versión gratuita, ya que las restricciones de esta son en tamaño máximo de la placa. Una vez que tenemos listados los elementos que necesitamos, ya que hemos hecho el diseño previo y sabemos que componentes van a estar presentes, podemos buscar librerías que nos traigan esos componentes para la herramienta Eagle.

### 4.1. Búsqueda de componentes

Vamos a buscar todos los componentes que van sobre la **PCB**, estos son todos los de la tabla de componentes electrónicos 3.7 y los interruptores de la tabla de Montaje 3.8.

#### Multiplexor

Para este componente se ha hecho una búsqueda en internet y la página donde se ha encontrado es Snapeda [27] en la sección de partes podemos hacer una búsqueda y encontramos fácilmente **74HC154D,653** [29].

#### Condensador

Para este componente se hizo lo mismo, pero al ser un componente más genérico se puede buscar por el tipo de formato que tiene, ya que hay muchos tipos de formatos para componentes como resistencias y condensadores. En el caso del componente elegido es el formato o paquete 1206, por lo que podemos buscar realmente cualquier condensador con ese tipo de paquete y el valor se lo cambiamos en el programa. En la misma página mencionada anteriormente encontramos el condensador del formato que buscamos [30].

### Resistencia

De la misma forma, se ha hecho con la resistencia, en este caso hay varias, pero se han escogido todas del mismo formato para simplificar el trabajo, el formato de la resistencia o paquete es el 0805. En la misma página que las dos anteriores podemos encontrar una resistencia genérica [35].

### TP4056

Este si es un chip específico, por lo que la búsqueda va a ser de este chip en concreto. En la misma página SnapEda [27] encontramos el chip **TP4056** sin problemas [36].

### DW01A

En la misma página SnapEda [27] encontramos el chip **DW01A** sin problemas [31].

### ME6211C33

Aunque es un chip específico, tiene un formato muy común, lo que facilita la búsqueda, ya que se corresponde con cualquier regulador de tensión **SMD**. En la misma página SnapEda [27] encontramos el chip **LM3940IMP** con el mismo formato que nuestro **ME6211C33** [34].

### FS8205A

Este chip es otro específico por lo que la búsqueda va a ser de este chip en concreto. En la misma página SnapEda [27] encontramos el chip **FS8205A** sin problemas [33].

### ESp32S3

Este chip es otro específico por lo que la búsqueda va a ser de este chip en concreto. En la misma página SnapEda [27] encontramos el chip **ESp32S3** sin problemas, aunque este posteriormente lo vamos a modificar. [32].

### LEDS

En cuanto a los **leds** podremos encontrar otros de forma genérica, igual que hicimos con las resistencias o condensadores. En la misma página que todo lo anterior podemos encontrarlo de nuevo [37].

**Diodos**

Como vuelve a ser algo genérico un diodo de paquete SOD-123 podemos buscar cualquier diodo de este formato, aunque el nuestro sea el **1N5819W**. En la página que hemos usado anteriormente podemos encontrar fácilmente muchos diodos con ese formato [28].

**Interruptores**

Para los interruptores vamos a prescindir de la página anterior, si no que vamos a buscar de nuevo en internet porque hay una buena comunidad detrás de todo este tipo de dispositivos y será mucho más fácil encontrar lo que necesitamos. En seguida encontramos un repositorio con la librería específica que necesitamos para nuestros interruptores [4].

**4.1.1. Creación de componentes**

Como se ha mencionado antes en la subsección de componentes 4.1, vamos a modificar el componente ESP32S3.

La modificación hará que soldar este componente sea más sencillo, ya que facilitará la soldadura **SMD** haciendo que la parte inferior con los pines de soldadura que van pegados a la placa sean visibles desde el otro lado de esta.

**Nota**

Ver el apéndice 12.1.1 para más información sobre el diseño de este componente.

## 4.2. Diseño Esquemático

Para el diseño esquemático se han importado todos los componentes a Eagle y se han creado distintas partes del sistema para posteriormente interconectarlas todas en el diseño final.

### Nota

Ver el apéndice 12.1 para más información sobre el diseño esquemático y las consideraciones tomadas.

### Matriz del Teclado

En esta parte se ha seguido el ejemplo básico que podemos ver en la figura 3.18 en el apartado de diseño, donde se opta por una configuración de filas y columnas con su respectivo diodo para evitar **Ghosting**. En la fase de diseño se decidió que iba a ser una matriz de 16\*6 teclas únicas, una tecla especial individual y 13 teclas repetidas. La matriz finalmente queda como se muestra en la figura 4.1.

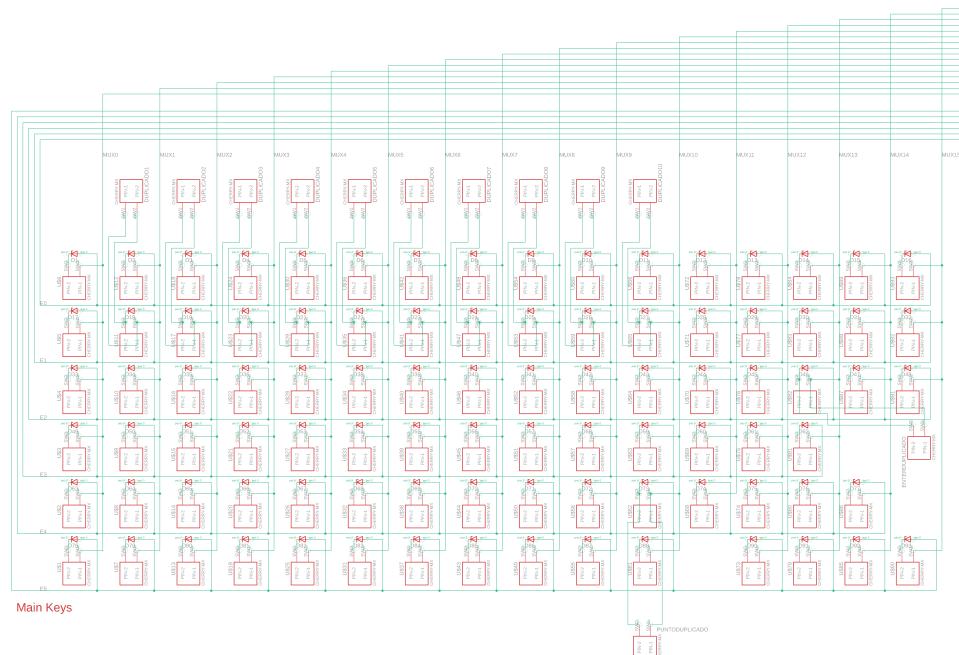


Figura 4.1: Matriz final de teclas [6]

### Multiplexor

Esta parte se trata de las entradas y salidas del **multiplexor** hacia la matriz y al microcontrolador. Esta parte es bastante sencilla como se puede apreciar en la figura 4.2.

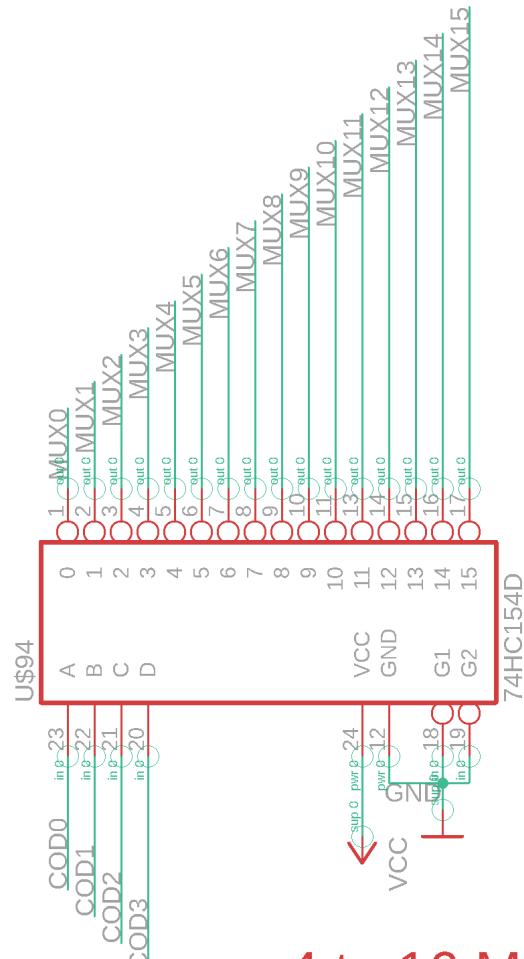
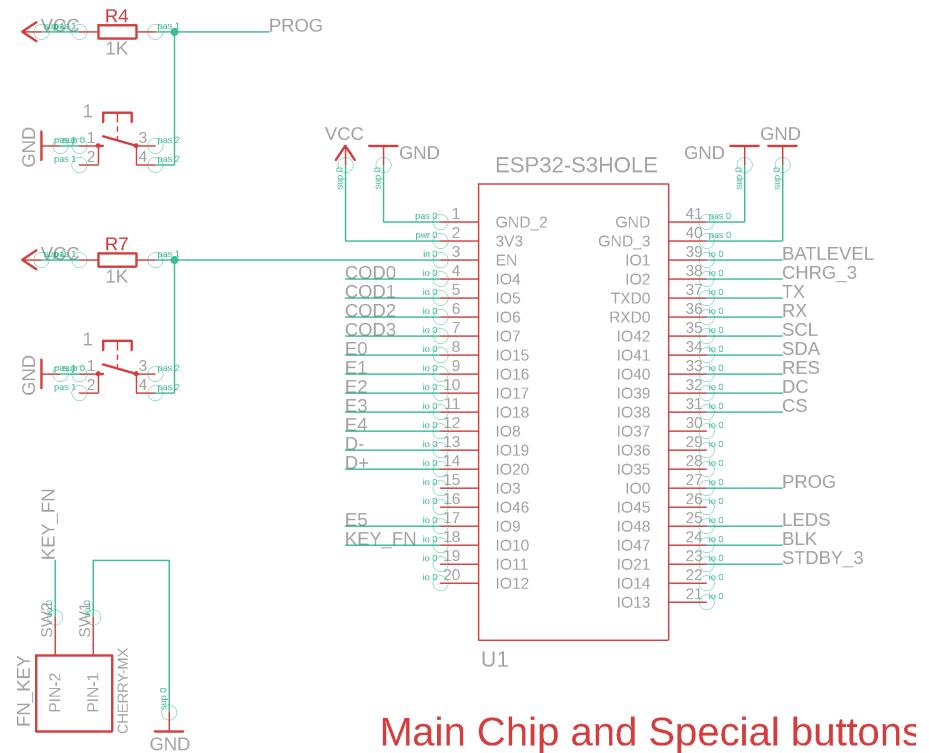


Figura 4.2: **Multiplexor** con todos los pines conectados a sus salidas/entradas [6]

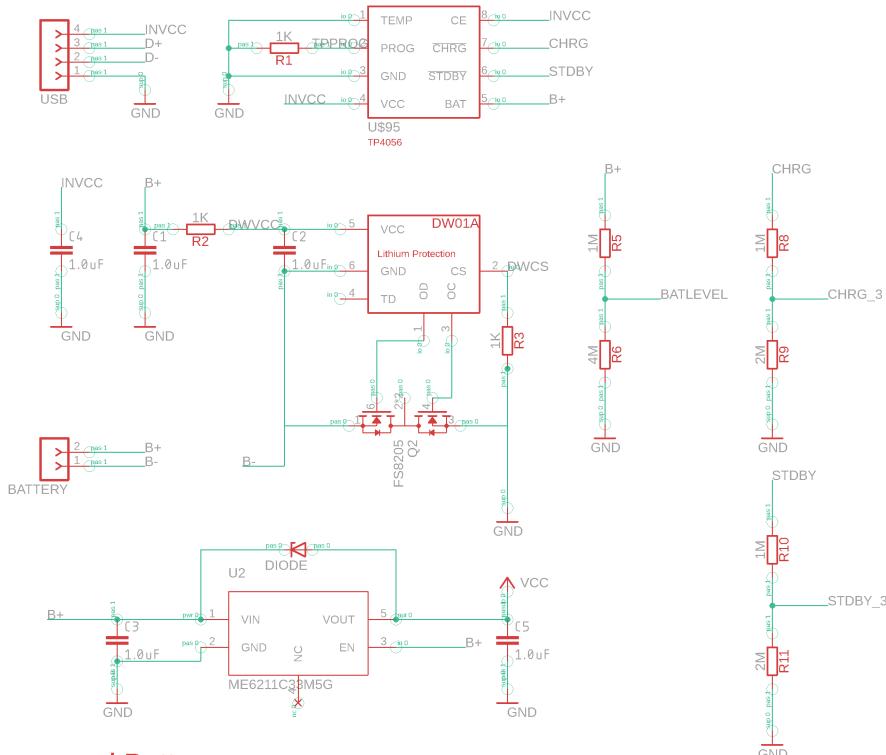
### Controlador Principal

Esta parte del circuito es el chip principal (ESP32S3) junto con los botones necesarios para programarla en el momento de subir el firmware, las resistencias que mantienen la placa encendida y la única tecla especial. Todo se puede apreciar en la figura 4.3



### Batería y alimentación

Como este teclado tiene que soportar alimentación por batería a 4.2 V, por USB a 5.0 V y a 3.3 V para el ESP32S3 es necesario un circuito que se encargue de estos voltajes, de proteger la batería y proteger todos los circuitos. Se puede ver el circuito encargado de esto en la figura 4.4. Todo el diseño se ha hecho siguiendo el apartado 3.4.2 y la figura 3.17.



Power and Battery

Figura 4.4: Sistema de alimentación y protección del teclado [6]

#### Errores

Ver apartado de errores 8.2.1 y 8.2.2 donde se explica por qué se ha añadido un diodo en la puerta del regulador de tensión.

## LEDS

En la fase de diseño, en el punto 3.4.2 se decidió que tuviera leds, ya que eran fáciles de programar y daban mucho juego. El apartado eléctrico de los leds se puede observar en la figura 4.5.

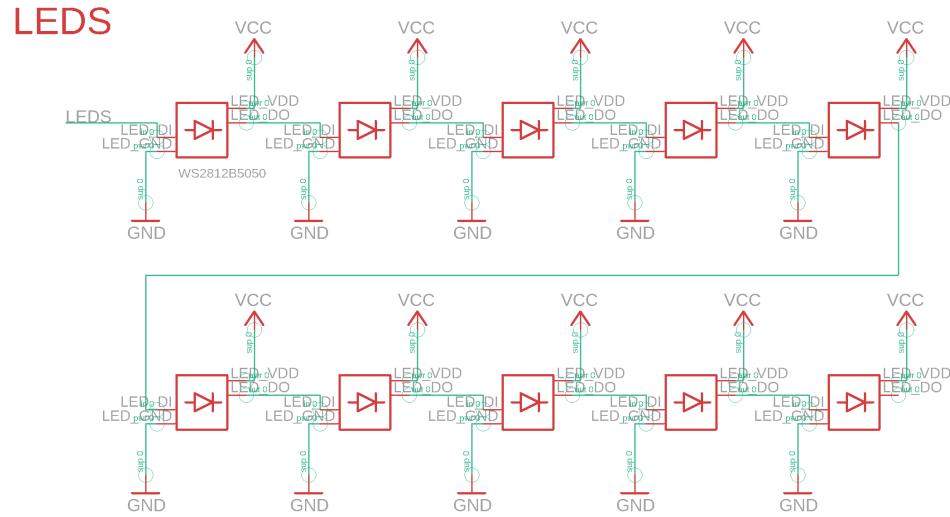
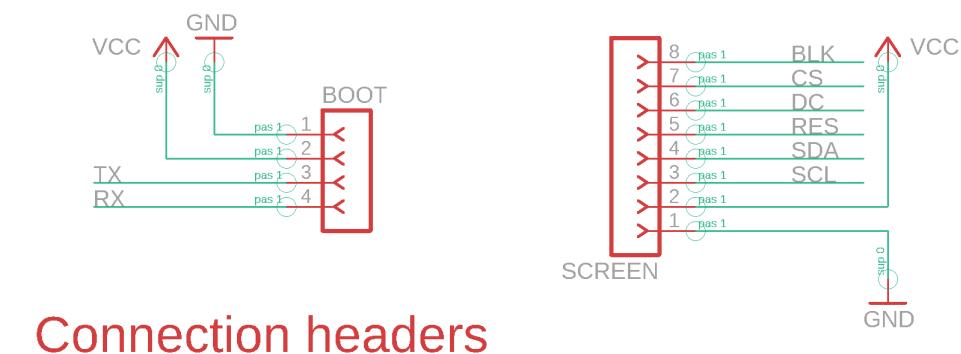


Figura 4.5: Sistema de leds del teclado [6]

## Conectores para el teclado

En el apartado de diseño 3.4.2 aparecen los conectores que necesita la pantalla. Además, se ha añadido otro conector más para poder programar el chip cuando éste sea soldado en la placa, por esta razón, como se observa en la figura 4.6, nuestro teclado se compone de 2 conectores básicos.



## Connection headers

Figura 4.6: Conector de programación y conector de la pantalla TFT [6]

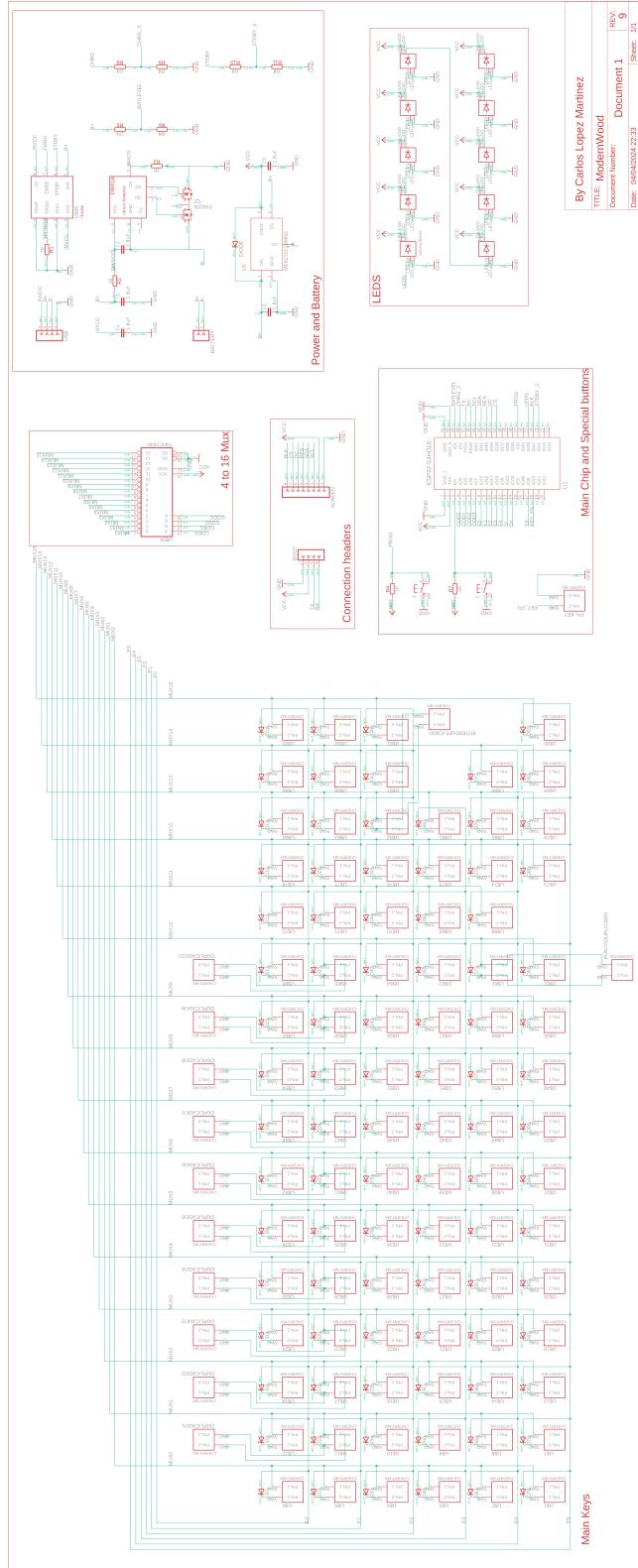


Figura 4.7: Esquema completo del circuito del teclado. [6]



# Capítulo 5

## PCB

### 5.1. Diseño físico

Como ya se decidió en la sección 3.3.1 y como ya se ha usado para el diseño esquemático, El diagrama de la disposición de las teclas nos permitirá trasladar los componentes a un modelo físico de la placa base. Pero antes de poder mover los componentes a la placa, deberíamos saber a qué lugar van y de que forma, por lo que antes de ponernos con el diseño de la placa deberemos diseñar el esquema físico del mismo. Para este apartado usaremos la herramienta AutoCad que se especifica en la sección 3.3.1.

