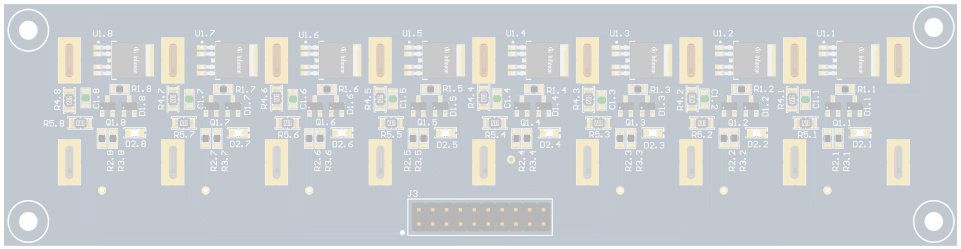
El objetivo principal del presente proyecto es desarrollar un producto dedicado a la conmutación de sistemas de iluminación en la industria de automoción. Está compuesto de tres bloques diferenciados : el desarrollo de la electrónica necesaria para generar la conmutación y su control, el firmware para la gestión de la comunicación y una estructura mecánica para el encapsulado de las diferentes partes hardware y terminales.



Álvaro Mazuecos Nogales es un ingeniero electrónico industrial de Granada, España. Terminó su Máster en Ingeniería Electrónica Industrial en 2022 en la Universidad de Granada.

Andrés María Roldán Aranda es el responsable académico de este proyecto y tutor del estudiante. Es profesor en el departamento de Electrónica y Tecnología de los Computadores en la Universidad de Granada.

TRABAJO FIN DE MÁSTER

Diseño de sistema de iluminación en automoción

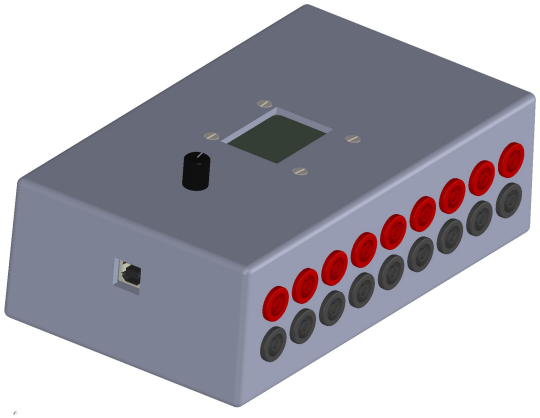
Álvaro Mazuecos Nogales

ELECTRÓNICA INDUSTRIAL

21/22

UNIVERSIDAD DE GRANADA

Máster en Electrónica Industrial



Trabajo Fin de Máster

Diseño de sistema de iluminación en automoción

Álvaro Mazuecos Nogales

2021/2022

Tutor: Andrés María Roldán Aranda

Printed in Granada, September 2022.

“Diseño de sistema de iluminación en automoción”

MÁSTER EN
ELECTRÓNICA INDUSTRIAL

Trabajo Fin de Máster

"Diseño de sistema de iluminación en automoción"

CURSO ACADÉMICO: 2022

Álvaro Mazuecos Nogales

MÁSTER EN ELECTRÓNICA INDUSTRIAL

"Diseño de sistema de iluminación en automoción"

AUTOR:

Álvaro Mazuecos Nogales

SUPERVISADO POR:

Prof. Andrés Roldán Aranda

DEPARTAMENTO:

Departamento de Electrónica y Tecnología de los Computadores

D. Andrés María Roldán Aranda, Profesor del departamento de Electrónica y Tecnología de los Computadores de la Universidad de Granada, como director del Trabajo Fin de Máster de D. Álvaro Mazuecos Nogales,

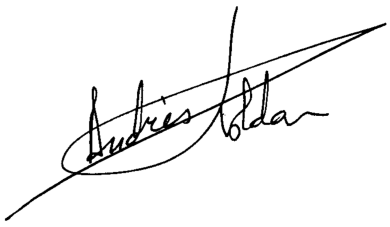
Informa:

Que el presente trabajo, titulado:

"Diseño de sistema de iluminación en automoción"

ha sido realizado y redactado por el mencionado alumno bajo mi dirección, y con esta fecha autorizo a su presentación.

Granada, a 12 de septiembre de 2022

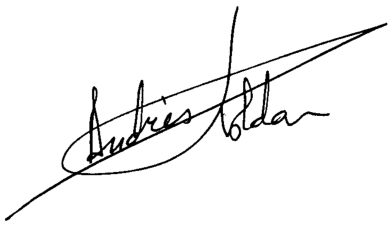


Fdo. Prof. Andrés María Roldán Aranda

Los abajo firmantes autorizan a que la presente copia de Trabajo Fin de Máster se ubique en la Biblioteca del Centro y/o departamento para ser libremente consultada por las personas que lo deseen.

Granada, a 2 de septiembre de 2022

Fdo. Álvaro Mazuecos Nogales



Fdo. Prof. Andrés María Roldán Aranda

“Diseño de sistema de iluminación en automoción”

Álvaro Mazuecos Nogales

KEYWORDS:

Altium Designer, Automotive , SolidWorks, EDA, Electronics, SDCC, MCU, PCB Design, Python,CH559.

ABSTRACT:

The aim of this project is to develop a product dedicated to switching lighting systems in the automotive industry. It is composed of three different blocks: the development of the hardware necessary to generate the channel commutation and its control, the firmware for the communication and control of the hardware components, and a mechanical structure dedicated to hold the electronics parts.

The complexity and multidisciplinary scope of this Master’s Thesis covers , the different specialties from the master’s degree in Industrial Electronics besides knowledge and skills from other engineering fields such as mechanics and software engineering. Advanced techniques from mechanics , manufacturing or simulation had been applied as well as low-level programming development.

The results of this project ends with a complete and functional system, that achieves the requirements exposed at the beginning and ends with the university master stage.

“Diseño de sistema de iluminación en automoción”

Álvaro Mazuecos Nogales

PALABRAS CLAVE:

Altium Designer, Automoción , SolidWorks, EDA, Electrónica, SDCC, MCU, PCB Design, Python,CH559.

RESUMEN:

El objetivo principal del presente proyecto es desarrollar un producto dedicado a la conmutación de sistemas de iluminación en la industria de automoción. Está compuesto de tres bloques diferenciados : el desarrollo de la electrónica necesaria para generar la conmutación y su control, el firmware para la gestión de la comunicación y control y una estructura mecánica para el encapsulado de las diferentes partes de software y componentes.

La complejidad y ámbito multidisciplinar de este Trabajo Fin de Máster permite cubrir, no sólo las diferentes especialidades del Máster de Ingeniería de **Electrónica Industrial**, sino también adquirir conocimientos y habilidades transversales o específicos de otros campos de la Ingeniería, como la **Mecánica** o la **Ingeniería del Software**. Se han analizado y aplicado técnicas avanzadas de **mecanizado**, **fabricación** (soldadura utilizando técnicas de *reflow*) o **simulación** de diferentes dispositivos, así como el desarrollo de software a bajo nivel.

El resultado de todo lo expuesto culmina con la obtención de un entorno completo y funcional, que cumple con los requisitos definidos en etapas iniciales, y con el cual se cierra la etapa universitaria de Máster.

Agradecimientos:

Comienzo dándole las gracias a mi tutor , Andrés Roldán, por ofrecerme este proyecto cuando le informé de los objetivos y conocimientos que quería adquirir durante este proyecto. Siempre he podido contar con su ayuda y supervisión, incluso en los momentos en los que no he podido estar atento en el proyecto por motivos personales. También agradecer a todos los compañeros y compañeras del grupo GranaSAT por la ayuda brindada ante dudas y problemas surgidos durante el desarrollo.

Por otro lado, quiero agradecer a los/as compañeros/as de ATISoluciones, empresa en la que comencé a trabajar en paralelo con el curso del Máster , siempre he sentido su apoyo a la hora de compaginar el proyecto con el trabajo. Además, la experiencia adquirida durante esta etapa me ha ayudado a conseguir los objetivos y resultados propuestos para este proyecto.

Gracias también a Paula, que ha sido de gran apoyo en mis momentos de agobio y procrastinación.

Finalmente , quiero agradecer a mi familia. Mis padres , Inma y Antonio, y mi hermana, Marina, por todo el apoyo y confianza depositada en mi durante estos años y a lo largo de toda mi vida.