Será necesario el diseño de las siguientes partes, **PCB**, Carcasa, Disposición de teclas, ubicación de los componentes y la ubicación de los conectores.

Estos diseños puede ser creados a partir de un mismo archivo de AutoCAD con las 3 vistas. Alzado, Perfil izquierdo y Planta. Esta última nos servirá para el diseño de la **PCB** y de la disposición de teclas, así como poder crear el archivo 3D para la carcasa. Con el resto de vista nos serviremos para saber si el tamaño de los cortes y disposiciones de los elementos del teclado encajan entre sí y hay suficiente espacio para albergarlos en el interior de la carcasa.

El orden del diseño va a ser el siguiente: Disposición de teclas, diseño de la **PCB**, diseño de la carcasa. Una vez tenemos esto podemos crear la **PCB** en el programa EAGLE. El resto de los planos serán de confirmación para asegurarnos de que lo que estamos haciendo saldrá correctamente.

#### Nota

Ver el apéndice 12.2 para más información sobre el diseño los planos y las consideraciones tomadas.

### 5.1.1. Diseño de la distribución

En el capítulo de diseño 3 en la sección 3.4.1 se encontró una página web “Plate & Case Builder” [40] que nos generaba un plano para AutoCad basándonos en la distribución de teclas creada en la página web “Keyboard Layout Editor” [19].

Por lo tanto, vamos a proceder a crear el plano primero para saber la distancia relativa entre las teclas y sus posiciones entre sí. Una vez introducido en “plate Layout” el texto que nos genera la página del editor de distribución le damos a generar archivo CAD. Este nos genera el plano que podemos ver en la figura 5.1.

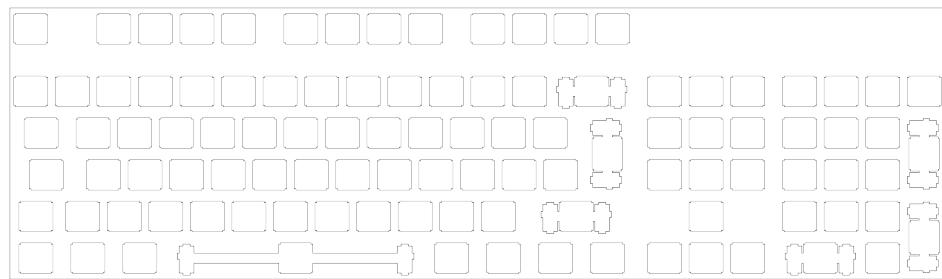


Figura 5.1: Imagen del plano generado por la página web “Plate & Case Builder” [40]

### 5.1.2. Medidas Físicas

A partir del plano generado en la sección 5.1.1 empezaremos a crear el resto de planos en AutoCad.

El primer paso es crear las dimensiones de la **PCB**, que en este caso van a ser el mínimo necesario para poder encajar todas las teclas y un borde de seguridad para que las **keycaps** no toquen el borde de la madera.

También para obtener el plano de la **PCB** es necesario añadir los tornillos, ya que como se diseñó en la sección 3.1 se decidió que el montaje del teclado iba a ser sin **plate** como en la figura 3.12, la fuerza de las manos presionando las teclas va a ser transmitida directamente a la **PCB**. Esta fuerza hará que se doble fácilmente por el material del que está fabricada, que es menos rígido que las **plate** convencionales. Se van a necesitar varios tornillos distribuidos a lo largo de toda la **PCB** para poder hacer un teclado robusto.

Una vez que se ha importado el plano generado de las posiciones de los interruptores se va a proceder a quitar algunos elementos innecesarios de marcado y simplificar algunos elementos del plano generado.

Los tornillos van a ser de 3 mm o los llamados M3. Estos son de un tamaño adecuado para que puedan ser ubicados entre las juntas de las teclas y en los bordes. Por lo que se dispondrán circunferencias a lo largo del plano en púrpura indicando las posiciones de los agujeros que además serán los tornillos ciegos de la carcasa en un futuro. Una vez hecho esto nos quedamos con 25 tornillos en la disposición final que podemos ver en el plano de la PCB en la figura 5.2.

El teclado va a necesitar otro tipo de orificios para sujetar un difusor de luz para los leds, por lo que Vamos a tener que idear la posición de los leds y además añadir los agujeros específicos para sujetar el difusor, al colocar los leds aprovechamos también para colocar los planos de los diferentes componentes. Vamos a añadir el **multiplexor**, el ESP32-S3, y el conector XS-12 y la pantalla que se había decidido colocar en la sección 3.4. Los agujeros para los leds estarán marcados en azul y los componentes en naranja.

Una vez añadidos al plano todos los componentes importantes, la pantalla y haber marcado todos los orificios que necesita la PCB nos queda, por fin, el plano completo de la PCB con todas las medidas correctas y en su lugar correspondiente. El plano puede verse en la figura 5.3.

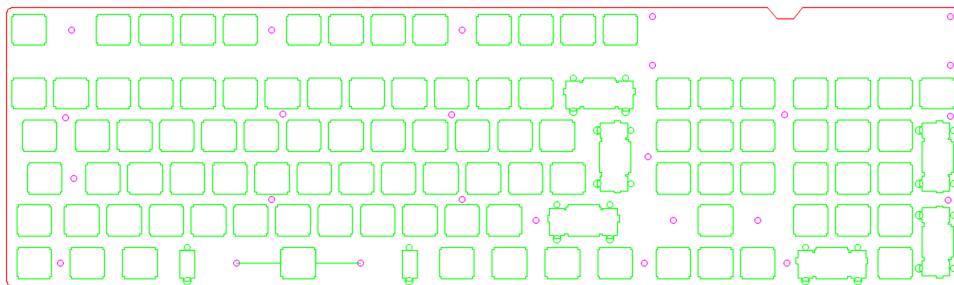


Figura 5.2: Imagen del plano de la PCB [7]

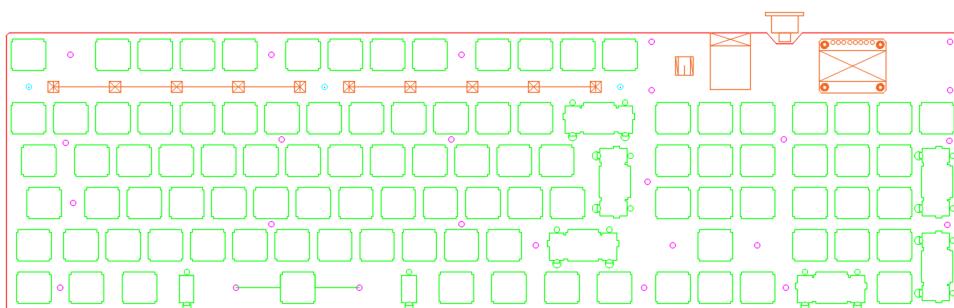


Figura 5.3: Imagen del plano completo de la PCB [7]

### 5.1.3. Creación de la placa en Eagle

Para la creación de la placa o del archivo necesario para poder encargar la placa por internet, vamos a volver a la herramienta Eagle, dado que ya tenemos el diseño esquemático completo hecho. Gracias a los componentes que hemos buscado en internet podremos hacer la parte física de una forma sencilla. Estos componentes tienen la información de como es la pieza física, por lo que si nos vamos al plano podemos encontrar todas las piezas dispersas como se ve en la figura 5.4.

Para poder colocar estos componentes en su lugar adecuado tendremos que importar el plano que hicimos en AutoCAD. Lo guardaremos como .DXF y en Eagle le daremos a importar dxf. Una vez importado el plano nos aparecerá una capa nueva llamada documentación como se puede ver en la figura 5.5. En esta capa estará albergado el plano de la **PCB**. Ahora la tarea pendiente es colocar los elementos y componentes a lo largo de todo el plano siguiendo el orden correspondiente.

Una vez colocados todos los elementos nos quedará listo para empezar a conectar todo con las llamadas líneas aéreas que son, simplemente, líneas amarillas que nos indican a donde tiene que ir la vía de conexión. Esto lo podemos ver en la figura 5.6.

En la misma figura 5.6 ya hemos añadido los agujeros a la placa en sus correspondientes lugares, además de dimensionar la capa de **PCB** que nos dará la dimensión de la misma. Los agujeros se han realizado con la herramienta de “Drill” que nos permite localizar un taladro en el lugar que lo pongamos del tamaño que le indiquemos, también hemos creado las líneas de la capa “dimensión” que nos indica como debe ser la **PCB**, éstas han sido colocadas siguiendo el borde rojo de la figura 5.3 en el plano de Eagle.

## PCB

### 5.1.3. Creación de la placa en Eagle

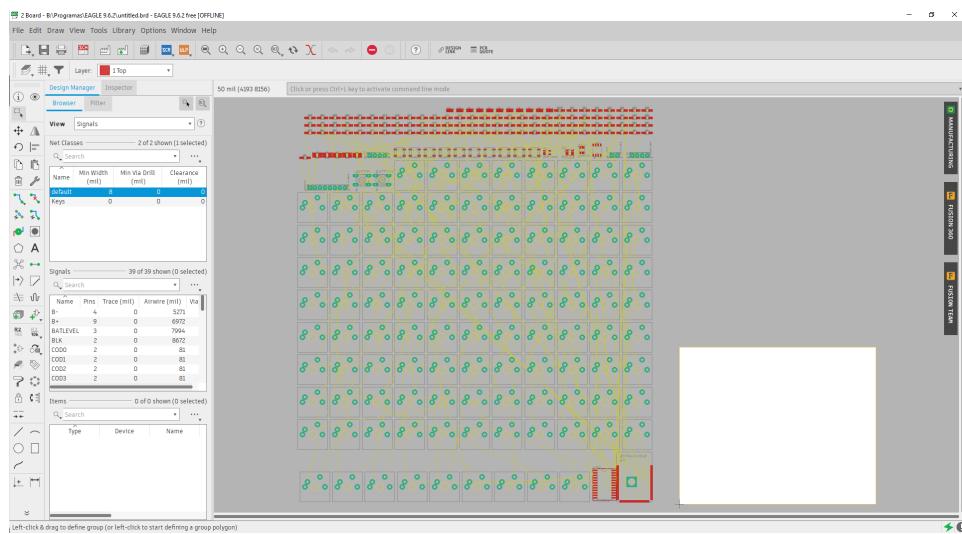


Figura 5.4: Imagen de la vista PCB en Eagle.

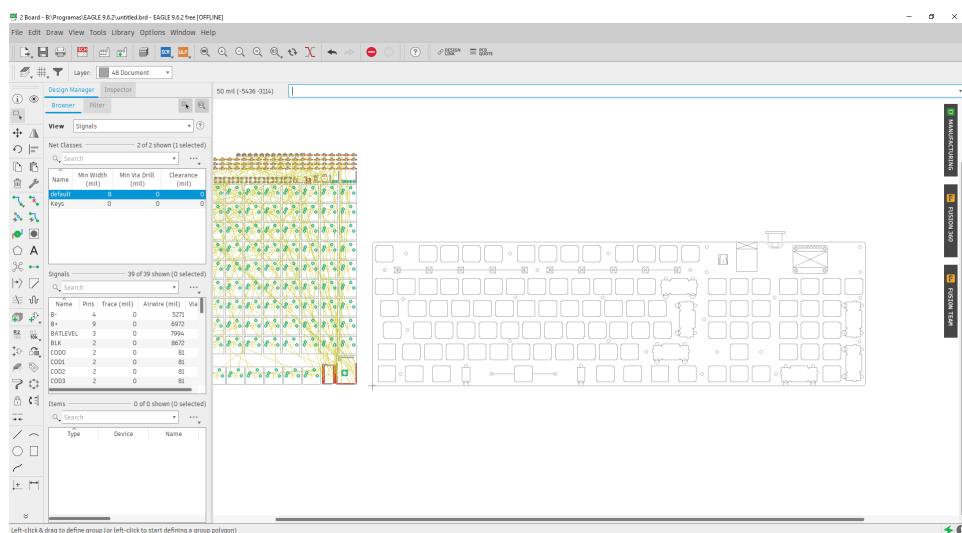


Figura 5.5: Imagen del plano importado en Eagle.

### 5.1.3. Creación de la placa en Eagle

PCB

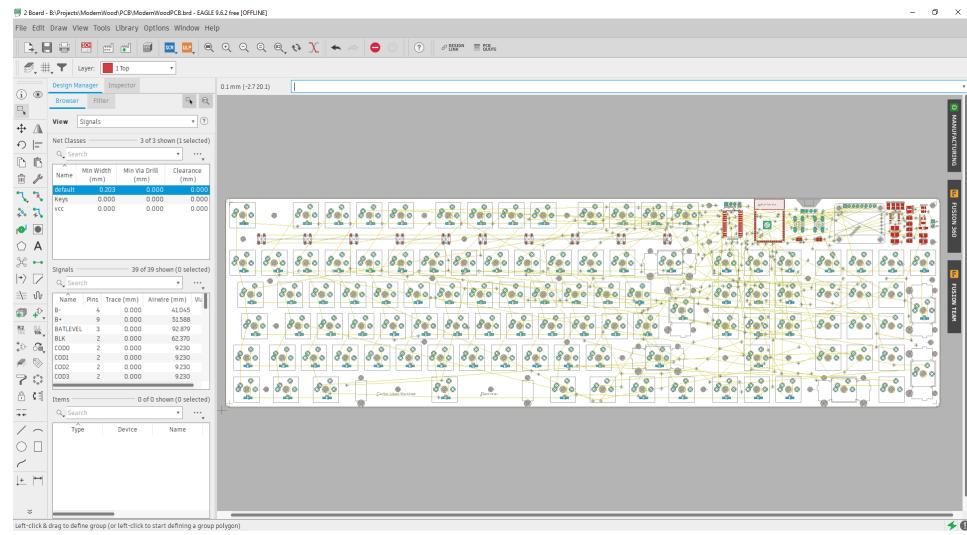


Figura 5.6: Imagen de la PCB con los componentes ubicados y los agujeros.

### Consideraciones de potencia y diseño

Durante el diseño ha sido fundamental prestar atención a los voltajes de funcionamiento de las diferentes partes del teclado, ya que hay varios niveles de voltaje en todo el proyecto. Estas consideraciones también tienen que estar presentes ahora, ya que durante el enrutamiento que vamos a realizar a continuación, las pistas/carriles o vías que creemos tienen que cumplir unas características para el correcto funcionamiento del dispositivo.

En primer lugar, se debe prestar especial atención al dimensionamiento de los carriles de alimentación y conexión entre los distintos componentes del teclado. Los carriles deben tener el tamaño adecuado para manejar la corriente requerida por los componentes sin causar caídas de tensión significativas que puedan afectar al rendimiento del teclado.

Además, la posición de los elementos de potencia y otros componentes críticos debe ser cuidadosamente considerada. Una disposición incorrecta puede resultar en interferencias electromagnéticas (EMI) o en problemas de disipación de calor, lo que podría afectar negativamente al funcionamiento del teclado y su durabilidad.

Listo para arreglar tus errores en 0.3 segundos. Para ello vamos a usar herramientas de cálculo de pistas o vías para saber qué características tienen que tener. Calcularemos que distancia mínima y qué tamaño tienen que tener las pistas. Para conseguir esto vamos a usar dos páginas que tienen calculadoras específicas para realizar esta tarea.

El primer valor va a ser el tamaño de la vía, para ello vamos a usar la página “4pcb” [3] y para la distancia entre pistas “protoexpress” [26].

Empecemos con el tamaño de la vía, los parámetros que nos pide son los siguientes.

- **Intensidad.** Este valor para nosotros va a valer como máximo 0,5 Amperios, ya que es el máximo que el **USB** nos da.
- **Grosor.** Este valor es un estándar, así que para nosotros es de  $2 \frac{\text{oz}}{\text{ft}^2}$ .
- **Temperatura,** Incremento y ambiente. Nuestra temperatura ambiente será de 25ºC y el incremento le pondremos de otros 25ºC.
- **Longitud de la pista.** La distancia. Aquí vamos a elegir la pista más larga en nuestro teclado que será de lado a lado de unos 50 cm.

El resultado que nos arroja esta calculadora es de un grosor de 0.0861 mm. Como podemos ver, el tamaño de la pista es sumamente pequeño, por lo que los valores por defecto de Eagle nos van a valer perfectamente y además mejorará la resistencia y la caída de voltaje.

Para la distancia de la pista los parámetros que nos piden son:

- **Máximo voltaje de la pista.** Para nosotros el voltaje más alto que tenemos en todo el circuito es de 5V (Entrada del [USB](#)).

Tras darle a calcular, el valor que obtenemos es de 0,0508 mm. Otra vez los valores por defecto del programa superan con creces este valor, por lo que por ahora cumplimos con todos los requisitos.

Cabe mencionar, que dado que el microcontrolador ESP32S3 es el que posee la antena [Bluetooth](#) integrada, no es necesario calcular nada relacionado con la antena. Ya que el fabricante del microcontrolador ya ha hecho este trabajo por nosotros.

### **Enrutamiento**

Para la sección de las teclas se ha usado la herramienta de enrutamiento automático de Eagle. Esta herramienta nos permite conectar todas las pistas de forma rápida y eficiente. Una vez que se ha realizado el enrutamiento automático se ha revisado manualmente para asegurarse de que todas las conexiones son correctas y que no hay errores en el diseño.

Para la sección del microcontrolador y los componentes críticos se ha realizado el enrutamiento manualmente. Esto se debe a que estas secciones son más críticas y requieren una mayor atención para asegurarse de que todas las conexiones son correctas y que no hay curvas extrañas que puedan afectar al rendimiento del teclado.

Una vez conectado todo nos quedaría la [PCB](#) terminada. La podemos ver en la figura 5.7. En esta figura podemos ver que todas las pistas están conectadas y que todos los componentes están en su lugar correspondiente. Además, se han añadido las pistas de alimentación y las pistas de conexión entre los diferentes componentes.

Además de conectar todos los componentes, se ha añadido un plano de tierra en la [PCB](#). Este plano de tierra ayuda a mejorar la disipación del calor y a reducir las interferencias electromagnéticas (EMI) en la placa. También ayuda a mejorar la integridad de la señal y a reducir el ruido en la misma, por lo que siempre es recomendable tener un plano de tierra. Se ha creado un plano para cada una de las capas de la [PCB](#). Como nosotros tenemos 2 capas, hemos creado un plano de tierra en la capa superior y otro en la capa inferior. Estos planos se pueden ver en la figura 5.8 y en la figura 5.9, respectivamente.

## PCB

### 5.1.3. Creación de la placa en Eagle

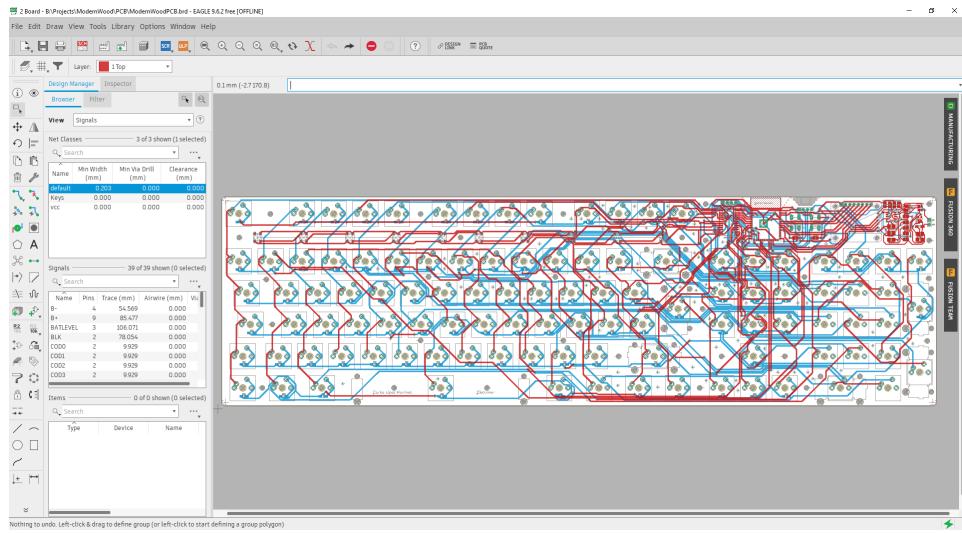


Figura 5.7: Imagen de la [PCB](#) con las pistas conectadas.

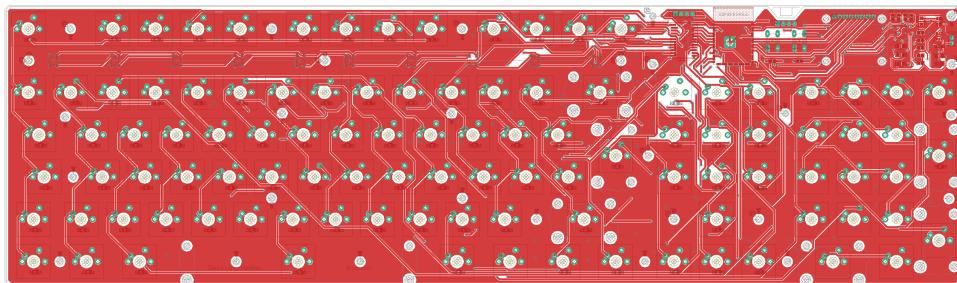


Figura 5.8: Imagen del plano de tierra superior de la [PCB](#) [8].

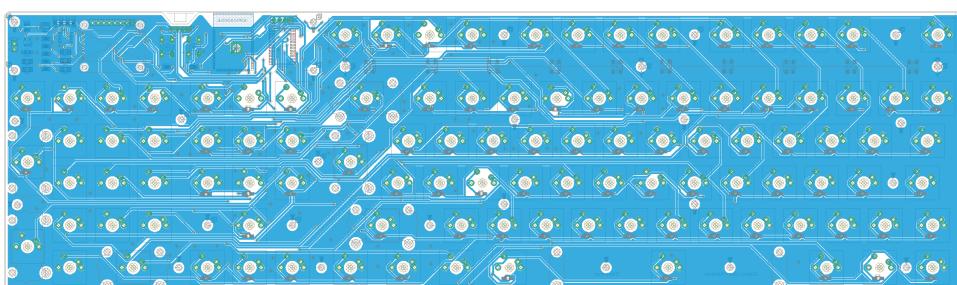


Figura 5.9: Imagen del plano de tierra inferior de la [PCB](#) [8].

**Estética**

Una vez que se ha realizado el enrutamiento y se han añadido los planos de tierra, se ha procedido a mejorar la estética de la **PCB**. Para ello, se han eliminado las marcas de las vías y se han movido las vías para que queden más estéticas. También se ha añadido el texto a mano y se han posicionado todos los elementos de forma correcta. Además, se han añadido indicadores para facilitar la lectura de la **PCB** y se han añadido los números de referencia de los componentes para facilitar la identificación de los mismos.

A su vez, también se han creado unos iconos para poder identificar los tipos de agujeros a los que están asociados. En total hay 4 iconos. Estos son para identificar el difusor de luz, los cuáles tienen un panel de metacrilato para poder proteger la sección de componentes electrónicos, otro para saber cuál es necesario emplear tuerca y el último para saber cuál es necesario atornillar a la carcasa. Respectivamente, estos iconos podemos previsualizarlos en las figuras 5.10, 5.11, 5.12 y 5.13 respectivamente.

Más adelante se usarán para saber de forma rápida que tipo de agujero es necesario en la carcasa y para facilitar el montaje del teclado.

- **Difusor de luz.**



Figura 5.10: Imagen del ícono de difusor de luz [9].

- **Panel de metacrilato.**



Figura 5.11: Imagen del icono del indicador de panel de metacrilato [10].

- **Tuerca.**



Figura 5.12: Imagen del icono del indicador de tuerca [11].

- **Atornillar a la carcasa.**



Figura 5.13: Imagen del icono del indicador tornillo a la carcasa [12].



# Capítulo 6

## Carcasa

La carcasa es la parte visible del producto, es la primera impresión que se lleva el usuario y, por tanto, es una parte muy importante del diseño. En este capítulo se describirá el proceso de diseño y fabricación de la carcasa del producto. Se describirán las medidas físicas, la ergonomía y el proceso de fabricación. Como se diseñó en el capítulo 3, esta será una única pieza de madera que se fabricará con una máquina [CNC](#).

El plano que se utilizó para el diseño de la carcasa se creará con la herramienta AutoCAD para seguir con la decisión de la sección 3.3.1, se exportará a FreeCAD para crear el modelo 3D y se exportará a G-Code para la fabricación con la máquina [CNC](#).

### 6.1. Diseño físico

Para el diseño de la carcasa nos vamos a basar en el plano que ya tenemos de la [PCB](#), ya que en la sección 5.1.2 se dispusieron los agujeros donde iban a ir los tornillos de la misma. Se va a diseñar una carcasa que sea tipo cajón, donde la [PCB](#) se va a deslizar por la parte superior y se va a atornillar a la parte inferior. Se va a dejar un espacio para la batería de la [PCB](#). Para empezar con el diseño importaremos el plano de la [PCB](#) a AutoCAD y se creará la carcasa alrededor de la [PCB](#). Los bordes de la carcasa se van a redondear para darle un aspecto más estético. Además, tendrán un grosor de 7 mm para darle resistencia y poder atornillar el conector XS-12.

También se tendrá en cuenta el espacio acordado de margen de error en la figura 12.2 para que la [PCB](#) pueda deslizarse sin problemas, teniendo en cuenta errores de precisión en la fabricación de la carcasa y la [PCB](#). El plano final que nos queda se puede ver en la figura 6.1.

### 6.1.1. Medidas Físicas

### Carcasa

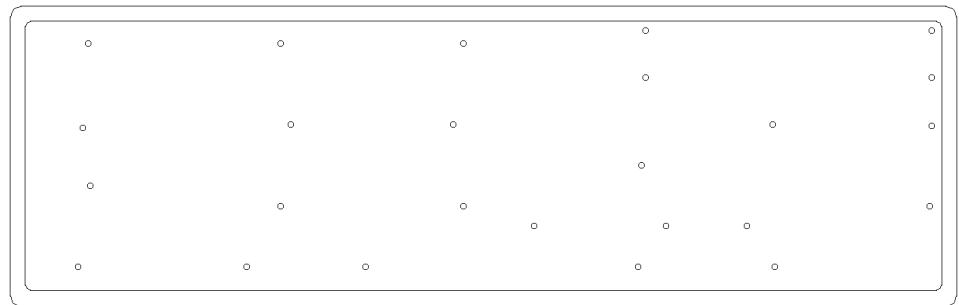


Figura 6.1: Imagen del plano de la carcasa del teclado.

### 6.1.1. Medidas Físicas

A partir del plano de la figura 6.1 se va a realizar el modelo 3D de la carcasa. Para ello vamos a usar FreeCAD y vamos a extrudir las diferentes partes del plano.

Para conocer las dimensiones de la extrusión se van a crear las diferentes vistas de la carcasa y todos sus elementos.

Tras realizar las diferentes vistas ya sabemos las dimensiones exactas para crear la pieza 3D. El resultado final se puede ver en la figura 6.3. Si añadimos el hueco para el conector XS-12 y renderizamos el modelo con una textura de madera podemos tener una idea de como será la carcasa una vez terminada este renderizado lo podemos ver en la figura 6.4.

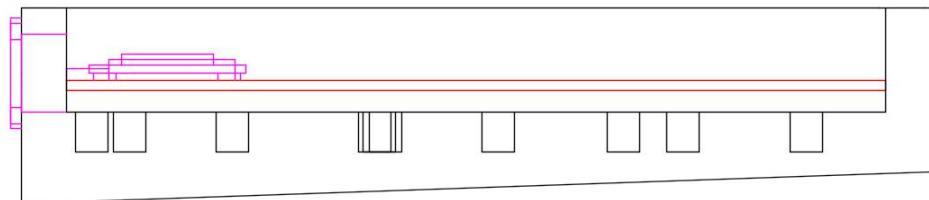


Figura 6.2: Imagen del plano lateral (Vista perfil izquierda) de la carcasa del teclado [7].

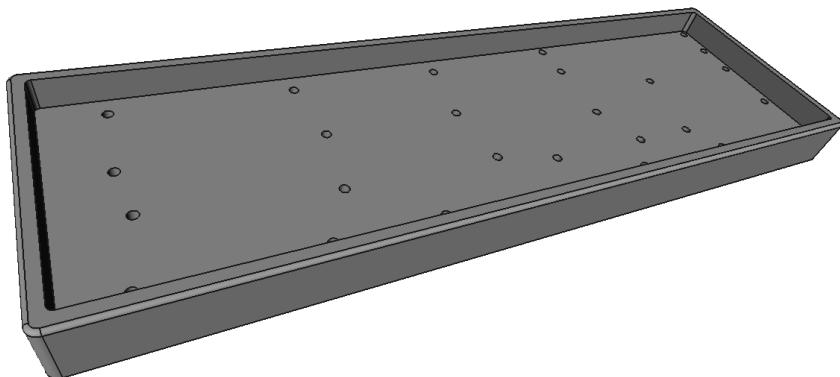


Figura 6.3: Imagen del modelo 3D de la carcasa del teclado.

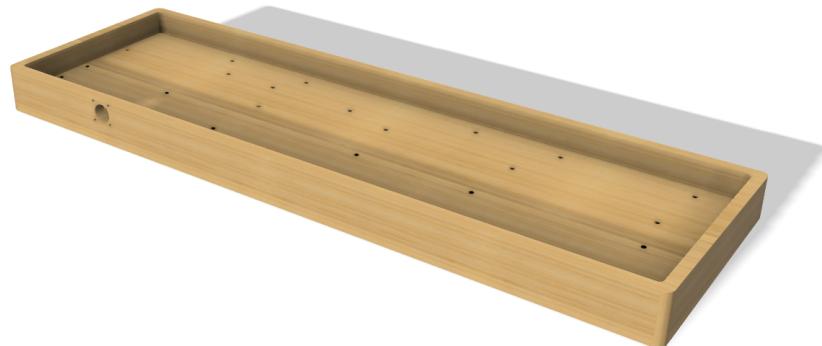


Figura 6.4: Imagen del modelo 3D Renderizado de la carcasa del teclado.

### 6.1.2. Ergonomía

Para la ergonomía de la carcasa se ha tenido en cuenta la altura de la misma. Se ha decidido que la altura e inclinación de la carcasa sea de  $2.5^{\circ}$  para que el usuario pueda escribir de forma más cómoda. La altura de la carcasa es de 3 cm en la parte trasera y 2,5 cm en la parte delantera. La inclinación de la carcasa se puede ver en la figura 6.2.

Para realizar la inclinación se hará cortando con ángulo la parte inferior una vez terminado el [fresado](#) de la carcasa. Una vez finalizado se procederá a lijar la carcasa para que quede con un acabado más fino.

Una vez terminado el [fresado](#) y el lijado se procederá a barnizar la carcasa para darle un acabado más profesional y resistente a la humedad y al uso.

## 6.2. Fabricación

Para la fabricación de la carcasa del producto, se seguirá un proceso detallado que garantice la precisión y calidad del resultado final. Este proceso constará de varias etapas, desde la preparación de los materiales hasta el acabado final de la carcasa. A continuación se detallan las etapas de fabricación.

### Preparación de materiales

Antes de iniciar el proceso de fabricación es crucial asegurarse de tener todos los materiales necesarios en las cantidades adecuadas. Para la carcasa del producto se requerirá una lámina de madera del tipo y grosor especificado en el diseño, así como los dispositivos de sujeción necesarios para fijar la madera durante el proceso de [fresado](#). También se necesitará un barniz protector para el acabado final de la carcasa. Los tornillos de empotrado y un fijador como bien puede ser epoxi para fijar los tornillos a la carcasa.

### Programación de la máquina [CNC](#)

Una vez asegurados los materiales, se procederá a programar la máquina [CNC](#) con el código G generado a partir del modelo 3D de la carcasa. Este código instruirá a la máquina sobre los movimientos necesarios para esculpir la forma deseada en la lámina de madera. Se verificará cuidadosamente que el código esté libre de errores y que refleje fielmente el diseño de la carcasa.

### [Fresado](#) de la carcasa

Con la máquina [CNC](#) debidamente programada, se llevará a cabo el [fresado](#) de la carcasa en la lámina de madera. Durante este proceso, se prestará especial atención para garantizar la precisión en cada paso, asegurando que las dimensiones y formas coincidan con las especificaciones del diseño. Se realizarán las operaciones de [fresado](#) necesarias para crear los contornos, agujeros y detalles requeridos en la carcasa.

### Corte con ángulo y lijado

Una vez completado el [fresado](#), se procederá al corte con ángulo en la parte inferior de la carcasa para darle la inclinación deseada. Este paso se realizará con una sierra de banco con medidor de ángulo para garantizar la precisión en el corte. Posteriormente, se lijará la carcasa para eliminar cualquier imperfección y suavizar las superficies. Se utilizarán lijas de grano

fino para poder aplicar el barniz de forma uniforme y obtener un acabado suave.

### **Atornillado**

Una vez lijada la carcasa se procederá a encolar las tuercas hexagonales para inserción en muebles. Estas tuercas se insertarán en los agujeros de la carcasa para fijar la **PCB** y el conector XS-12. Se utilizará un fijador como epoxi para asegurar que las tuercas queden firmemente sujetas a la madera. Se verificará que las tuercas estén alineadas correctamente y que los tornillos de la **PCB** y el conector XS-12 encajen de forma segura. Una vez completado el atornillado se taparán los agujeros internos con un tornillo o utilizando un tapón para evitar que el barniz entre en ellos.

### **Acabado final**

Finalmente, se aplicará un barniz protector a la carcasa para proporcionarle un acabado profesional y resistente. Este barniz no solo mejorará la apariencia estética de la carcasa, sino que también la protegerá contra la humedad y el desgaste durante el uso. Se dejará secar el barniz según las instrucciones del fabricante. Una vez seco, se inspeccionará la carcasa para asegurarse de que el acabado sea uniforme y de alta calidad.



# Capítulo 7

# Programación

El software de la placa ha sido desarrollado durante el transcurso de este proyecto. Se ha usado el lenguaje de programación C y el entorno de desarrollo [PlatformIO](#) en Visual Studio Code. A continuación se detallarán los aspectos más importantes de la programación de la placa, diseños de las funciones y estructura del código, estructura general de la máquina de estados y las funciones de cada uno de los estados.

También se detallará la estructura de la memoria flash de la placa, donde se guardan los datos de configuración y los datos de la batería.

Se explicará por qué se han tomado ciertas decisiones de diseño del microcontrolador y cómo se han implementado.

## Nota

Durante la mayor parte del desarrollo se ha utilizado una placa comercial para ir programando, ya que no se tenía acceso al producto final. Ver el apéndice 12.4 para saber cómo se han realizado las pruebas del software.

## 7.1. Plataforma

Como se decidió en el capítulo de diseño 3 en la sección 3.5.1. Se ha usado [PlatformIO](#) como entorno de desarrollo. [PlatformIO](#) es un entorno de desarrollo que permite programar en varios lenguajes de programación y para varios microcontroladores. En este caso se ha usado para programar en C para el microcontrolador ESP32.

Para poder usar [PlatformIO](#) vamos a necesitar instalar Visual Studio Code. Para instalar Visual Studio Code basta con ir a la página web de Visual Studio Code y descargar el instalador. Para instalar [PlatformIO](#) basta

con instalar la extensión de [PlatformIO](#) en Visual Studio Code. Para ello basta con ir a la pestaña de extensiones en Visual Studio Code y buscar [PlatformIO](#). Una vez instalada la extensión de [PlatformIO](#) ya se puede usar [PlatformIO](#) en Visual Studio Code.

En [PlatformIO](#) vamos a comenzar configurando el proyecto. Para que automáticamente se configure el proyecto con las librerías necesarias y el microcontrolador correcto. Para ello vamos a crear un nuevo proyecto en [PlatformIO](#) y seleccionamos el microcontrolador ESP32.

Después vamos a crear el archivo platformio.ini en la raíz del proyecto. En este archivo vamos a configurar el microcontrolador y las librerías necesarias para el proyecto. En este caso vamos a usar las librerías de Adafruit NeoPixel, NimBLE-Arduino y TFT\_eSPI. Estas librerías se pueden instalar desde el gestor de librerías de [PlatformIO](#). Para ello, basta con añadir las librerías al archivo platformio.ini. En este caso se han añadido las librerías en el archivo platformio.ini como se muestra en el código 7.2.

- Adafruit NeoPixel
- NimBLE-Arduino
- TFT\_eSPI

Hay algunas recomendaciones en el uso de [PlatformIO](#). Para la ESP32S3, en concreto para la placa de desarrollo ESP32-S3-DevKitC-1 se recomienda usar un filtro de monitor serie para poder ver los mensajes que envía la placa sin información adicional y que no sea relevante. Para ello se puede usar el siguiente filtro:

```
monitor_filters = esp32_exception_decoder
```

Código 7.1: Filtro recomendado de [PlatformIO](#)

El archivo de configuración platformio.ini será el descrito en el código 7.2, donde se configura el microcontrolador, las librerías y algunas opciones de compilación y entorno. Con este fragmento el proyecto se configurará automáticamente con las librerías necesarias y el microcontrolador correcto, dejando el proyecto listo para programar. Las librerías necesarias son las descritas en la sección 3.5.2.

```
[env:esp32-s3-devkitc-1]
platform = espressif32
board = esp32-s3-devkitc-1
framework = arduino
board_build.f_flash = 80000000L
board_build.flash_mode = qio
board_build.flash_size = 16MB
board_build.usb_cdc = 1
upload_speed = 921600
monitor_speed = 115200
board_name = ModernWood
board_upload.vid = 0x2001
board_upload.pid = 0x1111
lib_deps =
    adafruit/Adafruit_NeoPixel@^1.11.0
    h2zero/NimBLE-Arduino@^1.4.1
    bodmer/TFT_eSPI@^2.5.30
build_flags =
    -I modules/include
    -I include
build_src_filter = +<*> +<../modules/src/*>
monitor_filters = esp32_exception_decoder
check_skip_packages = yes
```

Código 7.2: Configuración PlatformIO

#### Nota

Dado que para nuestro dispositivo [HID](#) se quiere que sea algo que aparezca en el sistema como un teclado, se ha añadido a la configuración del proyecto las opciones de “`board.name`” y los correspondientes nuevos [PID](#) y [VID](#). Estos valores son necesarios para que el sistema operativo reconozca el dispositivo como un dispositivo nuevo y no como una placa de desarrollo. Más información en el apartado 12.3.4.