Índice general

Autorización defensa vii

Autorización depósito librería ix

Abstract xi

Agradecimientos xvii

Índice general xix

Índice de figuras xxii

Índice de Tablas xxiv

Glosario xxv

Siglas xxvi

1. Introducción xxvii

1.0.1. Objetivos . . . xxvii

1.0.2. Metodología . . . xxvii

1.0.3. Estructura proyecto . . . xxviii

1.0.4. Diagrama de Gantt . . . xxviii

2. Requisitos 1

2.1. Hardware . . . 1

2.2. Firmware . . . 1

2.3. Mecánicos . . . 2

3. Análisis del sistema **3**

3.1. Análisis software . . . 3

3.1.1. Firmware . . . 3

3.2. Análisis hardware . . . 3

3.2.1. Microcontrolador . . . 4

3.2.2. Periféricos . . . 5

3.2.2.1. Pantalla . . . 5

3.2.2.2. *Rotary encoder* . . . 6

3.2.3. Instrumentación . . . 6

3.2.4. Comunicación . . . 6

3.3. Análisis mecánico . . . 6

3.4. Test plan . . . 7

4. Diseño y descripción **9**

4.1. Diseño mecánico . . . 9

4.2. Diseño hardware . . . 9

4.2.1. PCB Control . . . 9

4.2.2. PCB de conmutación . . . 19

4.2.3. Fabricación . . . 25

4.3. Diseño firmware . . . 26

4.3.1. Entorno desarrollo y estructura . . . 27

4.3.2. Estructura de memoria CH559L . . . 27

4.3.3. Flujo programa . . . 28

4.3.3.1. *Bootloader* . . . 29

4.3.3.2. Interrupciones . . . 30

4.3.3.3. USB . . . 32

4.3.3.4. Generación ciclos . . . 32

4.3.3.5. Dataflash . . . 34

4.3.3.6. Lectura valores analógicos . . . 34

5. Sistema de verificación y pruebas **35**

6. Conclusiones y líneas futuras **40**

6.1. Conclusiones 40
6.2. Líneas futuras 40

Bibliografía **43**

A. Presupuesto proyecto **44**
A.1. Hardware 44
A.1.0.1. Instrumentación 44
A.1.0.2. Fabricación 44
A.2. Recursos humanos 45

B. BOM **46**

C. Archivos fabricación . Gerber **47**

Índice de figuras

1.1. EasyEDA logo xxvii

1.2. Metodología desarrollo xxviii

1.3. Diagrama de Gantt xxix

3.1. Diagrama bloques sistema 4

3.2. Herramientas utilizadas 4

3.3. Entornos desarrollo software 5

3.4. Placa desarrollo basada en CH559L 5

3.5. Pantallas consideradas 6

3.6. SolidWorks®[2] 7

4.1. Vista explosionada. 10

4.2. Vista explosionada. 11

4.3. Modelo 3D . Vista planta. 12

4.4. Modelo 3D . Vista suelo. 13

4.5. Modelo 3D . Vista isométrica. 14

4.6. Modelo 3D . Vista planta. 20

4.7. Modelo 3D . Vista suelo. 21

4.8. Modelo 3D . Vista isométrica. 22

4.9. Metodología fabricación 25

4.10. PCB conmutación con pasta de estaño aplicada 25

4.11. Colocación componentes 26

4.12. Soldadura por convección 26

4.13. Exceso estaño MCU 26

4.14. Estructura memoria CH559 28
4.15. Flujo programa 29
4.16. Pantalla entrada *bootloader* 30
4.17. Subida código por UART 30
4.18. Log comando *desmeg* en Linux 32
4.19. Listado ciclos pantalla 33
4.20. Diagrama flujo generación señal 34

5.1. Inspección visual 35
5.2. Prueba alimentación 36
5.3. PCB conectada faro. 37
5.4. Ejemplo señal generada 1. 38
5.5. Ejemplo señal generada 2. 39

Índice de Tablas

1.1. Objetivos proyecto XXVII

2.1. Control PCB - Requerimientos formales 1
2.2. Firmware - Requerimientos formales 1
2.3. Mecánica - Requerimientos formales 2

3.1. Comparación pantallas LCD 5
3.2. Estructura de un posible paquete de comunicación USB 6

4.1. Descripción interrupciones utilizadas 30
4.2. Descriptores USB 32
4.3. Estructura de un ciclo 33
4.4. Estructura señal 33

A.1. Coste total de la instrumentación y herramientas usadas para el desarrollo 44
A.2. Coste total fabricación y ensamblaje 44
A.3. Coste recursos humanos 45

B.1. BOM 46

Glosario

Altium Designer® 19 software used to design PCB from schematics. It allows 3D Design, as well as electronics simulation.

API Una interfaz de programación de aplicaciones o API (del inglés Application Programming Interface) es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Son usadas generalmente en las bibliotecas (también denominadas comúnmente *librerías*)..

EDA Plataforma de herramientas software para el diseño electrónico.

MCS51 Denominación oficial de Intel para los microcontroladores 8051.

Siglas

BOM Bill Of Material.

CDC Comunication Device Class.

CSV Comma Separate Values.

GND Ground.

I2C Inter-Integrated Circuit.

LCD Liquid Crystal Display.

MCU Microcontroller unit.

PCB Printed Circuit Board.

PWM Pulse Width Modulation.

ROM Read Only Memory.

SDCC Small Device C Compiler.

SMD Surface Mount Devices.

SPI Serial Peripheral Interface.

TFT Thin-Film Transistor.

TTL Transistor-Transistor Logic.

UART Universal Asynchronous Receiver-Transmitter.

USB Universal Serial Bus.

Capítulo 1

Introducción

El objetivo de este proyecto es el diseño de un producto capaz de someter a los sistemas de iluminación en automoción a largos periodos de conmutación según unos ciclos formados por diferentes patrones de señales cuadradas que han sido definidos previamente. A la vez, se obtendrán lectura del consumo durante estos periodos de conmutación.

Este trabajo de fin de Máster ha sido desarrollado junto al grupo GranaSAT en sus instalaciones en Granada.

Figura 1.1 – *EasyEDA logo*

1.0.1. Objetivos

En la tabla 1.1 se listan los objetivos general a desarrollar.

Objetivo	Descripción
Objetivo 1	Diseño de una placa para realizar la conmutación, que permitirá el suministro de alta corriente.
Objetivo 2	Diseño de una placa de control que permita al usuario seleccionar y configurar el sistema
Objetivo 3	Habilitar protocolos de comunicación para interactuar con el dispositivo
Objetivo 4	Proporcionar una estructura mecánica para encapsular la electrónica

Tabla 1.1 – *Objetivos proyecto*

1.0.2. Metodología

La metodología usada a la hora de trabajar en este proyecto sigue el esquema de la figura 1.2 y consta de las siguientes fases:

1

- **Requerimientos del sistema** : Se analizan principales requerimientos del sistema sin detalles técnicos. Visión general de las necesidades.
- **Análisis del sistema** : Se analizan las tecnologías que se van a usar durante el desarrollo, se discuten las distintas alternativas para ver cual es la que mejor se adapta a los objetivos, obteniendo así más conocimiento sobre el sistema.
- **Diseño del sistema** : Una vez se tenga claro las necesidades del sistema, en esta fase se desarrolla el producto atentiendo a los requirimientos definidos en la primera fase.
- **Test y validación** : Del diseño del sistema se obtiene un prototipo, durante esta fase se somete al producto a pruebas validando que todos los objetivos y requisitos son cumplidos. En caso contrario, se vuelve a retomar la fase de diseño para arreglar o modificar los problemas encontrados.

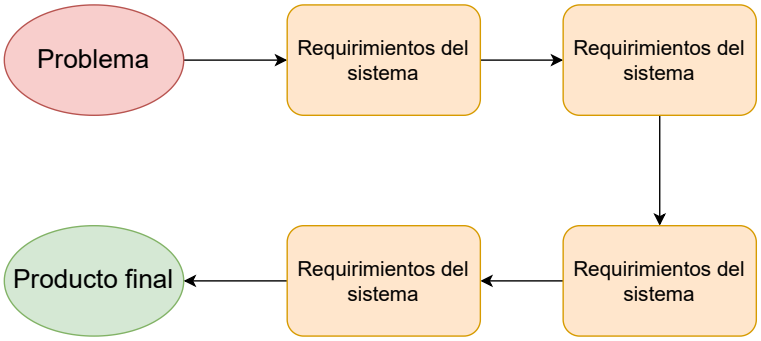


Figura 1.2 – *Metodología desarrollo*

Por otro lado, para el desarrollo de software se usará GitHub para el proceso de control de versiones. Los repositorios se almacenarán en el servidor de Gitlab de GranaSAT.

1.0.3. Estructura proyecto

Este proyecto está estructurado en 7 capítulos y 4 apéndices. La información descrita progresa desde conceptos básicos, ideas generales hasta una información más detallada del sistema. A continuación se describe brevemente la información descrita en cada capítulo:

- **Introducción**. Se comentan los objetivos que se quieren alcanzar, el tiempo empleado, la estructura y metodología de desarrollo del proyecto.
- **Requisitos del sistema**. Se proporcionan los requisitos que deben alcanzar las distintas partes del proyecto, se dividen en requisitos mecánicos, hardware y firmware.
- **Análisis del sistema**. Se hace un análisis sobre las diferentes alternativas para afrontar el proyecto, entornos de desarrollo y tecnologías que se van a usar.
- **Diseño del sistema** . El sistema es descrito de una manera más técnica y detallada de todas las partes.
- **Sistema de verificación y pruebas**. Se añaden resultados obtenidos una vez terminado el diseño, los métodos empleados para la verificación y validación de los requisitos.
- **Conclusiones y líneas futuras** . Se describen los resultados más relevantes, los fallos y se definen las posibles mejoras y nuevas ideas de avance para futuras versiones del proyecto.