#### 7.1.1. Compilación y entorno

Para compilar el proyecto se puede usar el comando `pio run` en la terminal de [PlatformIO](#), este comando compilará el proyecto y generará el archivo binario que se puede subir a la placa. Para subir el archivo binario a la placa se puede usar el comando `pio run -t upload`. Este comando subirá el archivo

binario a la placa y lo ejecutará. Para ver la salida de la placa se puede usar el comando *pio device monitor*, este comando abrirá una terminal serie donde se podrá ver la salida de la placa.

También se pueden usar los botones que nos aparecen en Visual Studio Code para compilar, subir y ver la salida de la placa. Estos botones se encuentran en la parte inferior izquierda de la pantalla de Visual Studio Code. PlatformIO por defecto detecta automáticamente el puerto serie donde está conectada la placa y la velocidad de comunicación. Si se quiere cambiar la velocidad de comunicación se puede hacer en el archivo platformio.ini en la sección de configuración del proyecto que se muestra en el código 7.2.

## 7.2. Interfaz

Para la interfaz de usuario se ha usado la librería TFT\_eSPI. Esta librería es una librería de Adafruit que permite controlar pantallas **TFT**. Esta librería es muy completa y permite controlar pantallas **TFT** de varios tamaños y resoluciones. En este caso se ha usado una pantalla **TFT** de 0.96 pulgadas y 80x160 píxeles.

Con este software se ha creado una interfaz de usuario muy sencilla. La interfaz de usuario se compone de un menú principal con 6 opciones y dentro de cada submenú hay varias opciones. La interfaz de usuario se controla con los botones del teclado. Los botones elegidos han sido Fn, Escape, las flechas de dirección y la tecla Enter. Con estos botones se puede navegar por el menú y seleccionar las alternativas deseadas. Se han creado unas imágenes para cada tipo de menú y su correspondiente categoría. Estas imágenes se han guardado en la memoria flash de la placa para poder ser leídas por la pantalla **TFT**. Podemos ver el menú principal en la figura 7.1.

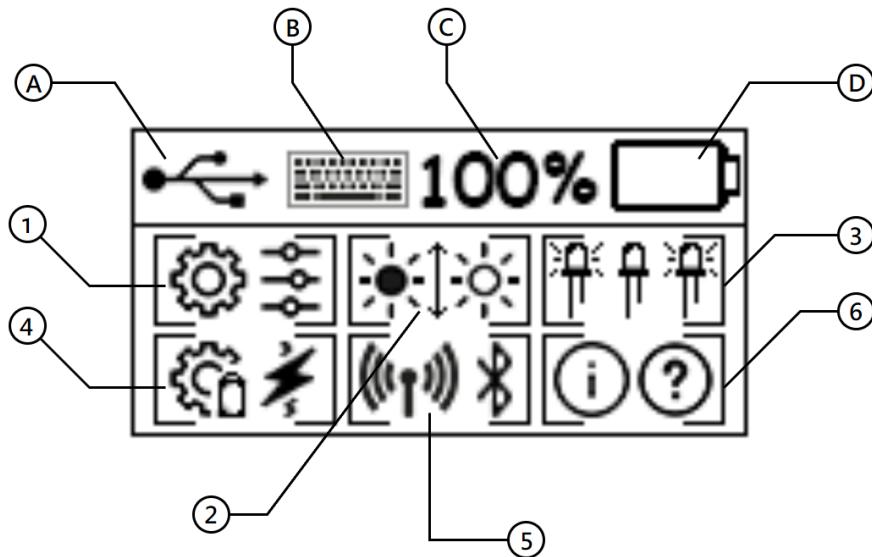


Figura 7.1: Menú principal en la pantalla TFT LCD 0.96“

### 7.2.1. Menú

El menú principal se compone de 6 opciones. Cada opción tiene un ícono que corresponde a una categoría. Además, como se puede ver en la figura 7.1 cada menú va a tener asociado un número. Este número es el indicador de la opción en el menú. Este número se va a usar para seleccionar la opción deseada. Para seleccionar una opción se va a usar la tecla Enter. Para moverse por el menú se van a utilizar las flechas de dirección. Para salir de una opción se va a usar la tecla Escape. Para volver al menú principal se va a usar la tecla ESC. Si se quiere entrar y salir del modo de configuración se va a usar la tecla Fn.

Los diferentes menús vienen señalados en la imagen por un número. Con este vamos a identificar al menú para poder explicar su funcionamiento. Los menús son los siguientes:

- 1. Ajustes
- 2. Brillo
- 3. Leds
- 4. Batería y Energía
- 5. Conexión
- 6. Extra y Ayuda

### Ajustes

Este menú se compone de varias opciones. En este menú se pueden ajustar los parámetros del funcionamiento del teclado. Aquí se pueden ajustar los parámetros de, activar y desactivar la pantalla, activar y desactivar el teclado, ajustar el tiempo de debounce del teclado y ajustar el idioma de la interfaz.

#### Brillo

Este menú se compone de los parámetros de brillo de la pantalla y de los **leds**. En este submenú se pueden encontrar dos ajustes numéricos del 0 al 100. Estos ajustes son el brillo de la pantalla y el brillo de los **leds**. También se pueden llegar a apagar los **leds** y la pantalla mediante este submenú, ya que podemos poner el brillo a 0.

#### LEDs

Este menú se compone de los parámetros de los **leds**. En este submenú se pueden encontrar los ajustes de activar o desactivar los **leds**, el color de los **leds** y el modo de los **leds** y la velocidad de los **leds**.

### Batería y Energía

Este menú se compone de los parámetros de la batería y la energía. En este submenú se pueden encontrar los ajustes de la batería. Para este submenú encontramos activar o desactivar la batería y activar o desactivar el modo de ahorro de energía.

### Conección

Este menú se compone de los parámetros de la conexión. En este submenú se pueden encontrar los ajustes de la conexión. Podemos seleccionar activar el modo de **Bluetooth** por preferencia, activar o desactivar el **Bluetooth** y seleccionar como preferente el modo **USB** sobre el modo **Bluetooth**.

### Extra y Ayuda

Este menú se compone de los parámetros de la ayuda y extras. En este submenú encontramos el restaurar la configuración de fábrica, el modo especial para las funciones programas por el usuario y dos opciones de información y ayuda.

### 7.2.2. Barra de estado

La barra de estado se compone de varios elementos. En la parte superior de la pantalla se puede ver el ícono de la conexión (A), el ícono del modo de funcionamiento del teclado (B), el porcentaje de batería restante (C) y el ícono de la batería (D). Estos elementos se pueden ver en la figura 7.1.

- **A.** Ícono de la conexión: Este ícono indica el estado de la conexión. Este puede alternar entre el ícono de **Bluetooth** y el ícono de **USB**.
- **B.** Ícono del modo de funcionamiento del teclado: Este ícono indica el estado del teclado. Este puede alternar entre el ícono de teclado activado y el ícono de configuración.
- **C.** Porcentaje de batería restante: Este porcentaje indica el porcentaje de batería restante que se actualiza cada 1 minuto.
- **D.** Ícono de la batería: Este ícono indica el estado de la batería. Este indica si la batería está cargando y también la carga que posee con un bloque en el interior de color.

## 7.3. Funcionalidad

La funcionalidad de la placa se ha dividido en varios apartados. Cada apartado se ha dividido en varios estados. Cada estado se compone de varias funciones que se ejecutan en función del estado en el que se encuentre la placa. Para cambiar de un estado a otro se han usado variables de estado a lo largo del código. Estas variables nos indican en qué modo estamos, en qué menú estamos, en qué opción estamos, si hemos cambiado alguna opción, si hemos pulsado algún botón, etc.

Lo que nos permite desde el bucle principal saber en qué estado nos encontramos de la ejecución del programa y qué funciones debemos ejecutar en cada momento. A continuación se detallarán los diferentes estados de la placa y las funciones que se ejecutan en cada uno de los estados.

### 7.3.1. Estados

#### Ahorro de energía

El primer apartado es el ahorro de energía. En esta sección de código se comprueba si el flag de tiempo de ahorro de energía ha sido activado por interrupción hardware de un reloj. Si el flag está activado, se activa el modo de ahorro de energía y se desactiva la pantalla y los **leds**. Si el flag está

desactivado y el flag de que estamos durmiendo está activado, se despierta la pantalla y los **leds**.

### **Batería**

Aquí comprobamos si desde el gestor de interrupciones de la batería ha actualizado el valor de la misma. Si el valor ha variado, se actualiza el valor de la batería en la pantalla y se actualiza el icono de la batería.

### **LEDs**

En esta sección se comprueba si los **leds** están activados en la configuración y si el modo de los **leds** es el modo de **leds** programados por el usuario. Si es así, se ejecuta el modo de **leds** programados por el usuario. Si no es así, se ejecuta el modo de **leds** por defecto.

### **Actualización de conexión**

En esta sección se comprueba si la conexión ha cambiado. Si ha cambiado, se actualiza el icono de la conexión y se actualiza el modo de conexión.

### **Pantalla**

En esta sección se comprueba si la pantalla está activada. En el caso de estar activada, se comprueba que se ha cambiado el estado de la pantalla. En el caso afirmativo se actualiza la pantalla con la información necesaria, además de mostrar siempre la pantalla con el brillo seleccionado.

### **Teclado**

Se comprueba primero que no estamos en el modo de teclas especiales o modo especial, ya que tendríamos que ejecutar la función especificada especial. Si no estamos en el modo especial se procede a funcionar en modo normal del teclado, mientras se comprueba que el flag de la interrupción de la tecla Fn no esté activado. Si está activado, se cambia el modo de teclado a configuración y se activa el flag de configuración. En el modo configuración nos encontramos con lo mismo, se comprueba que el flag de la interrupción de la tecla Fn no esté activado. Si está activado se cambia el modo de teclado a normal y se desactiva el flag de configuración.

En el modo de configuración cada vez que se pulsa una tecla se actualiza el valor de la tecla pulsada y se activa el flag de tecla pulsada, activando la función de actualizar la pantalla.

Estos estados se pueden ver en la figura 7.3 junto con la figura del diagrama de flujo del bucle principal 7.2.

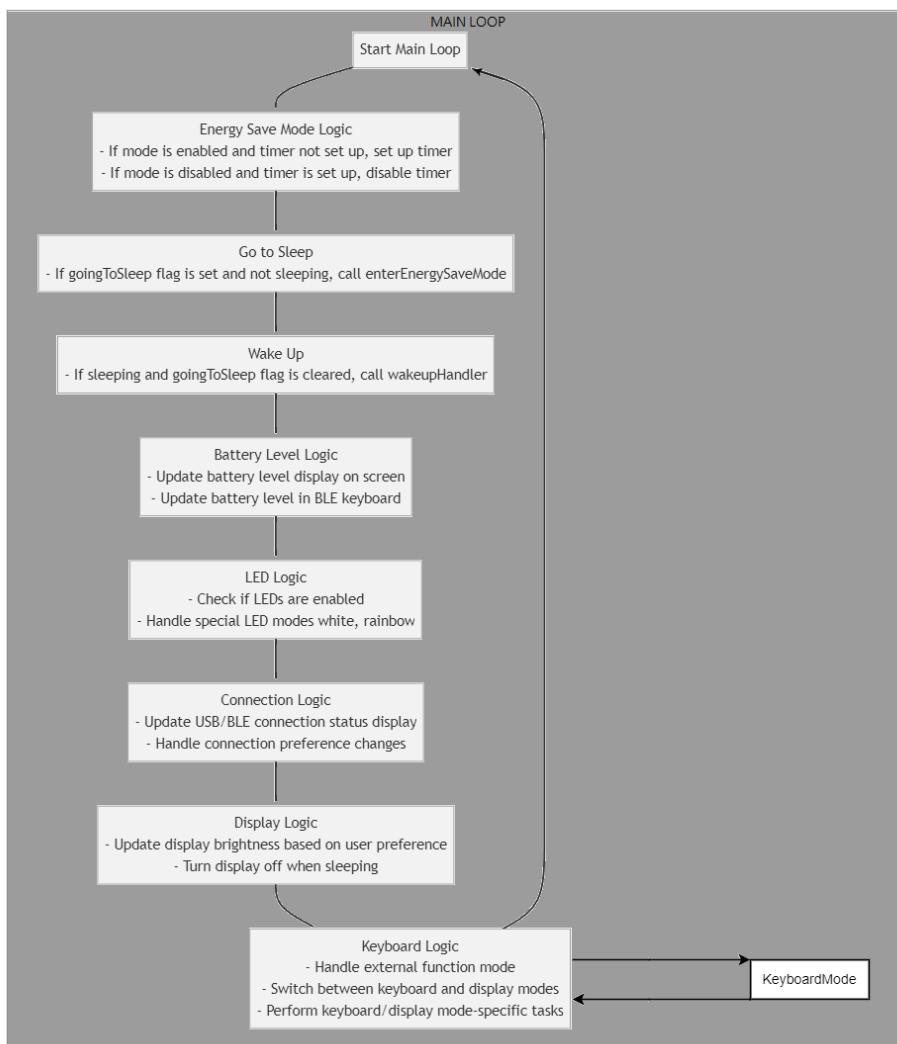


Figura 7.2: Diagrama de Flujo del bucle principal

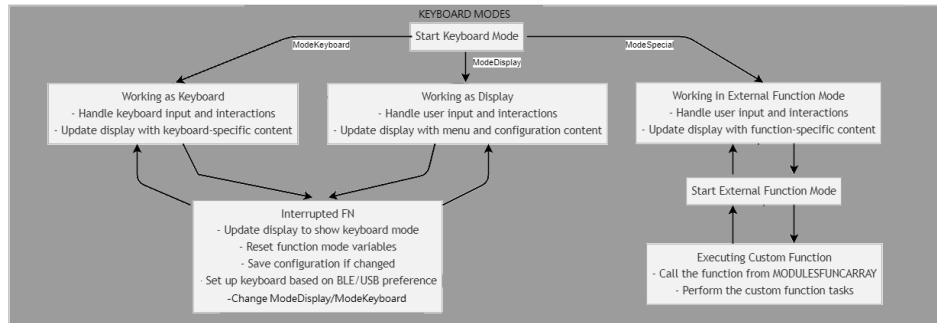


Figura 7.3: Diagrama de Flujo de la secuencia de cambio de modo de teclado

### 7.3.2. Conectividad

Para la conectividad se ha decidido que sea independiente el modo **Bluetooth** y el modo **USB**. Para ello se ha creado un menú de configuración donde se puede seleccionar el modo de conexión preferente. Si se selecciona el modo **USB** preferente, la placa se conectará por **USB** siempre que esté conectado. Si se selecciona el modo **Bluetooth** preferente, la placa se conectaría por **Bluetooth** siempre que esté conectado. Si no se selecciona ninguno de los dos modos, el teclado permanecerá desconectado. Se puede conectar por **USB** y **Bluetooth** a la vez, pero en tal caso tendrá preferencia el modo **USB**.

### 7.3.3. Leds

Para los **leds** se han creado varios modos de funcionamiento. Se ha creado el modo estático, donde el color seleccionado se mantiene fijo. El modo de color blanco y el modo arcoíris. Se podrán añadir más modos creando la función correspondiente y añadiéndola al menú de **leds**. Revisar el apéndice 12.3.5 para más información.

### 7.3.4. Macros

Para los macros o funciones especiales, se tendrá que activar el modo desde el menú “Extra y ayuda”, ya que estas funciones serán creadas por el usuario y se tendrán que añadir al código. Se podrán añadir más funciones creando la función correspondiente y añadiéndola al menú de macros. Revisar el apéndice 12.3.6 para más información.

## 7.4. Boot

Para poder cargar el código al microcontrolador se ha usado el conversor **TTL** que se indicó en la sección 2.2.2. Se ha usado un conversor de nivel lógico para poder programar el chip mientras este se alimenta con el **USB** 5.0 V y no con la batería. El código una vez cargado en el microcontrolador se ejecutará en cuanto el chip revisa el voltaje necesario para funcionar. En este caso el chip se enciende en cuanto se conecta el **USB** y se enciende la pantalla mostrando el logotipo del proyecto. Acto seguido se encienden los leds y se muestra el menú principal en el caso de estar activa la pantalla.

### Errores

Ver apartado de errores 8.2.1 y 8.2.2 donde se explica por qué es necesario el conversor de nivel lógico.



# Capítulo 8

# Prototipos

Durante el desarrollo del proyecto se han realizado varios prototipos para probar las diferentes funcionalidades del sistema. En este capítulo se describen los prototipos realizados y las pruebas realizadas. Se ha usado la placa de desarrollo ESP32S3-DevKitC-1 para realizar los prototipos en una protoboard junto con un [multiplexor](#) DIP en vez de [SMD](#) para facilitar la conexión de los cables. Durante el desarrollo de los prototipos se ha empleado sólo un pequeño porcentaje de las teclas del producto final. Sólo se han usado las teclas para controlar la pantalla y 3 teclas extra de funcionalidad, una letra, el shift y el espacio.

## 8.1. Versiones

Durante el desarrollo del proyecto se han realizado varias versiones de los prototipos. En este apartado se describen las diferentes versiones de los prototipos realizados.

### 8.1.1. V1: Correcciones

La primera de todas las versiones no contemplaba protocolo [bluetooth](#) y tampoco poseía la pantalla [OLED](#). Se realizó para probar el funcionamiento de los interruptores y la [PCB](#). Se usó un microcontrolador Atmega32U4 y un [multiplexor](#) DIP para facilitar la soldadura.

Principalmente, este prototipo fue para probar las dimensiones de la [PCB](#) y la disposición de los interruptores, además de la ergonomía del teclado. Esta primera versión se llama Tesseract [5].

### 8.1.2. V2: Correcciones

Esta versión fue una modificación del teclado Tesseract [5] para cambiar la ergonomía, ya que era muy tosco y no era cómodo de usar. Se cambió la disposición del puerto **USB**, ya que la primera versión usaba un **USB** B y la nueva versión usa un **USB** A.

### 8.1.3. V3: Modificaciones

Tras las últimas dos versiones se decidió ampliar la funcionalidad creando el nuevo modelo ModernWood. Este incluiría una pantalla **OLED** y un protocolo **Bluetooth** para poder usar el teclado con dispositivos móviles, se cambió el microcontrolador a un ESP32S3, se cambió el material de la carcasa a madera y se añadió un difusor para los **LEDs**, se añadió una batería de litio para poder usar el teclado sin cables y se cambió el conector **USB** a un conector XS-12 para que fuera todavía más robusto.

## 8.2. Problemas y Errores

En este apartado se describen los problemas y errores encontrados durante el desarrollo de los prototipos. Estos no han sido corregidos en las versiones de los prototipos, pero se han tenido en cuenta para el diseño del producto final.

### 8.2.1. Corriente inversa en el regulador de tensión

Es común que durante la operación normal de los componentes electrónicos, sobre todo los operados por personas, se produzcan errores en la conexión de los cables, estos son errores que pueden ser fatales para los componentes electrónicos. Uno de estos errores es la conexión inversa de la alimentación. Para evitar que la corriente inversa dañe el regulador de tensión se ha añadido un diodo en la puerta del regulador de tensión.

La función principal del diodo es bloquear cualquier corriente inversa que intente fluir hacia la entrada del regulador. El diodo permite el paso de corriente en una única dirección (de la entrada del regulador a la salida), y bloquea el flujo en la dirección opuesta, de esta manera, se protege el regulador de tensión de posibles daños por corriente inversa.

Además de su función de protección, el diodo también ayuda a estabilizar el funcionamiento del regulador de tensión en condiciones transitorias, como durante el encendido o apagado del circuito. Al evitar que la corriente inversa

llegue al regulador se minimizan las perturbaciones en la salida, asegurando una regulación de tensión más estable y fiable.

### 8.2.2. Voltaje en la salida del regulador de tensión

Este problema, aunque parecido al anterior, la razón es totalmente diferente, en este caso, el problema ocurre cuando queremos alimentar el teclado mediante el programador **TTL**, ya que durante la programación del microcontrolador, el programador **TTL** alimenta el teclado con 5 V o 3.3 V. Sin embargo, el regulador de tensión del teclado está diseñado para funcionar con una tensión de entrada de 3.3 V así, además la alimentación del chip sería directa.