1.0.4. Diagrama de Gantt

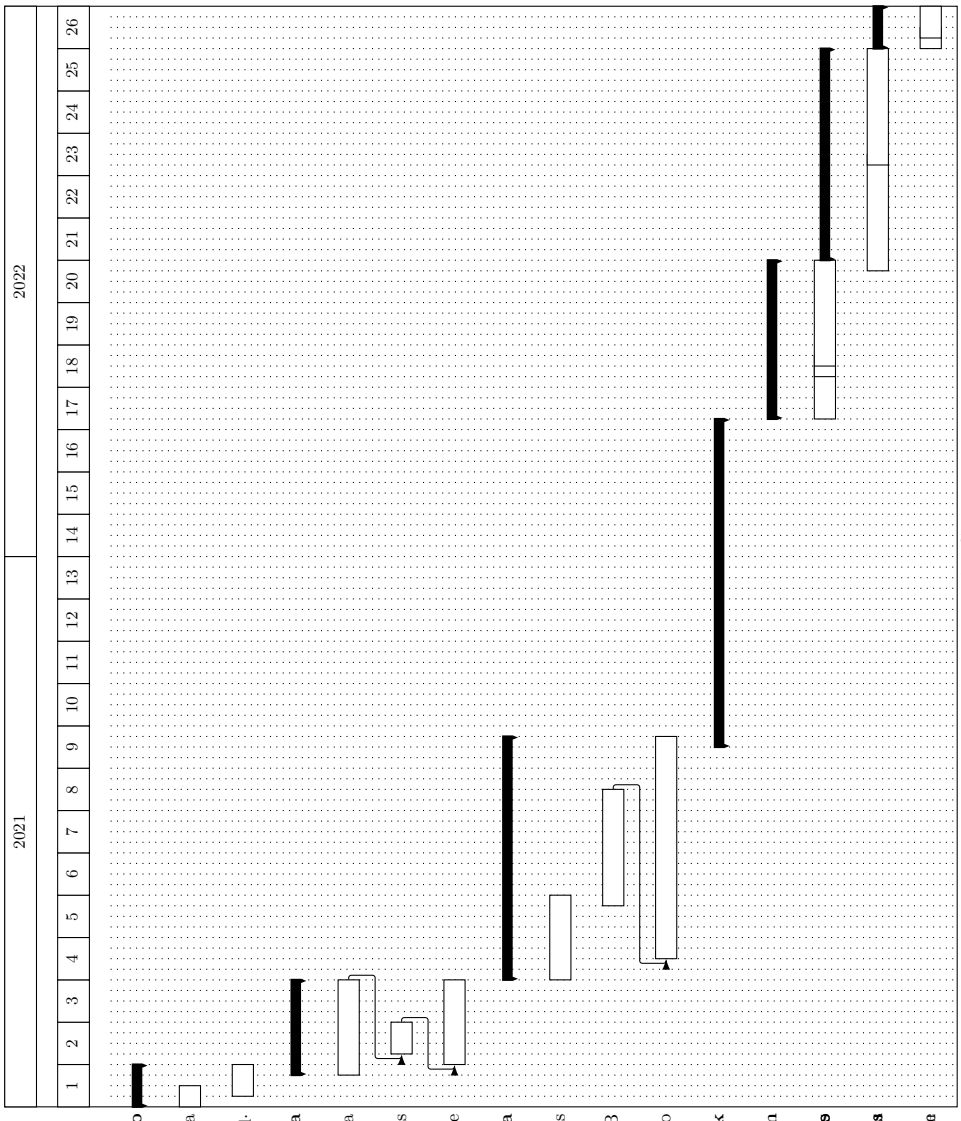


Figura 1.3 – *Diagrama de Gantt*

Capítulo 2

Requisitos

2.1. Hardware

Ref.	Requerimientos
CP.FoR.1	Comunicación USB 2.0.
CP.FoR.2	Rotary con interruptor y display.
CP.FoR.3	8 salidas digitales, al menos 2 con configuración PWM.
CP.FoR.4	8 entradas analógicas.
CP.FoR.5	La disposición del *display* , *rotary encoder* y el conector USB tiene que estar alineada y centrada en la placa.
CP.FoR.5	Entrada alimentación 12V. Mínimo 72W.

Tabla 2.1 – *Control PCB - Requerimientos formales*

2.2. Firmware

Ref.	Requerimientos
F.FoR.1	Dispositivo renocible como CDC
F.FoR.2	Creación de señales cuadras configurables con resolución de al menos 500 ms.
F.FoR.3	Selección de señales a través de un *rotary encoder*.
F.FoR.4	Comunicación I2C o SPI con *display*.
F.FoR.5	Permitir entrar al *bootloader* desde la aplicación.
F.FoR.6	Protocolo de comunicación para creación y configuración de señales en tiempo real.

Tabla 2.2 – *Firmware - Requerimientos formales*

2.3. Mecánicos

Ref.	Requerimientos
M.FoR.1	Carcasa para encapsular la electrónica y proveer una buena experiencia de uso al usuario.
M.FoR.2	Conectores banana hembra para la conexión de los sistemas de iluminación.
M.FoR.3	Salida para conector USB.
M.FoR.4	Panel de control con pantalla y *rotary* centrado en cara .
M.FoR.5	Tapa inferior para ensamblaje con 4 tornillos M3 longitud 16mm .
M.FoR.6	La placa superior se anclará con 4 tornillos M3 de longitud 10mm a través del módulo de la pantalla y la propia placa.

Tabla 2.3 – *Mecánica - Requerimientos formales*

Capítulo 3

Análisis del sistema

En este capítulo, se realizará una análisis completo del sistema. Se discutirán las diferentes tecnologías, componentes y decisiones tratadas en los diferentes sistemas, se confirmará y justificará la elección seleccionada de las alternativas presentadas. Este capítulo estará dividido en : análisis *software*, *hardware* y mecánico.

3.1. Análisis software

3.1.1. Firmware

Para el desarrollo de *firmware* el lenguaje de programación utilizado será C, ya que el compilador seleccionado ,Small Device C Compiler (SDCC), está enfocado en este lenguaje , un compilador de C para microprocesadores con arquitectura Intel MCS51, arquitectura en la que está basada el microcontrolador(figura 3.4) elegido para este proyecto.

Respecto a la estructura y desarrollo firmware , se seguirá en lo medida de lo posible las siguientes reglas:

- Mantener las diferentes funcionalidades separadas en paquetes o librerías, haciéndolas reutilizables en cualquier parte.
- No repetir código.
- Trasladar todo proceso repetitivo durante el proceso de desarrollo a un *script*.
- Las llamadas de las interrupciones no pueden contener mucho código, recomendable cambio de estado de variable o estado que será tratado por el proceso principal.
- Documentación del código.

3.2. Análisis hardware

Para cumplir con los requisitos hardware de la tabla 2.1, el sistema tendrá dos placas, una encargada del control y comunicación del sistema y otra en la que se produce la conexión y la conmutación de los transistores para alimentar los faros, en la figura 3.1 se puede observar un diagrama de bloques del hardware.

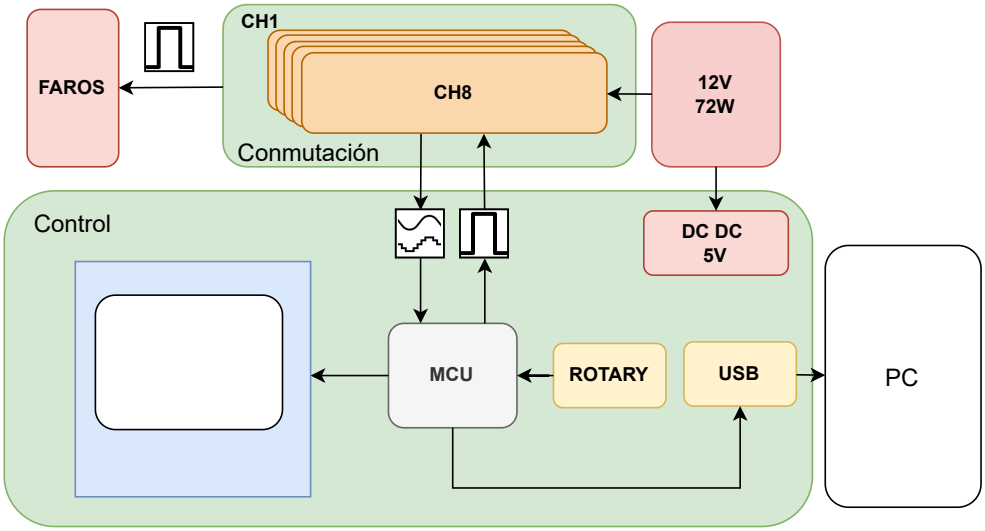


Figura 3.1 – *Diagrama bloques sistema*

Para el prototipado de las PCB se usará la herramienta Altium Designer® 19. Sin embargo, para el diseño de una placa de desarrollo usada durante la fase de diseño para el desarrollo *firmware*, se ha usado la herramienta , llamada EasyEDA 3.2,que pertenece al fabricante JLPCB, ya que tienen disponibles todos los modelos de los componentes que tienen *stock* de forma online,lo que permite un prototipado muy rápido. En la figura 3.4 se muestra la placa de desarrollo con la que se trabajará en las primeras fases, contienen los componentes principales que se van a usar como el *rotary*, los pines para colocar el *display*, el conector USB y todas las salidas necesarias.

(a) *Altium Designer* **(b)** *EasyEDA*

Figura 3.2 – *Herramientas utilizadas*

Para la selección de componentes SMD se tendrán en cuenta la disponibilidad en LCSC, ya que será el proveedor principal para el pedido de componentes, la principal razón de este proveedor es porque es el único que dispone del microcontrolador usado.

3.2.1. Microcontrolador

El microcontrolador es el encargado de generar los ciclos de señales, leer las señales analógicas para estimar la corriente usada durante el proceso de ciclado e implementar la interfaz de usuario del menú en la pantalla LCD. Entre todos los microcontroladores disponibles en el mercado se han considerado los siguientes : ATmega328, PIC18, CH554,CH559. Hoy en día, los más usados son los microcontroladores de Microchip, añadiendo a las familias AVR y PIC, pero también se puede encontrar alternativas más económicas como CH559L.

CH559 es un microcontrolador de núcleo 8051 compatible con la arquitectura de MCS51, la velocidad media de la instrucción es de 8 a 15 veces más rápida que la estándar de MCS51. Es un microcontrolador de bajo coste y enfocado al uso de aplicaciones USB, haciendo se adapte muy bien al requisito (añadir requisito CDC). Para el desarrollo del firmware en este MCU, disponemos de varios entornos de desarrollo (IDE) como Codeblocks o Keil, se descarta éste último ya que tiene que ser usado bajo licencia y por estar limitado solo al

sistema operativo *Windows*. Codeblocks es una muy buena opción, tiene una amplia gamas de herramientas para desarrollar en C y con el compilador seleccionado. Sin embargo, para el desarrollo se ha seleccionado usar *PlatformIO* un *framework* multiplataforma, multiarquitectura para el desarrollo de sistemas embebidos. Dispone además de una extensión para el IDE más utilizado a día de hoy, VSCode.

(a) *CodeBlocks* **(b)** *PlatformIO* **(c)** *VSCode*

Figura 3.3 – *Entornos desarrollo software*

Se ha diseñado una placa desarrollo basada en el microcontrolador CH559L, que se ajuste a los objetivos del proyecto, haciendo la fase de desarrollo firmware más cómoda y permitiendo trabajar en paralelo con el diseño hardware final.

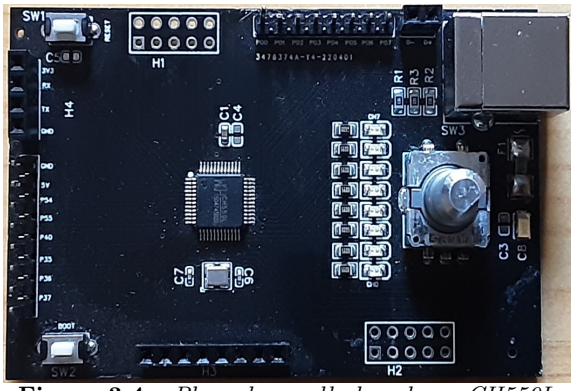


Figura 3.4 – *Placa desarrollo basada en CH559L*

3.2.2. Periféricos

3.2.2.1. Pantalla

En la pantalla LCD se mostrará el listado de ciclos de señales y las lecturas analógicas de cada canal durante la generación de éstos. Los parámetros para seleccionar la pantalla han sido tamaño, precio y comunicación , debido a que la pantalla irá integrada en la propia placa, el módulo de la pantalla debe de tener una altura y tamaño adecuado para combinarlo con la estructura mecánica del producto.

Model	**Pantalla**	**Precio €**	**Comunicación**	**Consumo**	**Seleccionado**
Nokia 5110	84 x 48	2	SPI	400 uA	Sí
128x64 SSH1106 OLED LCD	128x64 píxeles	2.54	SPI/I2C	490 uA	No
HDD4780 LCD	4x20 caracteres	3	I2C con adaptador	1 2mA	No

Tabla 3.1 – *Comparación pantallas LCD*

La pantalla seleccionada es el *Nokia 5110*, es un módulo antiguo , a diferencia de las pantallas TFT u Oled que podemos encontrar a día de hoy, pero su precio es mucho menor, además consumen menos que las

pantallas de tecnología TFT. Para el listado de señales que hay que mostrar este módulo LCD es adecuado.

(a) *Nokia 5110* **(b)** *SSH1106 OLED LCD* **(c)** *HDD4780 LCD*

Figura 3.5 – *Pantallas consideradas*

3.2.2.2. *Rotary encoder*

El *rotary encoder* permitirá la navegación a través del menú moviéndolo en dirección horaria u antihoraria y selección de señales por lo que debe de incorporar un interruptor. Es necesario que tenga una altura de más de 1.5 cm para que sobresalga de la carcasa. La salida será de tipo cuadrada de 2bits y no es necesaria precisión a la hora de selección.

3.2.3. Instrumentación

Para la medición de las diferentes señales generadas se ha hecho uso de un osciloscopio , un analizador lógico y un multímetro digital. Para probar los ciclos generados y el consumo se hará uso de los faros de iluminación disponibles en el laboratorio, ver sección 5.

3.2.4. Comunicación

El sistema será identificado como un dispositivo CDC según **requisito**, el sistema será reconocido como un puerto serie a través del USB. Esta configuración facilitará la comunicación de datos con el host, la descripción de descriptores y configuración USB en el firmware se ve simplificada.

La comunicación USB se usará para *debugear* el programa y para un posible protocolo de comunicación para controlar la generación de señales y configuración del sistema. El protocolo de comunicación se basaría en el envío y recepción de paquetes de datos que tendrían una estructura como en aparece en la tabla 3.2. Un identificador de un comando que indique la funcionalidad a realizar, la longitud de los datos que se van a transmitir para saber cuanto espacio en memoria reservar o asignar, un *checksum* para confirmar que no ha habido un error durante la transmisión y por ultimo, los datos correspondientes que se quieren transmitir o recibir.

CMD (2Bytes)	Length data (2 Bytes)	CheckSum (2 Bytes)	Data packet

Tabla 3.2 – *Estructura de un posible paquete de comunicación USB*

3.3. Análisis mecánico

Para el desarrollar el diseño mecánico del producto, que encapsulará las dos PCBs de forma robusta y permitirá el uso del sistema de manera cómoda, se hará uso del software de SolidWorks®.El uso de este software nos permite una sincronización entre el modelado 3D de las placas y la estructura mecánica, permitiendo ver de forma constante los cambios realizados.

Figura 3.6 – *SolidWorks®[2]*

3.4. Test plan

Para probar los diferentes ciclos de señales generadas se hará uso de un analizador lógico junto a algún software compatible con éste, como puede ser PulseView o Logic 2 de Salesae. Se ha elegido Logic 2 ya que proporciona una para la captura de las señales, mediante una librería de Python se pueden escribir test automáticos para la captura y lectura de señales. Otra opción, sería la automatización de captura de datos con un osciloscopio con entradas digitales mediante el protocolo de comunicación que éstos integran, almacenando los datos en el PC.

```
import matplotlib.pyplot as plt
import numpy as np

def cm_to_inch(value):
    return value/2.54
class Signal :
    def __init__(self,channel,duration) -> None:
        self.x = [0]
        self.y = [0]
        self.offset = 0
        self.s = 0
        self.signalDuration = duration
        self.channel = channel
    def process(self,signalPacket) :
        self.inValue    = signalPacket[0]
        self.tDelay     = signalPacket[1]
        self.onValue    = signalPacket[2]
        self.timeOn     = signalPacket[3]
        self.timePeriod = signalPacket[4]
        self.nCycles    = signalPacket[5]
        self.timeOff = self.timePeriod - self.timeOn
        self.__handlePeriod()
    def __handlePeriod(self) -> None :
        for n in range(0,self.nCycles ):
            if(n == 0):
                if(self.tDelay > 0):
                    self.s += self.tDelay
                    self.x.append(self.s)
                    self.y.append(self.inValue)
                self.s += self.timeOn
```

```
                    self.y.append(self.onValue)
                    self.x.append(self.s)
                    self.s += self.timeOff
                    self.x.append(self.s)
                    self.y.append(int(not self.onValue))
                else:
                    self.s += self.timeOn
                    self.x.append(self.s)
                    self.y.append(self.onValue)
                    self.s += self.timeOff
                    self.x.append(self.s)
                    self.y.append(int(not self.onValue))
    def plot(self) :
        self.y = list(map(lambda x : x - self.channel*1.1,self.y))
        if(self.x[-1] < self.signalDuration):
            self.y.append( - self.channel*1.1)
            self.x.append(self.signalDuration)
        plt.step(self.x,self.y,linewidth=5)
        plt.xticks(np.arange(0,self.signalDuration,step=10),rotation
=45)
        plt.yticks([])
        plt.grid(True)
```

Listado 3.1 – *Clase en Python para previsualización selañes*

Para una previsualización previa de todas las señales a generar se desarrollarán diverso métodos en Python,vease código en 3.4,haciendo uso de la herramienta Jupyter Notebook, lo que permite a su vez ir documentando los ciclos, de esta manera se tendrá una visión más clara de las salidas que se tienen que obtener de las salidas de la placa. A la comprobar que las salidas generadas de la parte electrónica se reducirá la duración a milisegundos, ya que lo que interesa es comprobar que las formas de onda generadas sean las correctas sin tener en cuenta los posibles desfases de tiempo que puedan ocurrir. Una vez comprobadas la forma de las señales, se harán las pruebas oportunas para comprobar que las señales se conmutan en la tolerancias especificadas. El desarrollo de test conlleva una gran cantidad de tiempo, pero a la vez hace el proceso de desarrollo más seguro y cómodo, ya que se pueden detectar fallos o anomalías de forma rápida.