Por lo que el voltaje lo tendríamos en la puerta de salida del regulador de tensión. Así se evitaría lo que se conoce como voltaje “**Back feed Current**” en los reguladores se ha añadido de nuevo un diodo en la puerta de salida del regulador de tensión que permite el paso de la corriente de la puerta de salida a la puerta de entrada saltando el regulador de tensión.

Por esta razón, durante la programación de esta versión del teclado es necesario los conversores de nivel lógico para poder alimentar el teclado con 5 V y a su vez programarlo a 3.3 V y así evitar el problema del voltaje en la puerta de salida del regulador de tensión.

### 8.3. Montaje

Esta sección describe el montaje del teclado en completo detalle. Se usará como explicación para el producto final y para saber el orden y la manera de proceder para el montaje del teclado, ya que está diseñado para ser ensamblado en unos pasos concretos.

#### Errores

Ver apartado de errores en el apéndice 12.5.1 donde se detallan los errores encontrados durante el montaje y los cambios realizados a las diferentes partes del proyecto para corregirlos.

#### Soldadura de los componentes de la [PCB](#)

El primer paso es soldar los componentes de la [PCB](#). Se debe soldar los componentes de menor tamaño primero y los de mayor tamaño después. Se debe tener cuidado con la polaridad de los componentes y con la temperatura de la soldadura para no dañar los componentes, esta debe de ser 250 °C para la pasta de soldadura y 300 °C para el soldador manual. Se debe soldar los componentes en el siguiente orden:

1. Resistencias
2. Condensadores
3. Diodos
4. Chip Batería
5. Microcontrolador
6. Pantalla
7. Interruptores
8. Cables para [USB](#)

Para soldar los componentes [SMD](#) se ha usado una pasta de soldadura de baja temperatura y una pistola de calor para fundir la pasta. Se ha usado un soldador manual para los componentes de mayor tamaño.

La pasta de soldadura se aplica junto con flux en las pistas de la [PCB](#) y se coloca el componente encima. Se calienta la pasta con la pistola de calor hasta que se funda y se adhiera el componente a la [PCB](#). El propio estaño debe deslizarse hasta su posición en la pista y no debe ser empujado con

la pistola de calor. Después de unos segundos de calentar, el chip debería haberse asentado en su posición. Se debe tener cuidado con la temperatura de la pistola de calor para no dañar los componentes.

Se ha de tener en cuenta la fuerza de la pistola de calor, ya que si se pone la fuerza de la misma muy alta podría llegar a calentar otros componentes adyacentes y desoldarlos o dañarlos. [17]

### **Atornillado de los estabilizadores**

Una vez soldados los componentes de la **PCB** se ha de colocar los estabilizadores en la **PCB**, para ello se ha de colocar los estabilizadores en los agujeros de la **PCB** y atornillarlos con los tornillos correspondientes. Se ha de tener cuidado con la fuerza ejercida para no romper la **PCB** ni los estabilizadores.

### **Programación del microcontrolador**

El siguiente paso es programar el microcontrolador. Para ello se debe conectar el microcontrolador a un programador, en nuestro caso el FT232RL y cargar el **firmware** en el microcontrolador. Se debe tener cuidado con la polaridad de los pines y con la conexión de los cables. Se debe conectar el programador al microcontrolador y al ordenador y cargar el **firmware** en el microcontrolador. Se debe comprobar que el **firmware** se ha cargado correctamente y que el microcontrolador funciona adecuadamente.

Para realizar esta tarea se emplearán los pines designados en la **PCB** para la programación del microcontrolador.

### **Difusor leds**

Primero se ha de colocar el difusor sobre los **leds** y se ha de atornillar con los tornillos pasantes sobre el difusor hasta una tuerca autoblocante que deberemos colocar en el otro extremo del tornillo. Se ha de tener cuidado con la fuerza ejercida para no romper el difusor ni la **PCB**.

### **Soldadura de cables para USB**

Tanto en la placa base como en el conector XS-12 tenemos que soldar un cable de conexión Gx16-4 a la parte interna del conector y la parte hembra del conector a los pines correspondientes D+,D-,GND y VCC. Para ello se ha de soldar el cable a la parte interna del conector y a los pines de la placa base. Se ha de tener cuidado con la polaridad de los cables.

**Conektor XS-12**

Con todos los pasos anteriores realizados tendriamos la **PCB** lista para ser montada en la carcasa que, de fabrica, ya tiene las tuercas empotradas en sus agujeros correspondientes y el barnizado completo.

Una vez que tenemos la carcasa barnizada y con las tuercas empotradas debemos colocar el conector XS-12 en su lugar correspondiente, para ello se ha de colocar el conector en la carcasa y atornillarlo con los tornillos M1.7 x 5 mm de cabeza plana.

Después de atornillar el conector XS-12 y tener soldado los dos conectores Gx16 tendremos que conectarlos para dejarlos por debajo de la carcasa.

**Tornillos**

Una vez colocado el difusor y el conector XS-12 se puede colocar la **PCB** en la carcasa, para ello a lo largo de toda la carcasa se encuentran unos agujeros roscados en las tuercas empotradas M3 para atornillar la placa.

En la carcasa se deben colocar los espaciadores de 3 mm en los agujeros de las tuercas empotradas y atornillarlos con un destornillador. Se debe tener cuidado con la fuerza ejercida para no mover o arrancar las tuercas empotradas pegadas con epoxi.

Una vez colocados se pondrá la **PCB** sobre los espaciadores dejando la batería en la parte inferior. Se atornillará la **PCB** a los espaciadores con los tornillos M3 de cabeza redondeada.

Una vez montada la **PCB** sobre la carcasa se colocará el protector de metacrilato sobre los circuitos de la **PCB**. Para poder atornillar el protector de metacrilato a la carcasa se han de colocar unos espaciadores de 5 mm en los agujeros correspondientes a los del protector. Estos agujeros no han debido ser atornillados previamente. Ya que los espaciadores de 5 mm servirán para poder fijar la **PCB** a la carcasa y para poder servir de tuerca al protector de metacrilato.

Una vez colocados los espaciadores de 5 mm se colocará el protector de metacrilato sobre la **PCB** y se atornillará con los tornillos M3 de cabeza redondeada a los espaciadores de 5 mm.

**Keycaps**

Para colocar los **Keycaps** en los interruptores se ha de colocar la tecla sobre el interruptor y presionar hasta que se mantenga sobre el interruptor. Se ha de tener cuidado con la fuerza ejercida para no romper el interruptor ni la **PCB**.

# Capítulo 9

## Validación

En todos los proyectos de desarrollo de software y hardware es necesario realizar pruebas para comprobar que el sistema funciona correctamente. Aquí se destacarán las pruebas realizadas en el sistema para comprobar su correcto funcionamiento.

### 9.1. Pruebas Eléctricas

Para realizar todos los test y pruebas se ha hecho una plantilla típica para llenar, donde se le atribuye un nombre a la prueba, una descripción de la misma, cómo se realiza y el resultado obtenido. Esta se puede ver en la tabla 9.1.

#### Prueba de Continuidad

La prueba de continuidad se llevó a cabo utilizando un multímetro digital. Se verificó la continuidad en los circuitos de las teclas, los LED indicadores y los cables de conexión. No se detectaron interrupciones en la continuidad, lo que indica una correcta conexión de los componentes.

#### Prueba de Resistencia

Se utilizaron instrumentos de medición adecuados para medir la resistencia eléctrica en diferentes puntos del teclado. Los valores de resistencia obtenidos se compararon con los rangos especificados en el diseño. Todos los componentes mostraron valores de resistencia dentro de los límites aceptables.

**Prueba de Corriente y Voltaje**

Se midió la corriente y el voltaje en varios puntos del circuito utilizando un amperímetro y un voltímetro. Los valores de corriente y voltaje se compararon con las especificaciones del diseño. Se observó un comportamiento adecuado de los circuitos, con corrientes y voltajes dentro de los rangos esperados.

**Prueba de Funcionamiento de los LED**

Se realizó una prueba específica para verificar el funcionamiento de los LED indicadores del teclado. Se encendieron y apagaron los LED para confirmar que emitían luz de manera adecuada y que no presentaban fallos de conexión o funcionamiento.

**Prueba de Comunicación USB**

Se verificó la comunicación entre el teclado y la computadora a través del puerto **USB**. Se enviaron datos desde el teclado a la computadora para confirmar que la comunicación era estable y que no había pérdida de información.

**Prueba de Interferencias Electromagnéticas**

El teclado fue expuesto a fuentes de interferencias electromagnéticas para verificar su inmunidad a este tipo de interferencias. Se comprobó que el teclado seguía funcionando correctamente incluso en presencia de campos electromagnéticos externos no muy fuertes.

En resumen, las pruebas eléctricas confirmaron la integridad y el correcto funcionamiento del teclado diseñado, asegurando su fiabilidad y rendimiento en diferentes condiciones de operación.

**9.2. Pruebas en Windows**

Las pruebas en el sistema operativo **Windows** se llevaron a cabo para verificar la compatibilidad y funcionalidad del teclado diseñado en este entorno. A continuación, se detallan las pruebas realizadas. Se ha seguido una plantilla típica para llenar, se puede ver en la tabla 9.2.

### Prueba de Funcionamiento de las Teclas

Se probó cada tecla del teclado para asegurar que todas fueran reconocidas correctamente por el sistema operativo [Windows](#). Se verificó que la pulsación de cada tecla generara la salida esperada en la pantalla y que no se produjeran errores de reconocimiento.

### Prueba de Comunicación [USB](#)

Se verificó la comunicación entre el teclado y la computadora a través del puerto [USB](#) en el sistema operativo [Windows](#). Se confirmó que el teclado fuera detectado correctamente por el sistema y que la comunicación fuera estable y sin errores.

### Prueba de Funcionamiento de los [LED](#)

Se probó el funcionamiento de los [LED](#) indicadores del teclado en el sistema operativo [Windows](#). Se verificó que los [LED](#) se encendieran y apagaran correctamente según el estado de las funciones correspondientes.

## 9.3. Pruebas en [Linux](#)

Las pruebas en el sistema operativo [Linux](#) se realizaron para garantizar la compatibilidad y funcionalidad del teclado diseñado en este entorno. A continuación, se describen las pruebas realizadas. Se ha seguido una plantilla típica para llenar, se puede ver en la tabla 9.3.

### Prueba de Reconocimiento del Teclado

Se verificó que el teclado fuera reconocido correctamente por el sistema operativo [Linux](#) al conectarlo a la computadora. Se comprobó que el sistema asignara los [controladores](#) adecuados y que el teclado estuviera listo para su uso sin necesidad de configuraciones adicionales.

### Prueba de Funcionamiento de las Teclas

Se probó el funcionamiento de cada tecla del teclado en el sistema operativo [Linux](#). Se verificó que todas las teclas generaran la salida esperada en la pantalla y que no se produjeran errores de reconocimiento o asignación de caracteres.

### Prueba de Funcionamiento de los LED

Se verificó el funcionamiento de los [LED](#) indicadores del teclado en el sistema operativo [Linux](#). Se confirmó que los [LED](#) se encendieran y apagaran correctamente según el estado de las funciones correspondientes.

En resumen, las pruebas realizadas en ambos sistemas operativos confirmaron la compatibilidad y funcionalidad del teclado diseñado en diferentes entornos de software.

Nombre	Descripción	Como se realiza	Resultado
Continuidad	Verifica la continuidad	Utilizando un multímetro digital	Sin interrupciones
Resistencia	Mide la resistencia eléctrica	Con instrumentos de medición adecuados	Valores dentro de los límites aceptables
Corriente y Voltaje	Mide la corriente y el voltaje	Utilizando un amperímetro y un voltímetro	Corriente y voltaje dentro de los rangos esperados
<a href="#">LEDS</a>	Verifica el funcionamiento de los <a href="#">LED</a>	Encendiéndolo y apagándolo	Emisión de luz adecuada
<a href="#">USB</a>	Verifica la comunicación <a href="#">USB</a>	Envío de datos desde el teclado a la computadora	Comunicación estable sin pérdida de información
<a href="#">EMI</a>	Prueba la inmunidad a interferencias electromagnéticas	Exponiendo el teclado a fuentes de <a href="#">EMI</a>	Funcionamiento correcto incluso en presencia de campos electromagnéticos externos

Tabla 9.1: Pruebas eléctricas realizadas

Nombre	Descripción	Como se realiza	Resultado
Reconocimiento	Verifica el reconocimiento de las teclas en <a href="#">Windows</a>	Se prueba cada tecla del teclado en <a href="#">Windows</a>	OK
Envío de Teclas	Verifica el envío de teclas desde el teclado en <a href="#">Windows</a>	Se verifica la comunicación <a href="#">USB</a> en <a href="#">Windows</a>	OK
<a href="#">LEDS</a>	Verifica el funcionamiento de los <a href="#">LED</a> en <a href="#">Windows</a>	Se prueba el encendido y apagado de los <a href="#">LED</a>	OK

Tabla 9.2: Pruebas en [Windows](#)

Nombre	Descripción	Como se realiza	Resultado
Reconocimiento	Verifica el reconocimiento de las teclas en <a href="#">Linux</a>	Se prueba cada tecla del teclado en <a href="#">Linux</a>	OK
Envío de Teclas	Verifica el envío de teclas desde el teclado en <a href="#">Linux</a>	Se verifica la comunicación <a href="#">USB</a> en <a href="#">Linux</a>	OK
<a href="#">LEDs</a>	Verifica el funcionamiento de los <a href="#">LED</a> en <a href="#">Linux</a>	Se prueba el encendido y apagado de los <a href="#">LED</a>	OK

Tabla 9.3: Pruebas en [Linux](#)



# Capítulo 10

## Mejoras

### 10.1. Posibles mejoras

#### 10.1.1. Software

Dentro de las posibles mejoras a nivel de software se podría considerar la implementación de nuevas funcionalidades o la optimización del [firmware](#) del teclado. Por ejemplo, se podría agregar soporte para macros programables, configuración de teclas multimedia adicionales. Además, se podría explorar la posibilidad de desarrollar software complementario o driver que permita una personalización más avanzada del teclado, como la asignación de funciones específicas a cada tecla o la creación de perfiles de usuario personalizados desde una interfaz gráfica. Todo esto con el objetivo de mejorar la experiencia de usuario y la versatilidad del teclado.

También se podría considerar la implementación de un sistema de actualización de [firmware](#) over-the-air (OTA) que permita la actualización del [firmware](#) del teclado de forma inalámbrica, sin necesidad de conectarlo a un computador. Esto facilitaría la corrección de errores, la adición de nuevas funcionalidades y la mejora de la seguridad del teclado.

Otra posible mejora sería la implementación de un sistema de detección de fallos y errores en el teclado, que permita identificar y notificar al usuario sobre posibles problemas en el funcionamiento del teclado, como teclas atascadas, errores de conexión, entre otros. Esto permitiría una mejor experiencia de usuario y una mayor confiabilidad del teclado.

Una mejora para la pantalla sería la implementación de un sistema de brillo automático que ajuste el brillo de la pantalla de acuerdo a las condiciones de iluminación del entorno, lo que permitiría una mejor visibilidad de la información mostrada en la pantalla y una reducción del consumo de energía. Además de poder mostrar información adicional como notificaciones

de mensajes, correos electrónicos, entre otros. También se podría añadir un modo de personalización de zonas de la pantalla, para que el usuario pueda elegir qué información desea mostrar en cada zona de la pantalla.

También se podría crear un software o driver que controle la iluminación del teclado, permitiendo al usuario personalizar la iluminación de cada tecla de forma individual, crear efectos de iluminación personalizados y sincronizar la iluminación con otros dispositivos compatibles. Esto permitiría una mayor personalización del teclado y una experiencia de usuario más inmersiva.

### **10.1.2. Hardware**

Se podría explorar la posibilidad de incorporar nuevas características físicas, como una iluminación LED más avanzada con opciones de personalización adicionales o la integración de una pantalla táctil para facilitar el control de funciones adicionales. También se podría considerar la implementación de un sistema de carga inalámbrica para la batería del teclado, lo que permitiría una mayor comodidad y versatilidad en el uso del teclado.

Además, se podría explorar la posibilidad de integrar un sistema de reconocimiento de huellas dactilares en el teclado, que permita una mayor seguridad en el acceso al dispositivo y la autenticación de usuarios. Esto permitiría poder usar el teclado para desbloquear el computador o acceder a aplicaciones y servicios de forma segura.

Una mejora que he intentado implementar en el teclado es que la opción de que este use [switches](#) ópticos, pero no he podido encontrar los componentes necesarios para poder hacer esta implementación. Actualmente, estoy buscando los componentes necesarios para poder hacer esta tarea. Los [switches](#) ópticos son más duraderos que los [switches](#) mecánicos y no sufren de problemas de doble pulsación. Además, los [switches](#) ópticos son más rápidos que los [switches](#) mecánicos, ya que no tienen partes mecánicas que se muevan y, por tanto, no tienen un tiempo de respuesta tan alto como los [switches](#) mecánicos.

### **10.1.3. Materiales**

En cuanto a los materiales utilizados en la fabricación del teclado, se podría considerar la utilización de materiales más resistentes y duraderos, como el aluminio o el acero inoxidable, que permitan una mayor durabilidad y resistencia del teclado ante el uso diario. También se podría explorar la posibilidad de utilizar materiales reciclados o biodegradables en la fabricación del teclado, con el objetivo de reducir el impacto ambiental de su producción y promover la sostenibilidad.

## 10.2. Consideraciones Personales

Durante todas las versiones que he estado desarrollando del teclado, he ido añadiendo funcionalidades que me gustaría tener en un teclado, como la pantalla [OLED](#), la iluminación RGB, la batería recargable, entre otras. Sin embargo, hay algunas funcionalidades que me gustaría añadir en futuras versiones del teclado, como la posibilidad de personalizar la iluminación de cada tecla de forma individual, la implementación de un sistema de carga inalámbrica y la adición de un sistema de actualización de [firmware](#) over-the-air (OTA).

Realmente en cuanto a materiales siempre he intentado conseguir el máximo con lo mínimo. He buscado los materiales más bonitos y resistentes que he podido encontrar a un precio aceptable y que no sean muy complicados de trabajar. He probado con diferentes tipos de madera, diferentes tipos de plásticos, diferentes tipos de pinturas, etc. Siempre he tenido en mente que este teclado sea un producto que pueda durar de por vida.

Aunque hay todavía cuestiones por solventar, como las piezas que no fabrico yo. Como los interruptores, que normalmente son los primeros en fallar. Durante el desarrollo del teclado se me han ocurrido una opción de desarrollar unos [switches](#) propios, pero eso ya es un proyecto para el futuro. Estos [switches](#) serían de tipo óptico y en vez de usar un muelle o resorte para devolver la tecla a su posición original, usarían una columna de imanes de neodimio en una configuración de Halbach para que no afectaran a los demás [switches](#). Estos nos tendrían nada mecánico que sufriese un desgaste y, por tanto, teniendo en cuenta que el material del [switches](#) sería de un material resistente, estos [switches](#) podrían durar toda la vida.

Para conseguir un teclado que pueda pasar de generación en generación, se tendría que tener en cuenta que el teclado sea actualizable, que se puedan cambiar las piezas que se desgasten y que se puedan actualizar las funcionalidades del teclado. Para ello tendríamos que tener en cuenta que el teclado sea modular, que se puedan cambiar las piezas de forma sencilla y que se puedan añadir nuevas funcionalidades de forma sencilla. Y al fin y al cabo que el teclado sea bonito y agradable de usar. Ese es el objetivo que me propuse al principio de este proyecto y que espero poder conseguir en futuras versiones del teclado.



## Capítulo 11

# Conclusiones

Aunque este proyecto empezase hace unos meses las primeras versiones de este empezaron hace años. Desde que empecé a interesarme por la programación y la electrónica, siempre he querido hacer un teclado por las razones que expuse en mi motivación. Por eso, aunque no haya sido un proyecto fácil, ha sido un proyecto muy gratificante que me ha enseñado mucho. Más aún cuando con cada versión me doy cuenta de que puedo hacer un teclado mejor y que dentro de un tiempo, pueda mirar este proyecto, tomar lo que he aprendido y las posibles mejoras que propuse y volver a empezar de nuevo el viaje.

Durante el proceso de diseño y desarrollo, se han enfrentado varios desafíos, se han tomado decisiones importantes para garantizar la calidad y el rendimiento del teclado. Se han realizado pruebas exhaustivas para verificar la funcionalidad y la compatibilidad del teclado en diferentes entornos, lo que ha permitido identificar áreas de mejora y optimización. Uno de los aspectos más destacados de este proyecto ha sido la oportunidad de aplicar conocimientos teóricos aprendidos en un entorno práctico.

En cuanto a los resultados obtenidos, el teclado diseñado ha demostrado ser funcional, robusto y altamente adaptable a las necesidades del usuario. Aunque todavía hay mucho margen para mejorar, el teclado ha demostrado ser una alternativa viable a los teclados convencionales y ofrece una serie de ventajas y características únicas que lo hacen atractivo para una amplia gama de usuarios.

En conclusión, este proyecto ha sido una experiencia enriquecedora que ha permitido aplicar conocimientos teóricos en un contexto práctico y desarrollar habilidades técnicas y profesionales. El teclado diseñado representa un avance significativo en mi camino hacia alcanzar la excelencia en estos campos y me ha motivado a seguir explorando nuevas oportunidades y desafíos en el futuro. El proyecto de los teclados mecánicos personalizados ha

sido un largo camino durante estos años; pero en cada avance, en cada progreso, siento la necesidad de volver a comenzar de nuevo para mejorar la versión anterior.

### **11.1. Trabajo futuro**

Con este proyecto terminado ya estoy pensando en futuras versiones del teclado. Aunque este ya tiene muchas funcionalidades que me gustan tener en un dispositivo, hay algunas funcionalidades que me gustaría añadir en futuras versiones. Algunas de las mejoras que me gustaría añadir en futuras versiones del teclado, para que este verdaderamente sea una obra de ingeniería y diseño, son los [Switches](#) ópticos y magnéticos, la implementación de un sistema de carga inalámbrica, la adición de un sistema de actualización de [firmware](#) over-the-air (OTA), desarrollar un driver que permita personalizar el teclado de forma gráfica desde un programa en el ordenador y la implementación de módulos complejos como diccionarios de palabras, calculadoras o control de dispositivos [IoT](#).

# Capítulo 12

## Apéndice

En esta sección, se pueden encontrar diversos tipos de contenido, tales como datos brutos, códigos fuente, resultados detallados, diagramas extensivos, y cualquier otra documentación que, aunque esencial, podría interrumpir el flujo de lectura si se incluyera en el cuerpo principal del texto. El objetivo del apéndice es facilitar al lector interesado un acceso directo a estos materiales, permitiéndole profundizar en los aspectos técnicos o metodológicos del proyecto sin sobrecargar el texto principal.

### 12.1. Circuitos

Los circuitos son una parte fundamental de cualquier proyecto de electrónico, y su correcto diseño y funcionamiento son esenciales para el éxito de la implementación. En esta sección, se incluyen los esquemáticos de los circuitos utilizados en el proyecto, así como cualquier información adicional relevante para su comprensión y reproducción.

Todos los circuitos que componen el proyecto se han originado de versiones anteriores de los mismos, y han sido adaptados y mejorados para cumplir con los requisitos específicos de la presente implementación.

#### 12.1.1. Esp32S3 con Agujero

Dado que el componente ESP32S3 es un componente de montaje superficial y este posee una parte inferior con los [pads](#) de soldadura a tierra totalmente debajo del componente, se ha decidido modificar el componente (Parte de Eagle) para que sea más sencillo de soldar. Para ello, se ha añadido un agujero en la parte inferior del componente para que los [pads](#) de soldadura sean visibles desde el otro lado de la placa.

Para ello vamos a ir a Eagle e importaremos la librería de ESP32S3. iremos a editar el **footprint** y donde están los terminales inferiores en mitad del dispositivo vamos a sustituirlos por un agujero de soldadura de las mismas dimensiones.

Todos los pasos han sido realizados siguiendo el tutorial descrito en la asignatura de Circuitos Impresos. Una descripción breve de los pasos a seguir es la siguiente.

**1. Abrir o Crear una Librería.**

Abra EAGLE y cree una nueva librería con el nombre deseado (en nuestro caso, “ESP32S3HOLE”).

**2. Seleccionar y Nombrar el Device.**

Seleccione “Device” y asígnale un nombre (por ejemplo, “ESP32S3HOLE”).

**3. Crear el Símbolo del Componente.**

Dibuje el símbolo del componente y añada los pines necesarios, asignando nombres a cada pin según corresponda. Trace el diseño del símbolo de acuerdo con sus especificaciones. En nuestro caso podemos encontrar las dimensiones, pines y demás en la hoja de datos del componente ESP32S3 [16].

**4. Usar un Package Existente.**

Para ello, abra la librería en edición, vaya a la librería en el menú principal, elija el componente, haga clic con el botón derecho y seleccione la opción “Copiar a Librería”.

**5. Editar el **footprint**.**

Abra la librería en edición, seleccione el componente, haga clic con el botón derecho y seleccione la opción “Edit Package”. A continuación, edite el **footprint** según sus especificaciones. Este paso es el más importante, ya que aquí es donde se añadirá el agujero de soldadura. Para ello, seleccione los pines inferiores del componente y sustítúyalos por un agujero de soldadura de las mismas dimensiones. Asegúrese de que el agujero de soldadura esté correctamente alineado con los pines del componente. Ahora que el **footprint** ha sido modificado, guarde los cambios y cierre la ventana de edición.

**6. Unir Pines.**

Para unir los pines del símbolo del componente con los pines del **footprint**, seleccione el componente en la librería en edición, haga clic con el botón derecho y seleccione la opción “Connect Package”.

### 7. Añadir Atributos.

Añada los atributos necesarios al componente, como el valor, la descripción, etc.

De forma que quedará como podemos ver en la figura 12.1.1.

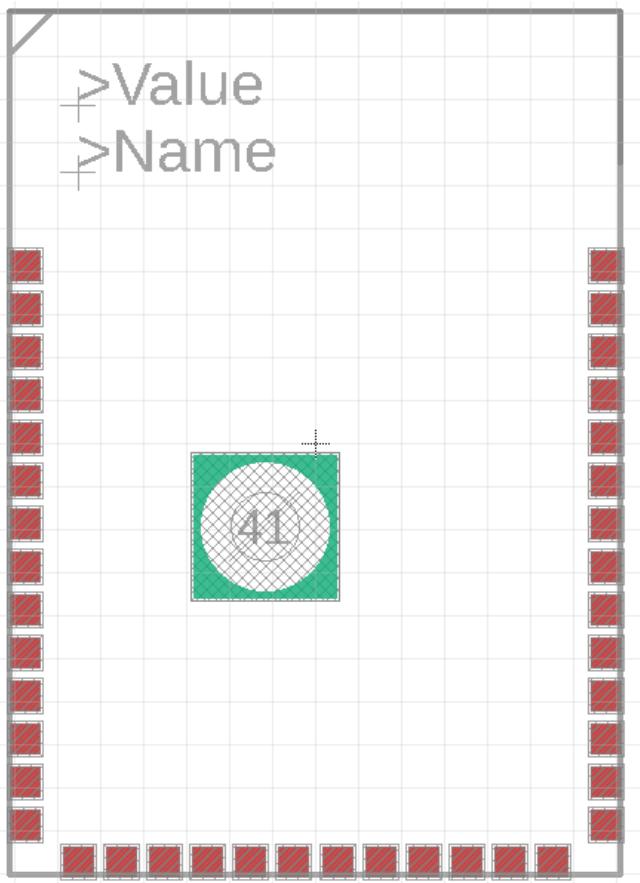


Figura 12.1: Imagen de la ESP32S3 modificada

## 12.2. PCB

### 12.2.1. Dimensiones físicas

Teniendo en cuenta que el proyecto se va a realizar en una placa de madera, y esta debe contener todos los elementos, y dicha placa va a ser fresada con una fresadora CNC, se ha decidido dejar márgenes de 1 mm en cada lado de la placa. Además vamos a añadir un margen extra a la [PCB](#) para que pueda ser atornillada más fácilmente.

En la figura 12.2 podemos ver en rojo el borde de la **PCB** y la línea más próxima, en negro, la madera. Se ha dejado medio milímetro de margen para las máquinas **CNC** y de fabricación de **PCB**. Los marcadores de los interruptores generados por el software de la sección 5.1.1 se ha dejado en verde.

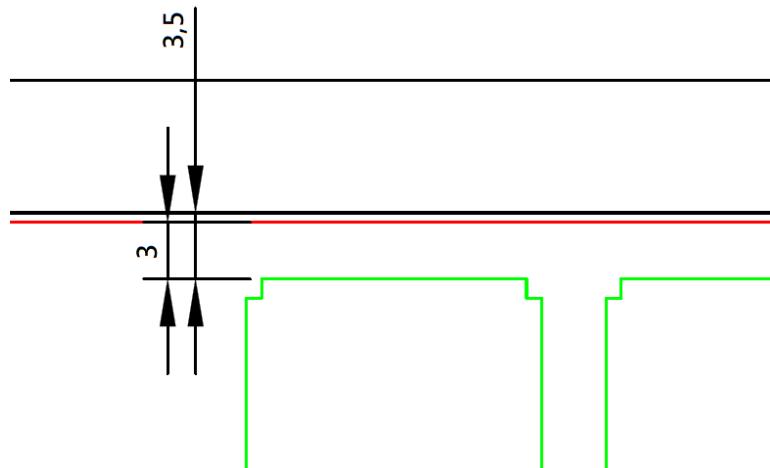


Figura 12.2: Imagen del plano acotado del espacio entre las **keycaps** y la madera.

La idea siempre ha sido ajustar las dimensiones de la carcasa y **PCB** para que no se pudiera ver la **PCB** una vez que el teclado tuviese las **keycaps** montadas. Esto gracias a que la **plate** no dejaría mucho hueco para ver la **PCB**, que se vería peor estéticamente. Por eso, uno de los objetivos es intentar que la carcasa esté lo más cerca de las **keycaps**.

## 12.3. Programación

### 12.3.1. Código fuente

Todo el código fuente del proyecto se encuentra disponible en el repositorio de GitHub del proyecto [13]. En este repositorio, se pueden encontrar todos los archivos necesarios para la implementación del proyecto, incluyendo el código fuente, los esquemáticos, las librerías, y cualquier otra documentación relevante. Este GitHub tiene todo el contenido que se ha necesitado para la realización del proyecto. Cualquier persona que quiera realizar el proyecto puede descargarse el código y los esquemáticos y realizarlo sin problemas. Además de servirse de la documentación que se ha ido generando a lo largo del proyecto para su montaje y fabricación.

### 12.3.2. Estructuras

En esta sección se describen las estructuras de datos y funciones más relevantes utilizadas en el proyecto. Estas van desde la configuración de la EEPROM, hasta la configuración de los distintos modos de funcionamiento del teclado. Las variables de estado o las clases principales que se han utilizado en el proyecto, así como donde poder encontrar estas estructuras en el código fuente.

#### Variables de estado

Estas variables son las que se han utilizado para saber el estado del teclado. Estas se han ido creando a lo largo del proyecto y se han ido añadiendo a medida que se iban necesitando. Estas variables se encuentran en el archivo de ModernWood/ModernWood/include/ModernWood.h. Un ejemplo de estas variables se puede ver en el código 12.1.

```
// Variable to switch between keyboard and display mode
extern bool volatile WorkingAsKeyboard;
extern bool volatile interrupted_FN;

// Variables to know the state of the keyboard
extern bool displayChanged; //Main display changed (to
update the display in the main loop)
extern bool volatile isUSBConnected;
extern bool volatile isBLEConnected;
extern bool isBluetoothOn;
extern bool connectionChanged;

// Battery energy save mode variables
extern bool goingToSleep;
```

```

extern bool Sleeping;
extern bool timerSetupDone;

//Battery level (need to update the display when it changes
)
extern bool batteryLevelChanged;

extern bool inExternalFunctionMode; //To know if we are in
the external function mode
extern bool executingCustomFunction; //To know if we are
executing a custom function
//If there is a actual function begin executed, which is
the row and col of the key
extern int actualFunctionRow;
extern int actualFunctionCol;

//To know if we are in the menu and which option is
selected
extern int option_selected;
extern int option_selected_submenu;

```

Código 12.1: Variables de estado del teclado

### Configuración EEPROM

Una de las estructuras más importantes que se han utilizado en el proyecto es la estructura de la configuración a guardar en la memoria [EEPROM](#). Esta se compone de un enumerado con los distintos tipos de configuraciones que se pueden guardar por submenú, un [string](#) con el nombre de la opción del submenú, un [string](#) con el código de la variable, el tipo de la variable y vector de punteros a las variables que se van a guardar en la [EEPROM](#). Todas de tipo [int](#). Posteriormente, en la sección de funciones 12.3.3 se explicará como se interpretan estos enteros ([int](#)) para formar todo tipo de variables.

Estos struct se encuentran en el archivo de ModernWood/ModernWood/include/ModernWood.h. Uno de estos se puede ver en el código 12.4.

```

enum SubMenuConfig {
    _EnableDisplayOption,
    _EnableKeyboardOption,
    _DebounceOption,
    _LanguageMenuOption,
    SizeSubMenuConfig};

const String SubMenuConfigText[SizeSubMenuConfig] = {
    "Enable Display",
    "Enable Keyboard",
    "Debounce (ms)",
    "Language"};

```

```

const String SubMenuConfigKeys[SizeSubMenuConfig] = {
    "ENADIS",
    "ENAKEY",
    "DEBNCE",
    "LANGUA"};
```

```

const String SubMenuConfigVarType[SizeSubMenuConfig] = {
    "bool",
    "bool",
    "int",
    "language"};
```

```

extern int* SubMenuConfigVar[SizeSubMenuConfig];
```

Código 12.2: Primer Struct de configuración del teclado

### Clase RGB

Como se ha comentado en la sección de configuración de la [EEPROM](#) 12.3.2, todas las variables que se guardan en la [EEPROM](#) son de tipo [int](#). Para poder guardar los colores de los leds RGB se ha creado una clase RGB. Esta clase se encarga de guardar los colores en formato HSV y RGB y empaquetado en [int](#). Así como calcular el color en función de los valores de HSV y del empaquetado. Esta clase se encuentra en el archivo de ModernWood/-ModernWood//include/customRGB.h. Este archivo también es público en el repositorio del proyecto [13]. La definición de los 3 tipos de color se puede ver en el código 12.3.

```

int r, g, b; // RGB
int color;   // 16-bit color packed (For EEPROM)
float h, s, v; // HSV HUE, SATURATION, VALUE
```

Código 12.3: Definicion de los 3 datos que maneja la clase RGB

### 12.3.3. Funciones

En esta sección se describen las funciones más relevantes utilizadas en el proyecto. Estas funciones se pueden encontrar en el archivo de ModernWood/ModernWood/include/ModernWood.h y su descripción en el archivo de ModernWood/ModernWood/src/ModernWood.cpp. Estas van desde la configuración de la [EEPROM](#), hasta la configuración de los distintos modos de funcionamiento del teclado. Las variables de estado o las clases principales que se han utilizado en el proyecto. Así como donde poder encontrar estas estructuras en el código fuente.

### Modos de funcionamiento

Para el teclado tenemos 3 modos de funcionamiento. Estos son el modo teclado (Se compone de teclado por [USB](#) y [Bluetooth](#)), el modo pantalla/configuración, donde se pueden cambiar las opciones del teclado y el modo especial (Se compone de la pantalla y el teclado). Este modo es el que ejecuta el código de las macros o funciones especiales. Para cambiar de modo se usa la tecla FN.

```
void WorkingInExternalFunctionMode(
    TFT_eSPI &tft,
    BleKeyboard &bleKeyboard,
    USBHIDKeyboard &Keyboard,
    bool volatile &isBLEConnected,
    bool volatile &isUSBConnected);

void WorkingModeKeyboard(
    TFT_eSPI &tft,
    BleKeyboard &bleKeyboard,
    USBHIDKeyboard &Keyboard,
    bool volatile &isBLEConnected,
    bool volatile &isUSBConnected);

void WorkingModeDisplay(
    TFT_eSPI &tft,
    BleKeyboard &bleKeyboard,
    USBHIDKeyboard &Keyboard,
    bool volatile &isBLEConnected,
    bool volatile &isUSBConnected);
```

Código 12.4: Funciones principales del teclado ModernWood

### Función de cambio de configuración

En el apartado 12.3.2 se ha hablado de la estructura de configuración de la [EEPROM](#). En este apartado se va a hablar de la función que se ha creado para cambiar la configuración de la [EEPROM](#). Complementariamente, se ha creado otra función para poder alterar el tipo de dato. Ya que como se mencionó, todas las variables son enteros para simplificar el guardado y lectura de estas variables. Estas funciones se encuentran en el archivo de ModernWood/ModernWood/src/ModernWood.cpp. Las funciones se puede ver en el código 12.5.

```
void ChangeVar(String varType, int *var, bool &
changed_option_subMenu, bool right)
{
    // Bool variable: True to False and False to True
    if (varType == "bool")
```

```
{  
    *var = !*var;  
    changed_option_subMenu = true;  
}  
  
// Int variable: Increase the value  
else if (varType == "int")  
{  
    if (right)  
    {  
        *var = modulo_p((*var + 1), 101);  
    }  
    else  
    {  
        *var = modulo_p((*var - 1), 101);  
    }  
    changed_option_subMenu = true;  
}  
  
// RGB variable: Increase the value  
else if (varType == "rgb")  
{  
    if (right)  
    {  
        // Using HSV  
        LedsColor.h += 0.01;  
        if (LedsColor.h >= 1.0)  
        {  
            LedsColor.h -= 1.0;  
        }  
        LedsColor.hsvToRgb();  
        LedsColor.CalculateColor();  
    }  
    else  
    {  
        // Using HSV  
        LedsColor.h -= 0.01;  
        if (LedsColor.h <= -1.0)  
        {  
            LedsColor.h += 1.0;  
        }  
        LedsColor.hsvToRgb();  
        LedsColor.CalculateColor();  
    }  
    changed_option_subMenu = true;  
}  
  
// language variable : Change the language  
else if (varType == "language")  
{  
    if (right)  
    {  
        *var = modulo_p((*var + 1), 2);  
    }  
}
```

```

        else
        {
            *var = modulo_p(*var - 1), 2);
        }
        changed_option_subMenu = true;
    }
}

String varToText(String varType, int *var)
{
    String ret = "";
    // Check the type of the variable it can be "bool" or "int" or "rgb" or "none"
    if (varType == "bool")
    {
        // Check if var is true or false
        if (*var)
        {
            ret = "True";
        }
        else
        {
            ret = "False";
        }
    }
    else if (varType == "int")
    {
        ret = String(*var);
    }
    else if (varType == "rgb")
    {
        RGB temp;
        temp.color = *var;
        temp.CalculateRGB();
        ret = String(temp.r) + "," + String(temp.g) + "," +
String(temp.b);
    }
    else if (varType == "none")
    {
        ret = "+";
    }
    else if (varType == "language")
    {
        if (*var == 0)
        {
            ret = "ENG";
        }
        else
        {
            ret = "ES";
        }
    }
}

```

```
    return ret;
}
```

Código 12.5: Funciones para mostrar y editar las variables de le EEPROM

Si se desea ver el resto de funciones menos relevantes, se puede acceder al repositorio del proyecto [13]. Todas las funciones están comentadas y se puede ver su funcionamiento en el archivo de ModernWood/ModernWood/include/ModernWood.h.