Capítulo 4

Diseño y descripción

En este capítulo se va a profundizar en el diseño de las diferentes partes y funcionalidades del sistema, con las diferentes tecnologías y componentes mencionados en el análisis del sistema. Se explicará el diagrama de flujo del firmware, el diseño de las PCB, la estructura del diseño mecánico y ensamblaje.

4.1. Diseño mecánico

En esta parte se va a mostrar el diseño de la parte mecánica del producto, éste contiene un componente principal donde se integran las dos placas, un componente inferior, que es la tapa. Todas las dimensiones han sido parametrizadas y relacionadas entre sí, para que ante el cambio de alguna el sistema y todas las relaciones de posición se encuentren bien definidas. Las dos partes mecánicas comparte un archivo en CSV donde se encuentran las variables definidas. En la figura 4.2 se muestra el producto una vez ha sido ensamblado. En la figura 4.1 se ofrece una vista explosionada del sistema con todos sus componentes.

4.2. Diseño hardware

4.2.1. PCB Control

Para el diseño de esta placa hay que tener en cuenta el diseño mecánico del producto. Una vez han sido fijados las dimensiones de la carcasa se puede ir definiendo la posición de los componentes que interactúan con el usuario. En las siguientes páginas se pueden consultar el esquemático de esta placa, donde se muestra además información y notas acerca del diseño. Las procedimiento a la hora de diseño de esta placa han sido las siguientes :

- La pantalla debe de estar centrada en la cara superior de la carcasa.
- El centro del *rotary* está alineado con el centro de la pantalla.
- El centro conector CSV está alineado con estos elementos también.
- La dimensión de la placa viene definida principalmente por la distancia entre la pantalla y el conector USB, ya que un elemento está en el centro del producto y otro sobresale por la cara derecha.
- Los botones se encuentran en la para inferior de la placa como se puede ver en la figura 4.4, para poder acceder a ellos por la parte inferior del producto, ya que en caso contrario,una vez ensamblada la placa con la mecánica habría que desmontarla para pulsarlos.

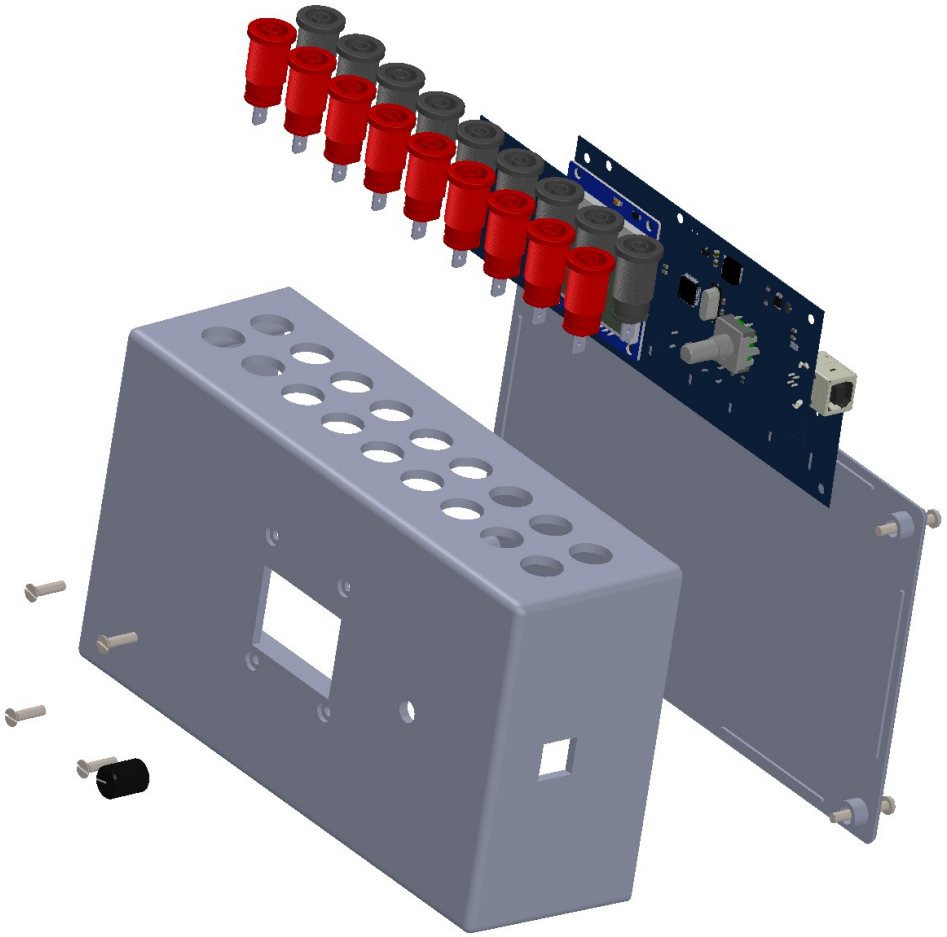


Figura 4.1 – *Vista explosionada.*

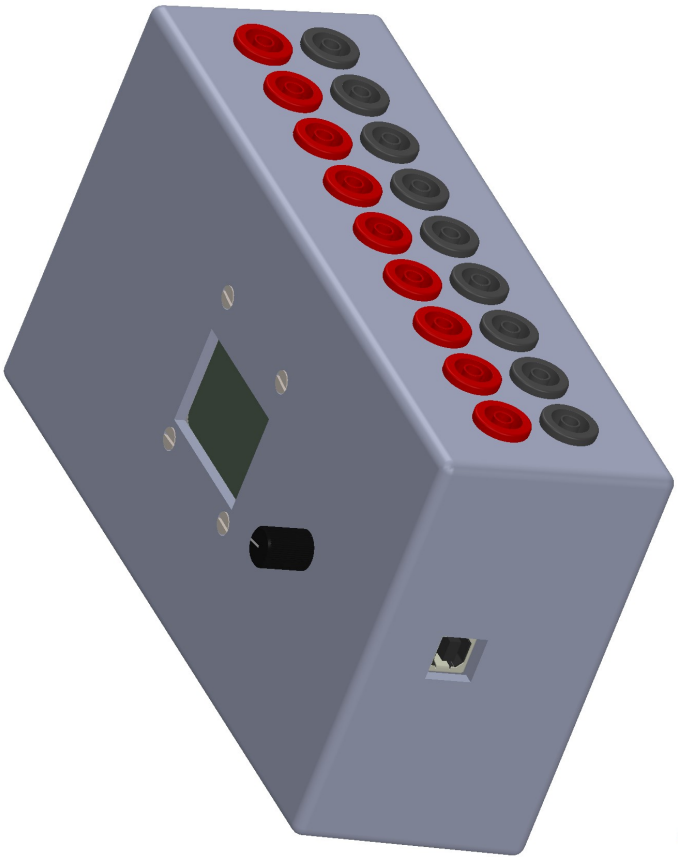


Figura 4.2 – *Vista explosionada.*

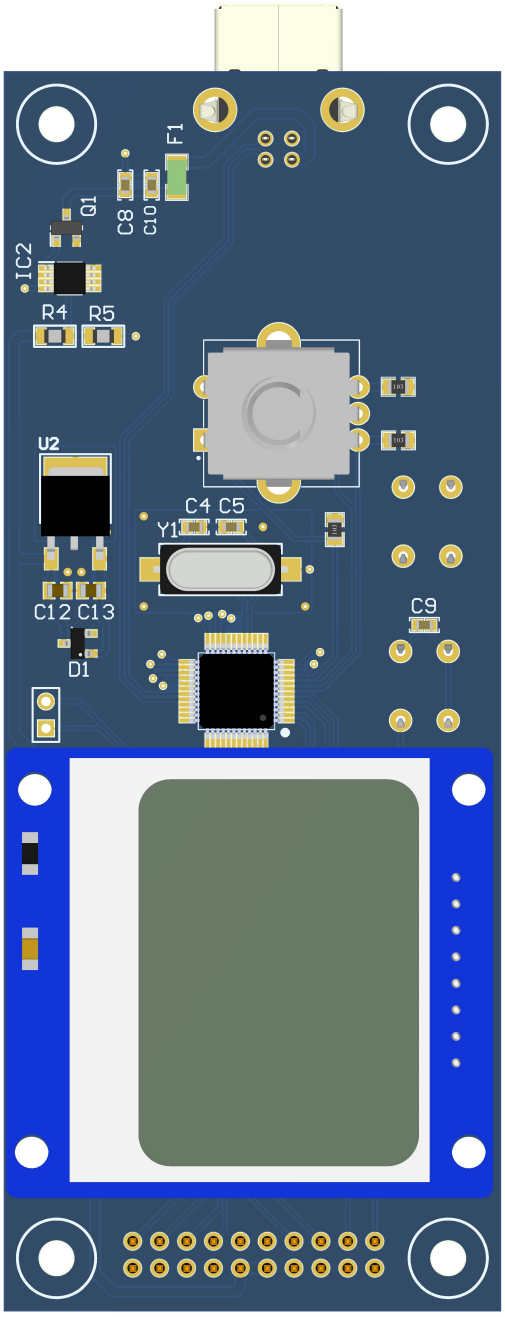


Figura 4.3 – *Modelo 3D . Vista planta.*

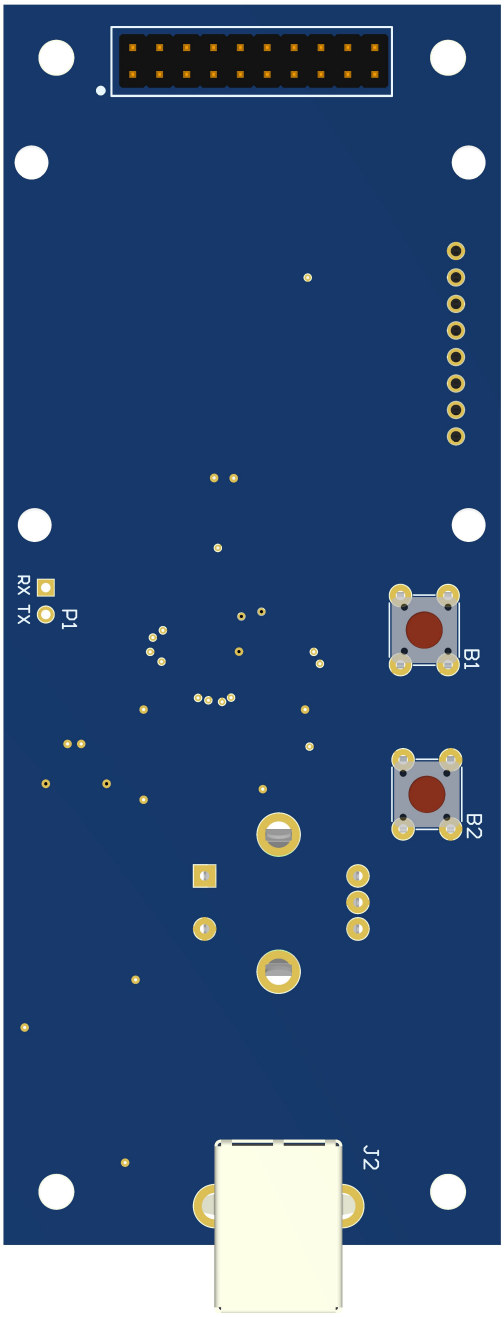


Figura 4.4 – *Modelo 3D . Vista suelo.*

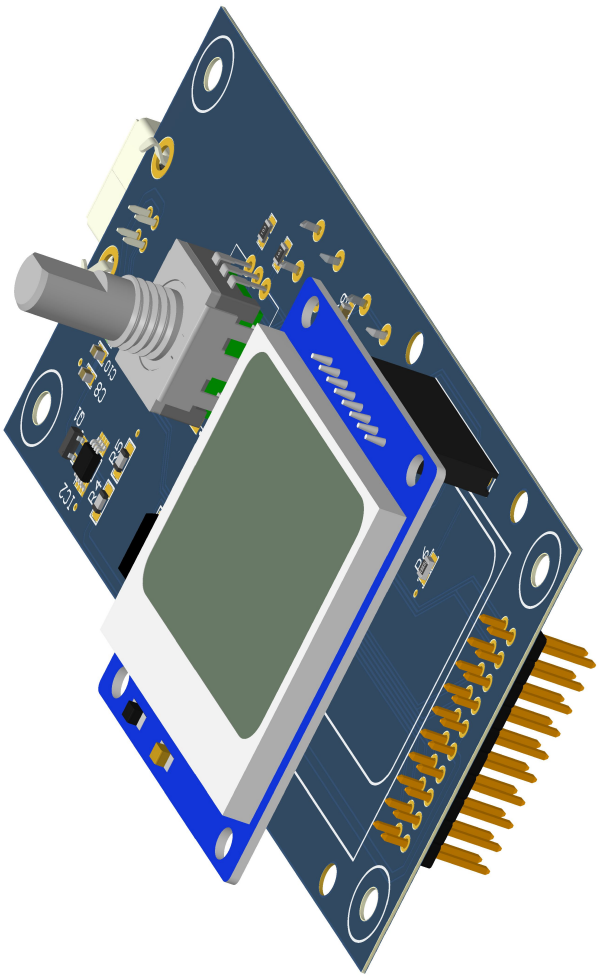
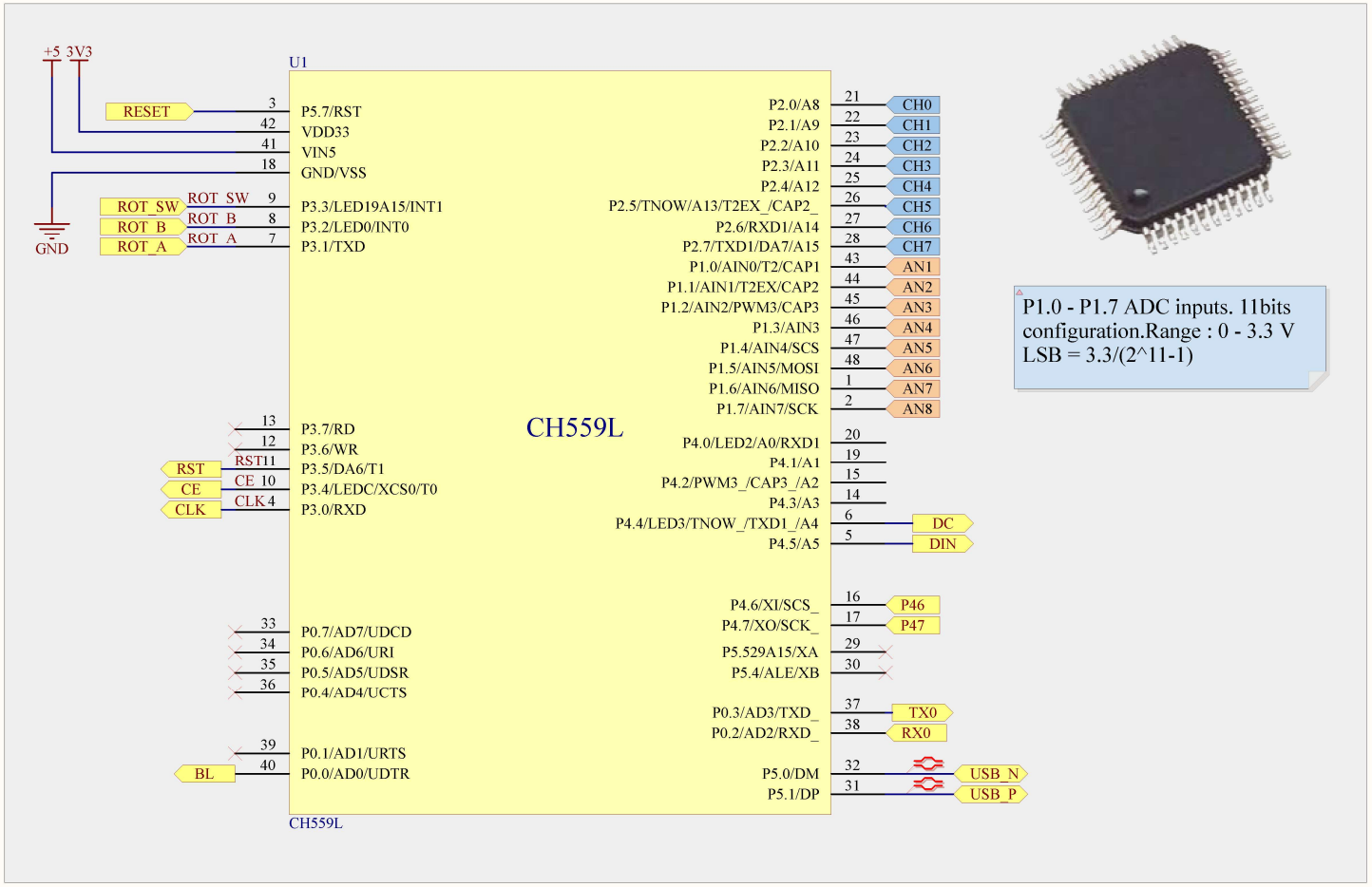