### 12.3.4. PID y VID

El **PID** y **VID** son los identificadores de producto y de vendedor respectivamente. Estos identificadores son necesarios para que el sistema operativo pueda identificar el dispositivo conectado. En el caso de los teclados, estos identificadores son necesarios para que el sistema operativo pueda cargar el controlador adecuado y permitir la comunicación entre el teclado y el ordenador. Aunque estos identificadores son asignados por el fabricante, es posible modificarlos para adaptarlos a las necesidades específicas del proyecto. Dado que se quiere que el teclado sea reconocido como un teclado normal, pero con un fabricante y producto específico, se ha decidido iniciar la comunicación con el ordenador mediante el **USB** con los identificadores de producto y vendedor de un teclado estándar, pero indicando que el fabricante se cambie a uno nuevo y el producto sea el ModernWood. Estos identificadores se pueden ver en el código 12.6.

```
VID = 0x2001
PID = 0x1111
```

Código 12.6: Números elegidos para el PID/VID

Una vez realizado esto, a la hora de emparejar el teclado con el ordenador, éste se reconocerá como un teclado normal pero con un nombre de producto ModernWood. De esta forma se podrá utilizar el teclado sin problemas en cualquier ordenador. Se puede ver el nombre después de la conexión en la figura 12.3.



Figura 12.3: Imagen del teclado ModernWood conectado a un ordenador en el apartado de dispositivos bluetooth.

### 12.3.5. Leds

En cuanto a los leds se han añadido varios modos de funcionamiento. Estos se pueden cambiar en la opción del menú 2 en su opción correspondiente. Se pueden ampliar estos modos de funcionamiento añadiendo más funciones en la sección del main loop 12.7. Un ejemplo de una función de arcoíris se puede ver en el código 12.8.

```
// Check if the leds are enabled
if (*SubMenuLedsVar[_EnableLeds] != 0 && !Sleeping)
{
    // Check for special functions like led control
    if (*SubMenuLedsVar[_LedsMode] != 0)
    {
        // Check if the led mode is white
        if (*SubMenuLedsVar[_LedsMode] == 1)
        {
            // Show white color
            float brightness = (*SubMenuBrightnessVar[_BrightnessLeds] / 100.0f);
            for (int i = 0; i < NUMBER_OF_LEDS; i++)
            {
                RgbLED.setPixelColor(i, (int)(255 * brightness),
                                      (int)(255 * brightness),
                                      (int)(255 * brightness));
            }
        }
        if (*SubMenuLedsVar[_EnableLeds] == 0)
        {
            RgbLED.clear();
        }
        RgbLED.show();
    }
    // Check if the led mode is rainbow
    else if (*SubMenuLedsVar[_LedsMode] == 2)
    {
        // Check if the led mode is rainbow
        rainbowEffect(RgbLED);
    }
    // Here we can add more modes
    // else if (*SubMenuLedsVar[_LedsMode] == 3) {}
}
}
```

Código 12.7: Funciones de los modos de los Leds en el main.cpp

```
void rainbowEffect(Adafruit_NeoPixel &leds)
{
    static unsigned long lastUpdate = 0;
```

```

        unsigned long now = millis();
        float hueStep = 1.0 / NUMBER_OF_LEDS;
        float speed = LedsSpeed / 50000.0;           // Ajusta la
velocidad
        float brightness = LedsBrightness / 100.0; // Ajusta el
brillo

        if (now - lastUpdate > 20)
        { // Actualizar cada 20ms
            lastUpdate = now;
            for (int i = 0; i < leds.numPixels(); i++)
            {
                float hue = fmod(now * speed + i * hueStep,
1.0);
                RGB color(hue, 1.0, brightness); // Hue,
Saturation, Brightness
                leds.setPixelColor(i, leds.Color(color.r, color
.g, color.b));
            }
            leds.show();
        }
    }
}

```

Código 12.8: Ejemplo de función de arcoíris en los leds en el ModernWood.cpp

### 12.3.6. Macros

Una de las ideas principales para el teclado es que al tener una pantalla, un teclado y bluetooth, el usuario pudiese programar funciones extra para el teclado. Estas funciones pueden ir desde pulsaciones de teclas simples a videojuegos en la pantalla de microcontrolador. Para ello se diseñó un modo especial que se activa desde la opción de ayuda del menú. Este modo especial efectuaría la función asignada en la tecla que el usuario haya deseado. Para ello se ha creado una estructura de datos que guarda el puntero a la función de entrada y la fila y columna de la tecla que la activa. Esta estructura se puede ver en el código 12.9. Y un ejemplo de asignación de la función vacía (base.h 12.11) en el código 12.10.

También se ha proporcionado una estructura de archivos para facilitar la creación de estas funciones. Estos archivos se pueden ver en el repositorio del proyecto [13]. Y los archivos son ModernWood/modules/src y ModernWood/modules/include.

```

#pragma once

#include <ModernWood.h>

```

```
//All modules headers must be included here
#include "Base.h"

//Array of function pointers to the functions that
//will be called when the key is pressed (In which
//module should be activated. Note that the function
//will be executed in a loop once the key is pressed.
//So, the exit condition must be implemented in the
//function itself (if needed). Fn Key will interrupt
//the loop and stop the function so this key MUST NOT
//be used.

//All the functions has to be declared as "void foo(void)
//{
extern void (
    *MODULESFUNCARRAY [KEYBOARDHEIGHT]
    [KEYBOARDWIDTH])(void);
```

Código 12.9: Estructura que contiene las funciones en Modulesmap.h

```
#include "ModulesMap.h"

//BASE MODULE
#define FN_NONE noneFN
#define FN_BASE ModuleGetVersion

//...
```

Código 12.10: Ejemplo de asignación a una tecla en el archivo ModernWood/modules/src/ModulesMap.cpp

```
#include "Base.h"

//None Module Function (00 Optimization due to do
optimization 0)
void __attribute__((optimize("00")) ) noneFN(void){
    //Just Exit Module
    //All modules must have this function to end the
    //module and return to the main menu (executed one time
)
    //If you don't have this function, the module will be
stuck
    //in a loop until FN KEY is pressed or the board is
reset
    exitModule();
}
```

Código 12.11: La función base vacía que no hace nada a forma de ejemplo. En base.h y base.cpp

## 12.4. Pruebas

Durante el desarrollo del proyecto se han realizado diversas pruebas para verificar el correcto funcionamiento de los distintos componentes y módulos utilizados. Estas pruebas han sido fundamentales para identificar y corregir errores, así como para garantizar la calidad y fiabilidad del sistema final. En esta sección se incluyen los detalles de las pruebas realizadas, así como los resultados obtenidos y las conclusiones derivadas de los mismos.

Durante todo el desarrollo se ha ido programando y probando el código en la placa de desarrollo ESP32S3. Se han realizado pruebas de los distintos componentes. El principal módulo que se ha estado probando ha sido el módulo de la pantalla **OLED** y el Multiplexor. La pantalla ha sido lo que más tiempo ha estado en desarrollo, ya que había que hacer que funcionase con todos los estados del teclado. Así mismo, esta tenía que ir cambiando de estado según el estado del teclado. Por lo que cualquier error en las coordenadas o en la escritura de la pantalla se notaba fácilmente y se podía corregir.

Para poder realizar la tarea de actualizar la pantalla se han ido ideando una serie de variables a lo largo del código que nos indicaría cómo está el estado del teclado; así podríamos, desde cualquier parte del código, saber en qué estado se encuentra el teclado y poder actualizar la pantalla en consecuencia.

Dado que el entorno que teníamos para las pruebas no constaba de un teclado normal se decidió crear unas funciones de **DEBUG** que nos permitirían, mediante el serial de la placa de desarrollo, simular la pulsación de las teclas. De esta forma podríamos probar el funcionamiento de la pantalla sin necesidad de tener un teclado conectado así como el funcionamiento de la batería, el bluetooth, etc. y poder ir cambiando las variables de estado del teclado.

Como podemos ver en el código 12.12 esta es la solución por la que se ha optado y también se puede encontrar en el GitHub del proyecto [13].

```
#ifdef DEBUG
    // For debug purposes show how many times the loop is
    // executed per second
    loop_counter++;
    if (millis() - last_loop_time > 1000)
    {
        Serial.print("Loop executed ");
        Serial.print(loop_counter);
        Serial.println(" times per second");

        Serial.print("Temperature: ");
        float result = 0;
```

```

        temp_sensor_read_celsius(&result);
        Serial.print(result);
        Serial.println(" C");

        loop_counter = 0;
        last_loop_time = millis();

        secs++;
        Serial.print("Seconds: ");
        Serial.println(secs);
    }

    // Read from serial
    char c = 'N';
    if (Serial.available())
    {
        c = Serial.read();
        Serial.print("Read from serial: ");
        Serial.println(c);
    }
    // wasd -> ARRIBA ABAJO DERECHA IZQUIERDA
    // Espace -> Enter
    // E -> Escape
    // Q -> MODO ESPECIAL
    switch (c)
    {
        case 'Q':
            WorkingAsKeyboard = !WorkingAsKeyboard;
            interrupted_FN = true;
            break;

        case 'E':
            MenuPressed[ArrEsc] = true;
            break;

        case ' ':
            MenuPressed[ArrEnter] = true;
            break;

        case 'W':
            MenuPressed[ArrUp] = true;
            break;

        case 'S':
            MenuPressed[ArrDown] = true;
            MenuPressed[ArrDown] = true;
            break;

        case 'A':
            MenuPressed[ArrLeft] = true;
            break;

        case 'D':
            MenuPressed[ArrRight] = true;
    }
}

```

```

        break;

    case '1':
        batteryLevel--;
        batteryLevelChanged = true;
        break;

    case '2':
        batteryLevel++;
        batteryLevelChanged = true;
        break;

    case '3':
        connectionChanged = true;
        isUSBPreferred = !isUSBPreferred;
        isBLEPreferred = !isBLEPreferred;
        break;

    case '4':
        connectionChanged = true;
        isBLEConnected = !isBLEConnected;
        break;

    default:
        break;
}

#endif

```

Código 12.12: Código de debug para la placa de desarrollo

## 12.5. Ensamblaje

El ensamblaje del teclado ha sido una tarea compleja debido a la integración de múltiples componentes. A pesar de tener en cuenta posibles errores, se cometieron algunos debido al desconocimiento o la falta de experiencia. Estos errores han requerido la modificación de componentes y ajustes en el diseño. En esta sección se detallan el proceso de ensamblaje del teclado, los errores cometidos y las soluciones adoptadas para corregirlos.

### 12.5.1. Errores

El único error cometido en el ensamblaje ha sido el del montaje en la carcasa. Los tornillos hembra/macho, que iban a ser atornillado a la carcasa de madera, eran demasiado grandes. Por esta razón se optó por pegar estos tornillos a la carcasa con epoxi en el agujero correspondiente al tornillo. Así se podría desmontar la placa de madera del teclado sin problemas. Estos

mismos agujeros tuvieron que ser ampliados para poder meter la tuerca sin dificultad. Estos agujeros pasaron de 5 mm a 6 mm.

### **12.5.2. Montaje Nuevo**

El montaje del teclado está descrito en la sección 8.3 de este documento. En esta sección se puede ver el montaje detallado del teclado. Describiéndose paso a paso como se ha montado el teclado. Ahora, con los cambios realizados en la carcasa, el montaje cambia en uno de los puntos. En ese mismo punto, en la sucesión de tornillos, 8.3, se ha cambiado el procedimiento de montaje, en vez de atornillar los tornillos a la carcasa, se han de pegar con epoxi sin necesidad de atornillarlos. (Los errores de dimensiones se prevén corregidos para la versión final en sus planos).

# Glosario

**API** Una API (Interfaz de Programación de Aplicaciones) es un conjunto de definiciones y protocolos que permiten a los distintos componentes de software comunicarse entre sí. Proporciona una interfaz estandarizada para la interacción entre aplicaciones, permitiendo a los desarrolladores acceder a funciones específicas o datos de un software sin necesidad de conocer los detalles internos de su implementación. Las APIs son ampliamente utilizadas en el desarrollo de software para integrar servicios y funcionalidades de diferentes aplicaciones de manera eficiente y coherente.

**Arduino** Es una plataforma de hardware y software de código abierto diseñada para facilitar el desarrollo de proyectos electrónicos interactivos. Consiste en una placa de circuito impreso con un microcontrolador y un entorno de desarrollo integrado (IDE) que simplifica la programación y la interacción con los componentes electrónicos. Arduino es ampliamente utilizado por aficionados, estudiantes y profesionales para crear una amplia variedad de dispositivos y sistemas, desde simples proyectos de bricolaje hasta complejas aplicaciones de automatización y robótica.

**Back Feed Current** Corriente de retroalimentación, que se refiere a la corriente que fluye en sentido contrario a través de un dispositivo o circuito, lo que puede causar daños o interferencias en el sistema.

**Bluetooth** Una tecnología de comunicación inalámbrica de corto alcance que permite la transmisión de datos entre dispositivos electrónicos. El Bluetooth se utiliza comúnmente para la conexión de dispositivos como auriculares, altavoces, teclados y ratones sin necesidad de cables.

**CNC** Son las siglas en inglés de "Control Numérico por Computadora" (Computer Numerical Control). Se refiere a un sistema automatizado de control de máquinas herramienta, como fresadoras, tornos y cortadoras láser, mediante el uso de un programa computarizado. Los sistemas CNC son capaces de ejecutar operaciones de mecanizado de

alta precisión basadas en instrucciones digitales, lo que los hace indispensables en la fabricación moderna.

**Controladores** Software que permite la comunicación entre el sistema operativo y el hardware, permitiendo que los dispositivos funcionen correctamente.

**DEBUG** En programación, "DEBUG" se refiere al proceso de identificar y corregir errores o problemas en el código de un programa. El debugging es una parte fundamental del desarrollo de software y se realiza utilizando herramientas y técnicas especializadas para rastrear, analizar y solucionar problemas en el código.

**Deep Sleep** En el contexto de los microcontroladores y sistemas embedidos, "Deep Sleep" (sueño profundo) es un modo de bajo consumo de energía diseñado para minimizar el consumo de energía cuando el dispositivo no está activamente realizando tareas. Durante el sueño profundo, el microcontrolador reduce su consumo de energía al mínimo al apagar la mayoría de sus funciones y circuitos, permitiendo que el dispositivo permanezca en un estado de reposo prolongado hasta que se activa por una interrupción externa, como una señal de temporizador o una entrada de sensor. Esta funcionalidad es fundamental en aplicaciones de bajo consumo de energía, como dispositivos portátiles, sensores remotos y sistemas alimentados por batería, donde se busca maximizar la vida útil de la batería o minimizar la dependencia de fuentes de energía externas.

**DIY** Hace referencia a "Do It Yourself" (Hazlo tú mismo). Se trata de la práctica de crear, construir o reparar cosas por uno mismo, en lugar de comprarlas prefabricadas. Es una filosofía que fomenta la creatividad, la autonomía y la satisfacción personal a través de la realización de proyectos artesanales.

**Dongle** Un dongle es un dispositivo de hardware que se conecta a otro para proporcionar funcionalidad adicional. Comúnmente, se utiliza para referirse a pequeños dispositivos que permiten la conexión inalámbrica o la adaptación de interfaces, como un dongle USB para conectividad Bluetooth.

**EEPROM** Son las siglas en inglés de "Electrically Erasable Programmable Read-Only Memory" (Memoria de sólo lectura programable y borrable eléctricamente). Se trata de un tipo de memoria no volátil que puede ser programada, leída y borrada eléctricamente, lo que la hace ideal para almacenar configuraciones, datos y firmware en dispositivos

electrónicos. La EEPROM es ampliamente utilizada en microcontroladores y sistemas embebidos para almacenar información que debe persistir incluso cuando el dispositivo se apaga o reinicia.

**EMI** Interferencia Electromagnética, que se refiere a la interferencia causada por campos electromagnéticos que pueden afectar el funcionamiento de dispositivos electrónicos.

**Firmware** Software de bajo nivel almacenado en la memoria de hardware que proporciona control básico para los componentes del dispositivo.

**Footprint** Un footprint en el contexto de diseño de PCB se refiere al señalamiento ó ubicación de los pads y otros elementos de conexión en la superficie de la placa de circuito impreso (PCB) para un componente electrónico específico.

**Fresado** Un proceso de fabricación que utiliza una herramienta de corte rotativa para dar forma a materiales como metal o plástico.

**Ghosting** El ghosting se produce cuando el teclado envía solo un comando al PC tras presionar varias teclas a la vez. Por ejemplo, si pulsamos las teclas S + D + F, manda la tecla “S”, por ser la primera pulsada. Otros teclados no mandan ningún comando cuando pulsamos varias a la vez.

**HID** Interfaz Humano-Computadora, un protocolo que permite la comunicación entre dispositivos de entrada, como teclados y ratones, y la computadora.

**Hot-plugging** La capacidad de conectar o desconectar un dispositivo mientras el sistema está en funcionamiento sin necesidad de reiniciar.

**Int** En programación, ”int” es un tipo de dato que se utiliza para representar números enteros, es decir, números sin parte decimal. Dependiendo del lenguaje de programación, el rango de valores que puede almacenar un entero puede variar, pero generalmente se utiliza para representar números enteros positivos y negativos.

**Ion de Litio** Un tipo de tecnología de batería recargable comúnmente utilizada en dispositivos electrónicos debido a su alta densidad de energía.

**IoT** Son las siglas en inglés de ”Internet of Things” (Internet de las cosas). Se refiere a la red de dispositivos físicos que están integrados con sensores, software y otros componentes tecnológicos que les permiten conectarse y intercambiar datos a través de Internet. Estos dispositivos pueden incluir desde electrodomésticos y dispositivos portátiles hasta

vehículos y equipos industriales. La tecnología IoT permite la recopilación, monitorización y control remoto de datos en tiempo real, lo que ofrece una amplia gama de aplicaciones en diversos sectores, como el hogar inteligente, la salud, la agricultura, la industria, la logística y la ciudad inteligente, entre otros.

**ISO** Organización Internacional de Normalización, una entidad que establece estándares para asegurar la calidad y la eficiencia de productos y servicios.

**Keycaps** Se refiere a las tapas individuales de las teclas de un teclado. Esas tapas, a menudo personalizables, pueden tener diferentes diseños, colores o materiales para proporcionar una experiencia de escritura única y estética.

**Latencia** El tiempo que tarda un sistema en responder a una solicitud después de recibirla, a menudo asociado con retrasos en la transmisión de datos.

**LCD** Son las siglas en inglés de "Liquid Crystal Display" (Pantalla de Cristal Líquido). Se trata de una tecnología de visualización que utiliza cristales líquidos entre dos láminas de material polarizado para producir imágenes. Los LCD son comúnmente utilizados en dispositivos como televisores, monitores de computadora, relojes digitales y paneles de instrumentos de vehículos.

**LED** Diodo Emisor de Luz, un dispositivo semiconductor que emite luz cuando una corriente eléctrica pasa a través de él.

**Linux** Un sistema operativo de código abierto basado en el kernel Linux y utilizado en una variedad de dispositivos, desde servidores hasta dispositivos embebidos.

**Multiplexor** Un multiplexor es un dispositivo electrónico que permite seleccionar una de varias señales de entrada y conectarla a una única salida. Es comúnmente utilizado en electrónica digital para reducir el número de líneas de control necesarias para seleccionar entre múltiples fuentes de datos.

**OLED** Son las siglas en inglés de "Organic Light-Emitting Diode" (Diodo Orgánico de Emisión de Luz). Se trata de una tecnología de visualización que utiliza diodos orgánicos para emitir luz y producir imágenes. A diferencia de los LCD, los OLED no requieren retroiluminación, lo que les permite ofrecer colores más vibrantes, negros más profundos y un mejor contraste. Los OLED son utilizados en dispositivos como

teléfonos inteligentes, televisores de alta gama y pantallas de dispositivos portátiles.