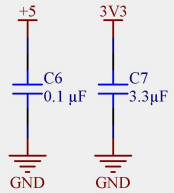
Figura 4.5 – *Modelo 3D . Vista isométrica.*

CH559 MCU



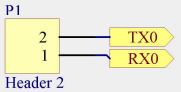
P1.0 - P1.7 ADC inputs. 11bits configuration.Range : 0 - 3.3 V
LSB = 3.3/(2^11-1)

Decoupling capacitors



Place as close as possible to their respective pins

UART



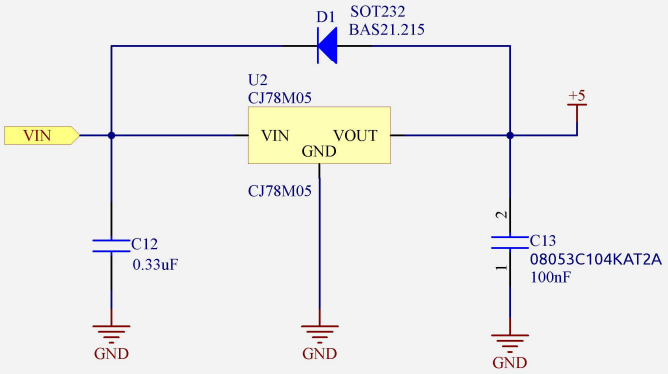
UART lines for debug or flash code in bootloader mode

Designer's signature	Sheet title: **MCU.SchDoc**		*GRANASAT*
	Project title: **CH55x_V1l1.PrjPcb**		
Supervisor's signature	Designer: **Álvaro Mazuecos Nogales**		
	Date: **12/09/2022**	Revision: **1.3**	Sheet: 2 of 5

Low Dropout Regulator (LDO)

Parameter	Symbol	Test conditions	Min	Typ	Max	Unit
Output Voltage	Vo	T_J=25℃	4.85	5	5.15	V
		7V≤V_i≤20V, Io=5mA-350mA	4.75	5	5.25	V
Load Regulation	ΔVo	Io=5mA-0.5A,T_J=25℃		15	100	mV
		Io=5mA-200mA,T_J=25℃		5	50	mV
Line Regulation	ΔVo	7V≤V_i≤25V, Io=200mA,T_J=25℃		3	100	mV
		8V≤V_i≤25V, Io=200mA,T_J=25℃		1	50	mV
Quiescent Current	Iq	T_J=25℃		4.2	6	mA
Quiescent Current Change	ΔIq	8V≤V_i≤25V, Io=200mA			0.8	mA
	ΔIq	5mA ≤I_O≤350mA			0.5	mA
Output Noise Voltage	V_N	10Hz≤ f ≤100KHz,T_J=25℃		40	200	μV/Vo
Ripple Rejection	RR	8V≤V_i≤18V,f=120Hz,Io=300mA	62	80		dB
Dropout Voltage	Vd	Io=350mA,T_J=25℃		2	2.5	V
Short Circuit Current	Isc	Vi=10V,T_J=25℃		300		mA
Peak Current	Ipk	T_J=25℃		0.5		A

LDO 5V (CJ78M05)



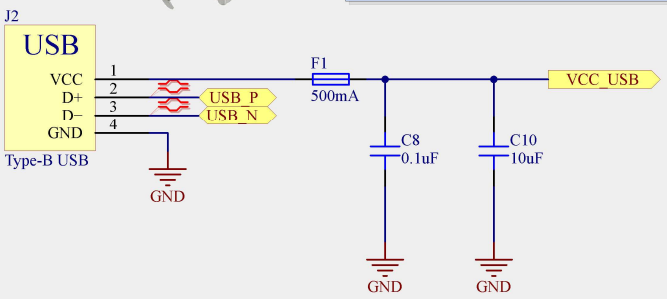
To avoid revese current a diodo is placed between the output and input of the LDO

Designer's signature	Sheet title: **LDO.SchDoc**		*GRANASAT*
	Project title: **CH55x_V1l1.PrjPcb**		
Supervisor's signature	Designer: **Álvaro Mazuecos Nogales**		
	Date: **12/09/2022**	Revision: **1.3**	Sheet: 3 of 5

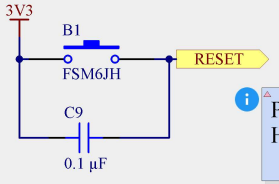
Misc

USB

However USB 2.0 protocol is limited to 500mA ,a fuse has been added to protect the MCU against current peaks

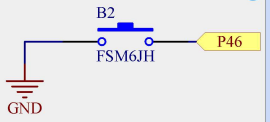


RESET



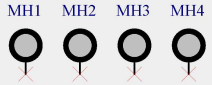
PIN 5.7(RST) is set HIGH to reset the chip

BOOTLOADER



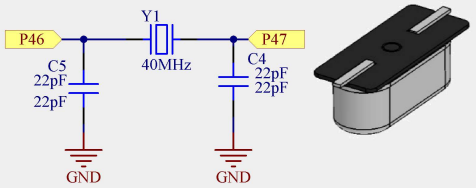
Set pin 4.6 to ground while powering the MCU to enter in bootloader mode and flash the code

MOUNTING HOLES

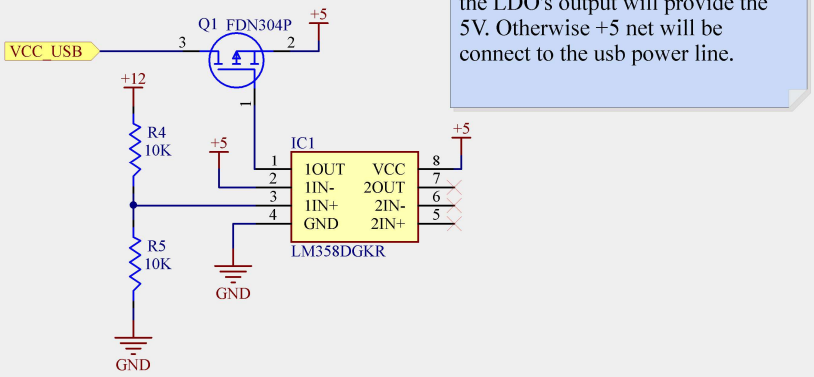


MOUNTING HOLES 7.2 DIAMETER, 3.2 MM DRILL PLATE

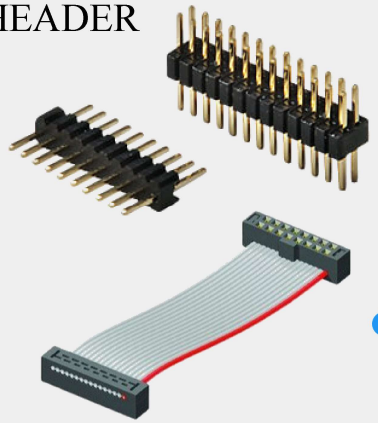
CRYSTAL OSCILATOR

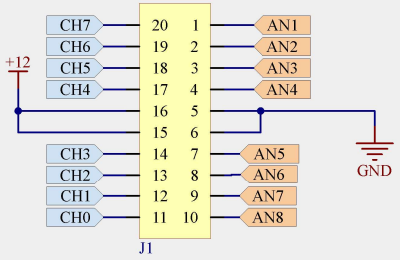


When IN1+ > 5V,Q1 will be off and the LDO's output will provide the 5V. Otherwise +5 net will be connect to the usb power line.