**Online** En el contexto de la tecnología y la informática, "Online" se refiere a la condición de estar conectado a Internet o a una red informática. Cuando un dispositivo está "online", puede comunicarse con otros dispositivos o acceder a recursos en la red, como sitios web, servicios en la nube, aplicaciones en línea, etc.

**Pads** Son las áreas metálicas en una placa de circuito impreso (PCB) donde se sueldan los componentes electrónicos.

**PCB** Placa de Circuito Impreso, un componente que proporciona conexiones eléctricas entre varios componentes en dispositivos electrónicos.

**PID** Product ID, es un identificador único asignado a un dispositivo USB para identificarlo de manera única entre otros dispositivos conectados al mismo sistema.

**Plate** En el contexto de los teclados personalizados o mecánicos, un "plate" se refiere a una pieza de material (como metal, plástico o acrílico) que se coloca debajo de las teclas y sobre la placa base del teclado. El plate proporciona rigidez estructural al teclado y determina la disposición física de las teclas. Además de su función estructural, el diseño del plate también puede afectar la sensación táctil y la respuesta de las teclas al ser presionadas, lo que lo convierte en un componente importante para los entusiastas de los teclados personalizados.

**PlatformIO** Es un entorno de desarrollo integrado (IDE) de código abierto para el desarrollo de software embebido y aplicaciones IoT. Proporciona herramientas y funcionalidades para escribir, compilar, depurar y cargar código en una variedad de plataformas de hardware, incluyendo Arduino, ESP8266, ESP32, Raspberry Pi y muchas otras. PlatformIO ofrece una interfaz unificada y fácil de usar que facilita el desarrollo y la gestión de proyectos de hardware y software en múltiples plataformas, lo que lo convierte en una opción popular entre los desarrolladores de sistemas embebidos y de IoT.

**Plug-and-Play** La capacidad de un sistema para reconocer automáticamente e instalar dispositivos sin intervención del usuario.

**Polling** El polling rate de un teclado se refiere a la frecuencia con la que el teclado envía información al dispositivo al que está conectado. Es medida en hercios (Hz) y representa cuántas veces por segundo el teclado actualiza su estado y envía los datos correspondientes al dispositivo.

Un polling rate más alto significa una respuesta más rápida del teclado, lo que puede resultar en una experiencia de usuario más suave y receptiva, especialmente en aplicaciones que requieren una entrada rápida y precisa, como los juegos.

**PS2** Se refiere al conector y protocolo de conexión utilizado comúnmente en teclados y ratones. Aunque ha sido ampliamente reemplazado por conexiones USB, el término PS/2 todavía se utiliza para referirse a dispositivos más antiguos o a sistemas compatibles con este estándar.

**QWERTY** El qwerty es un nombre que se le da a una disposición de las teclas del teclado de las máquinas de escribir que fue patentado en 1878 por Christopher Sholes. Fue el inventor de la máquina de escribir y el precursor del teclado moderno que conocemos hoy en día.

**SMD** Surface Mounted Device, que en inglés significa dispositivo de montaje superficial y se refiere tanto a una forma de encapsulado de componentes electrónicos, como a los equipos construidos a partir de estos componentes.

**String** En programación, una cadena (string) es una secuencia de caracteres, como letras, números y símbolos, que se utilizan para representar texto o datos en un lenguaje de programación. Las cadenas son un tipo de dato fundamental en la mayoría de los lenguajes de programación y se utilizan para almacenar y manipular información de texto, como nombres, mensajes, direcciones, etc.

**Switches** Los switches son los mecanismos debajo de cada tecla de un teclado que detectan cuándo se presiona una tecla y envían la señal al dispositivo electrónico.

**Termoretráctil** El termoretráctil es un tipo de material plástico que se contrae cuando se calienta, generalmente mediante el uso de una pistola de calor o una fuente de calor similar. Es ampliamente utilizado en la industria electrónica para proteger y aislar conexiones eléctricas y componentes, proporcionando una capa adicional de resistencia al agua, aislamiento y soporte mecánico.

**TFT** Son las siglas en inglés de "Thin-Film Transistor" (Transistor de Película Fina). Se refiere a una tecnología de pantalla que utiliza transistores de película delgada para controlar cada píxel de la pantalla de manera individual. Los paneles TFT son comúnmente utilizados en pantallas de cristal líquido (LCD) para mejorar la calidad de imagen, aumentar la velocidad de respuesta y permitir una mayor variedad

de colores. Los TFT son ampliamente utilizados en dispositivos como monitores de computadora, televisores y pantallas de dispositivos móviles.

**TTL** (Transistor-Transistor Logic) también se utiliza para referirse a un programador USB, que permite cargar nuevo firmware en dispositivos. Este programador puede ser empleado en otros proyectos de "bare bones Arduino" o como una interfaz serie USB a TTL de propósito general. Proporciona una conexión y comunicación serial para la programación y configuración de dispositivos electrónicos.

**USB** Siglas de "Universal Serial Bus" (Bus Universal en Serie). Es un estándar de conexión que permite la transferencia de datos y la conexión de dispositivos electrónicos, como impresoras, cámaras y dispositivos de almacenamiento, a través de un cable estándar.

**VID** Vendor ID, que se refiere al identificador de proveedor de un dispositivo USB.

**Vintage** Se refiere a objetos, productos o estilos que tienen una cierta edad y que son considerados clásicos o representativos de una época pasada. En el contexto de la tecnología, se utiliza para describir dispositivos antiguos que tienen un atractivo nostálgico o colección.

**Wearable** Dispositivo electrónico vestible que se lleva encima o se incorpora en la ropa y que tiene capacidades de computación y conectividad. Los wearables suelen estar diseñados para monitorizar datos relacionados con la salud, el fitness, la ubicación, entre otros, y pueden incluir dispositivos como smartwatches, brazaletes de fitness y gafas inteligentes.

**WiFi** Es una tecnología de comunicación inalámbrica que permite la conexión a Internet y la red de computadoras sin la necesidad de cables físicos. Utiliza ondas de radio de alta frecuencia para transmitir datos entre dispositivos, como computadoras, teléfonos inteligentes, tabletas y dispositivos de red, dentro de un área determinada llamada "zona de cobertura WiFi".

**Windows** Un sistema operativo desarrollado por Microsoft que es ampliamente utilizado en computadoras personales.



# Índice de figuras

2.1.	Planificación del proyecto con diagrama Gantt. . . . .	16
3.1.	Estructura física de un teclado [43] . . . . .	17
3.2.	Keycaps [22] . . . . .	18
3.3.	Switches o Interruptores mecánicos [24] . . . . .	19
3.4.	Carcasa Superior [38] . . . . .	20
3.5.	Carcasa Superior Reverso [39] . . . . .	20
3.6.	Montaje en carcasa inferior [20] . . . . .	21
3.7.	Montaje en carcasa superior [20] . . . . .	21
3.8.	Montaje en carcasa [20] . . . . .	22
3.9.	Montaje en carcasa sandwich [20] . . . . .	22
3.10.	Montaje en forma de carcasa superior [20] . . . . .	22
3.11.	Montaje en empaquetado [20] . . . . .	23
3.12.	Montaje sin plate [20] . . . . .	23
3.13.	Plate [1] . . . . .	24
3.14.	Carcasa Inferior de Metal de teclado 60% . . . . .	25
3.15.	ISO 105 sin modificar . . . . .	29
3.16.	ISO 105 Modificado para el teclado ModernWood . . . . .	29
3.17.	Esquemático del módulo TP4056 [23] . . . . .	34
3.18.	Ejemplo básico de matriz de teclado [44] . . . . .	36
3.19.	Imagen del conector XS-12 . . . . .	37
3.20.	Imagen del conector GX-16 . . . . .	37
3.21.	Imagen del conector USB tipo A . . . . .	38
3.22.	Imagen del LED WS2812B . . . . .	38

4.1.	Matriz final de teclas [6] . . . . .	50
4.2.	Multiplexor con todos los pines conectados a sus salidas/entradas [6] . . . . .	51
4.3.	Microcontrolador principal ESP32S3 y tecla principal [6] . . .	52
4.4.	Sistema de alimentación y protección del teclado [6] . . . .	53
4.5.	Sistema de leds del teclado [6] . . . . .	54
4.6.	Conector de programación y conector de la pantalla TFT [6]	54
4.7.	Esquema completo del circuito del teclado. [6] . . . . .	55
5.1.	Imagen del plano generado por la página web “Plate & Case Builder“ [40] . . . . .	58
5.2.	Imagen del plano de la PCB [7] . . . . .	59
5.3.	Imagen del plano completo de la PCB [7] . . . . .	59
5.4.	Imagen de la vista PCB en Eagle. . . . .	61
5.5.	Imagen del plano importado en Eagle. . . . .	61
5.6.	Imagen de la PCB con los componentes ubicados y los agujeros. .	62
5.7.	Imagen de la PCB con las pistas conectadas. . . . .	65
5.8.	Imagen del plano de tierra superior de la PCB [8]. . . . .	65
5.9.	Imagen del plano de tierra inferior de la PCB [8]. . . . .	65
5.10.	Imagen del icono de difusor de luz [9]. . . . .	66
5.11.	Imagen del icono del indicador de panel de metacrilato [10]. .	67
5.12.	Imagen del icono del indicador de tuerca [11]. . . . .	67
5.13.	Imagen del icono del indicador tornillo a la carcasa [12]. . .	67
6.1.	Imagen del plano de la carcasa del teclado. . . . .	70
6.2.	Imagen del plano lateral (Vista perfil izquierda) de la carcasa del teclado [7]. . . . .	70
6.3.	Imagen del modelo 3D de la carcasa del teclado. . . . .	71
6.4.	Imagen del modelo 3D Renderizado de la carcasa del teclado.	71
7.1.	Menú principal en la pantalla TFT LCD 0.96“ . . . . .	79
7.2.	Diagrama de Flujo del bucle principal . . . . .	83
7.3.	Diagrama de Flujo de la secuencia de cambio de modo de teclado . . . . .	84

---

## ÍNDICE DE FIGURAS

## ÍNDICE DE FIGURAS

12.1. Imagen de la ESP32S3 modificada . . . . .	107
12.2. Imagen del plano acotado del espacio entre las <a href="#">keycaps</a> y la madera. . . . .	108
12.3. Imagen del teclado ModernWood conectado a un ordenador en el apartado de dispositivos bluetooth. . . . .	115



# Bibliografía

- [1] Adafruit Industries. Anodized black aluminum metal keyboard plate for gh60 cases, 2021. <https://www.flickr.com/photos/adafruit/51479696571>, Last accessed on 13-03-2024.
- [2] Bodmer, Github. Tft\_espi, 2024. [https://github.com/Bodmer/TFT\\_eSPI](https://github.com/Bodmer/TFT_eSPI), Last accessed on 15-03-2024.
- [3] Brad Suppanz, 4pcb. Pcb trace width calculator, 2018. <https://www.4pcb.com/trace-width-calculator.html>, Last accessed on 10-04-2024.
- [4] @c0z3n, @daveho, @kartikg33; Github. cherrymx-eagle, 2024. <https://github.com/c0z3n/cherrymx-eagle/tree/master>, Last accessed on 5-04-2024.
- [5] Carlos López Martínez. Teclado teseracto, 2023. <https://github.com/Electroner/Teclado-Teseracto>, Last accessed on 29-04-2024.
- [6] Carlos López Martínez. Modernwood, 2024. <https://github.com/Electroner/ModernWood/blob/main/Images/PCB/Schematic.png>, Last accessed on 20-06-2024.
- [7] Carlos López Martínez. Modernwood, 2024. <https://github.com/Electroner/ModernWood/tree/main/Planos>, Last accessed on 20-06-2024.
- [8] Carlos López Martínez. Modernwood, 2024. <https://github.com/Electroner/ModernWood/blob/main/Images/PCB/Board.png>, Last accessed on 20-06-2024.
- [9] Carlos López Martínez. Modernwood, 2024. <https://github.com/Electroner/ModernWood/blob/main/Images/PCB/Board.png>, Last accessed on 20-06-2024.
- [10] Carlos López Martínez. Modernwood, 2024. <https://github.com/Electroner/ModernWood/blob/main/Images/PCB/Board.png>, Last accessed on 20-06-2024.

- [11] Carlos López Martínez. Modernwood, 2024. <https://github.com/Electroner/ModernWood/blob/main/Images/PCB/Board.png>, Last accessed on 20-06-2024.
- [12] Carlos López Martínez. Modernwood, 2024. <https://github.com/Electroner/ModernWood/blob/main/Images/PCB/Board.png>, Last accessed on 20-06-2024.
- [13] Carlos López Martínez. Modernwood, 2024. <https://github.com/Electroner/ModernWood>, Last accessed on 20-06-2024.
- [14] CDW. Types of keyboards, 2022. <https://www.cdw.com/content/cdw/en/articles/hardware/types-of-keyboards.html>, Last accessed on 6-03-2024.
- [15] Chron, Andy Walton. Types of wireless keyboards for computers. <https://smallbusiness.chron.com/types-wireless-keyboards-computers-69487.html>, Last accessed on 6-03-2024.
- [16] Espressif Systems. Esp32-s3 datasheet, 2023. [https://www.espressif.com/sites/default/files/documentation/esp32-s3-wroom-1\\_wroom-1u\\_datasheet\\_en.pdf](https://www.espressif.com/sites/default/files/documentation/esp32-s3-wroom-1_wroom-1u_datasheet_en.pdf), Last accessed on 1-06-2024.
- [17] EURobotics.org. Como soldar componentes smd, 2024. <http://ieb-srv1.upc.es/gieb/tecniques/Manuales/smd.pdf>, Last accessed on 29-04-2024.
- [18] h2zero, Github. imble-arduino, 2023. <https://github.com/h2zero/NimBLE-Arduino>, Last accessed on 15-03-2024.
- [19] ijprest, Github. Keyboard layout editor, 2023. <https://www.keyboard-layout-editor.com/>, Last accessed on 13-03-2024.
- [20] Keyboard University. Keyboard mounting styles. <https://www.keyboard.university/200-courses/keyboard-mounting-styles-4lpp7>, Last accessed on 6-03-2024.
- [21] Keyboard University. Keyboard sizes & layouts. <https://www.keyboard.university/100-courses/keyboard-sizes-layouts-gdeby>, Last accessed on 6-03-2024.
- [22] Mercury, Wikipedia. Double-shot keycaps.jpg, 2020. [https://commons.wikimedia.org/wiki/File:Double-shot\\_keycaps.jpg](https://commons.wikimedia.org/wiki/File:Double-shot_keycaps.jpg), Last accessed on 13-03-2024.

- [23] MrNiceThings, Reddit. Hugo - how did i make my esp8266 remote?, 2020. [https://www.reddit.com/r/esp8266/comments/ehmfkx/hugo\\_how\\_did\\_i\\_make\\_my\\_esp8266\\_remote/](https://www.reddit.com/r/esp8266/comments/ehmfkx/hugo_how_did_i_make_my_esp8266_remote/), Last accessed on 14-03-2024.
- [24] Multicherry, Wikipedia. Cherry mx green switch, 2020. [https://commons.wikimedia.org/wiki/File:Cherry\\_MX\\_Green\\_switch\\_%28composite\\_ii%29.jpg](https://commons.wikimedia.org/wiki/File:Cherry_MX_Green_switch_%28composite_ii%29.jpg), Last accessed on 13-03-2024.
- [25] POPULAR SCIENCE. A beginner's guide to the world of custom mechanical keyboards, 2022. <https://www.popsci.com/diy/mechanical-keyboard-basics/>, Last accessed on 6-03-2024.
- [26] Sierra Circuits. Pcb conductor spacing and voltage calculator, 2024. <https://www.protoexpress.com/tools/pcb-conductor-spacing-and-voltage-calculator/>, Last accessed on 10-04-2024.
- [27] SnapEDA. Design electronics in a snap., 2024. <https://www.snapeda.com/discover/>, Last accessed on 1-04-2024.
- [28] SnapMagic, SnapEDA. 1n5819hw-7-f, 2024. <https://www.snapeda.com/part/1N5819HW-7-F/Diodes%20Inc./view-part/>, Last accessed on 5-04-2024.
- [29] SnapMagic, SnapEDA. 74hc154d,653, 2024. <https://www.snapeda.com/part/74HC154D,653/Nexperia/view-part/>, Last accessed on 1-04-2024.
- [30] SnapMagic, SnapEDA. C1206c105k5ractu, 2024. <https://www.snapeda.com/part/C1206C105K5RACTU/KEMET/view-part/>, Last accessed on 5-04-2024.
- [31] SnapMagic, SnapEDA. Dw01a, 2024. <https://www.snapeda.com/part/DW01A/Fortune%20Semiconductor/view-part/>, Last accessed on 5-04-2024.
- [32] SnapMagic, SnapEDA. Esp32s3wroom1n16r2, 2024. <https://www.snapeda.com/part/ESP32S3WR00M1N16R2/Espressif%20Systems/view-part/?ref=search&t=ESp32S3>, Last accessed on 5-04-2024.
- [33] SnapMagic, SnapEDA. Fs8205a, 2024. <https://www.snapeda.com/part/FS8205A/Fortune%20Semiconductor/view-part/?ref=search&t=FS8205A>, Last accessed on 5-04-2024.
- [34] SnapMagic, SnapEDA. Lm3940imp-3.3/nopb, 2024. <https://www.snapeda.com/part/LM3940IMP-3.3%2FNOPB/Texas%20Instruments/view-part/>, Last accessed on 5-04-2024.

- [35] SnapMagic, SnapEDA. Rc0805fr-07910rl, 2024. <https://www.snapeda.com/parts/RC0805FR-07910RL/Yageo/view-part/>, Last accessed on 5-04-2024.
- [36] SnapMagic, SnapEDA. Tp4056, 2024. <https://www.snapeda.com/parts/TP4056/NanJing%20Top%20Power%20ASIC%20Corp./view-part/>, Last accessed on 5-04-2024.
- [37] SnapMagic, SnapEDA. Ws2812b, 2024. <https://www.snapeda.com/parts/WS2812B/Worldsemi/view-part/?ref=search&t=WS2812B%20>, Last accessed on 5-04-2024.
- [38] snuci, wikimedia. Cherry kfn3-8358 keyboard case top interior, 2016. [https://deskthority.net/wiki/File:Cherry\\_KFN3-8358\\_keyboard\\_case\\_top\\_exterior.jpg](https://deskthority.net/wiki/File:Cherry_KFN3-8358_keyboard_case_top_exterior.jpg), Last accessed on 13-03-2024.
- [39] snuci, wikimedia. Cherry kfn3-8358 keyboard case top interior, 2016. [https://deskthority.net/wiki/File:Cherry\\_KFN3-8358\\_keyboard\\_case\\_top\\_interior.jpg](https://deskthority.net/wiki/File:Cherry_KFN3-8358_keyboard_case_top_interior.jpg), Last accessed on 13-03-2024.
- [40] Swill, Swillkb. Plate & case builder, 2023. <http://builder.swillkb.com/>, Last accessed on 13-03-2024.
- [41] Switch & click. How much does a custom keyboard cost? <https://switchandclick.com/what-is-the-average-cost-for-a-custom-mechanical-keyboard/>, Last accessed on 6-03-2024.
- [42] The Silicon Underground, David L. Farquhar. Types of keyboard connections, illustrated, 2022. <https://dfarq.homeip.net/types-of-keyboard-connections-illustrated/>, Last accessed on 6-03-2024.
- [43] Theresa McDonough, techwithtech. Mechanical keyboard parts: Names & functions?, 2023. <https://techwithtech.com/mechanical-keyboard-parts/>, Last accessed on 13-03-2024.
- [44] w4ilun, Github. pocket-keyboard, 2019. [https://github.com/w4ilun/pocket-keyboard/blob/master/images/schem\\_keys.png](https://github.com/w4ilun/pocket-keyboard/blob/master/images/schem_keys.png), Last accessed on 15-03-2024.