PIN HEADER

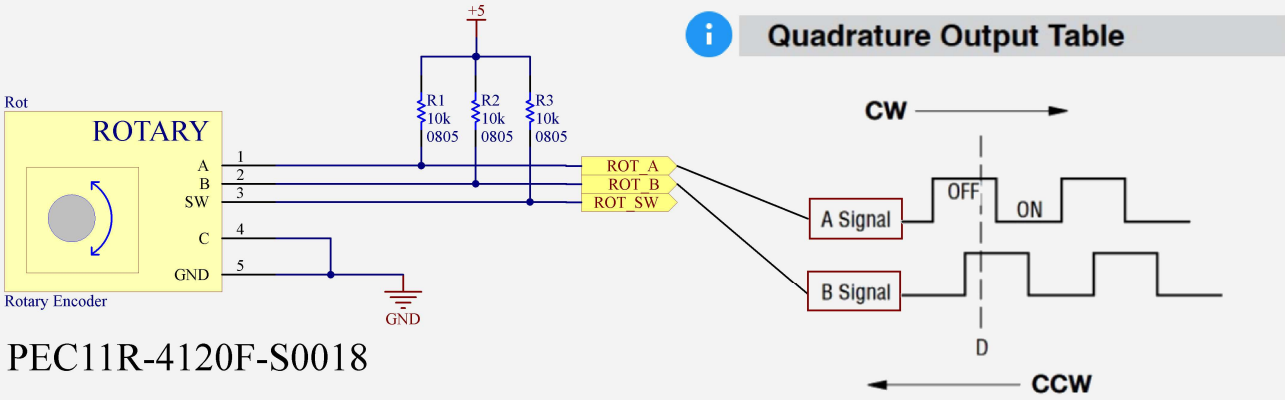


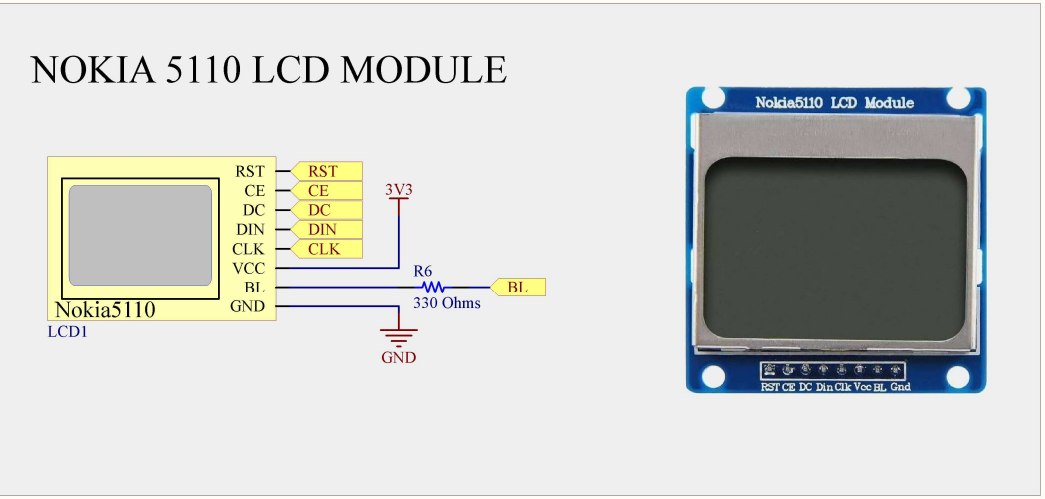


Use ribbon wire to make the connection with the PCB

Designer's signature	Sheet title: **MISC.SchDoc**		*GRANASAT*
	Project title: **CH55x_V1l1.PrjPcb**		
Supervisor's signature	Designer: **Álvaro Mazuecos Nogales**		
	Date: **12/09/2022**	Revision: **1.3**	Sheet: 3 of 5

Display and Rotary





Designer's signature	Sheet title: **display_and_rotary.SchDoc**		*GRANASAT*
	Project title: **CH55x_V1l1.PrjPcb**		
Supervisor's signature	Designer: **Álvaro Mazuecos Nogales**		
	Date: **12/09/2022**	Revision: **1.3**	Sheet: 4 of 5

4.2.2. PCB de conmutación

En esta placa se realizarán las conmutaciones y conectarán las cargas o los faros a través de unos conectores banana, la colocación de los conectores se realiza en la parte frontal de la carcasa **??**, hay un total de 8 canales donde cada canal tiene un terminal positivo y otro negativo, además de los terminales de alimentación, se distribuyen los terminales en un matriz de 2x9 , donde la fila superior corresponde con los terminales positivos y la fila inferior los terminales negativos. La dimensión de la placa viene dada por la colocación de los terminales a lo largo de la placa.

En las siguientes páginas se puede consultar el esquemático de esta placa, donde se muestra además información y notas acerca del diseño. El circuito de conmutación está basado en el componente **BTS6143D**, un transistor de potencia para conmutación inteligente de lado alto (*Smart Highside Power Switch*). Infineon proporciona un modelo en **LTSpiceVI** de este dispositivo, por lo que se ha simulado para obtener un mayor conocimiento.

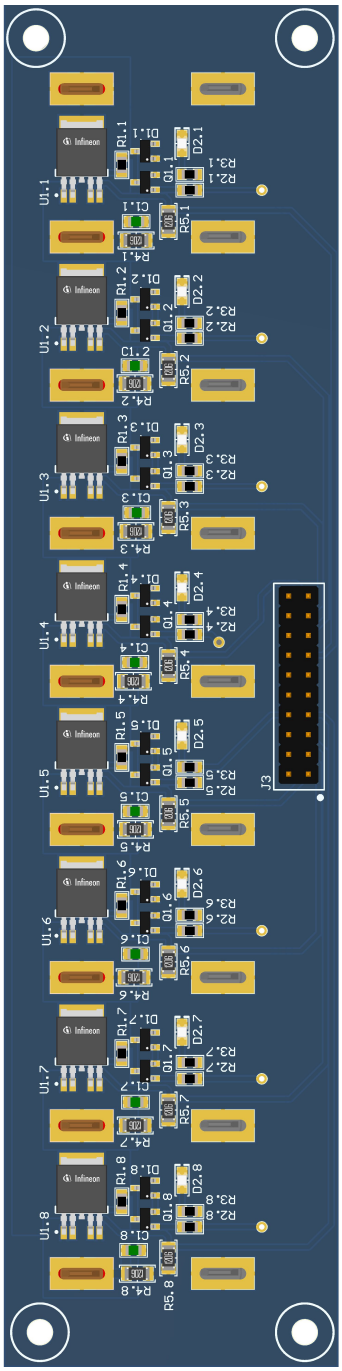


Figura 4.6 – *Modelo 3D . Vista planta.*

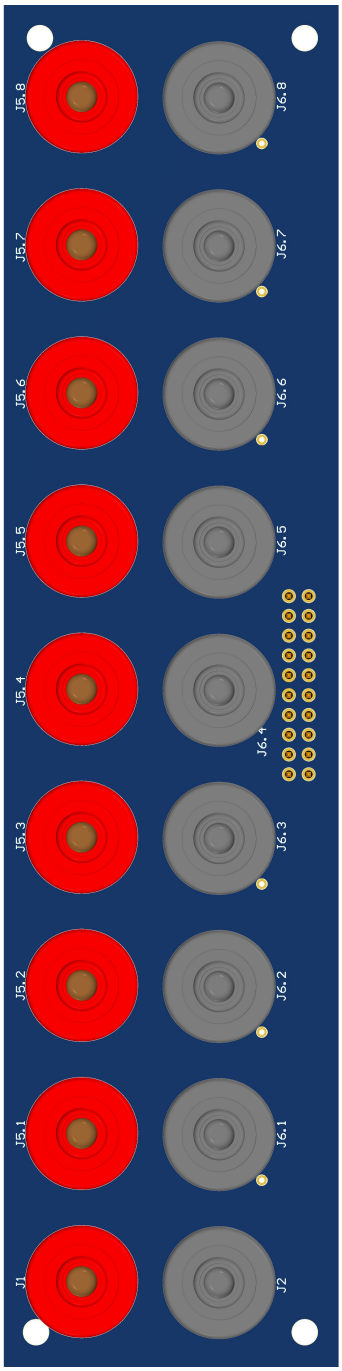


Figura 4.7 – *Modelo 3D . Vista suelo.*

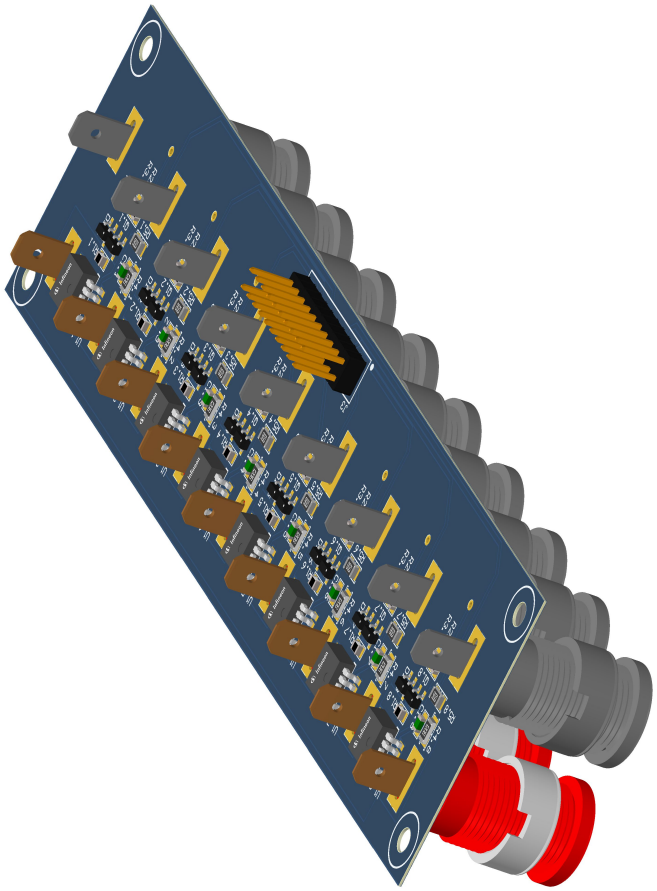
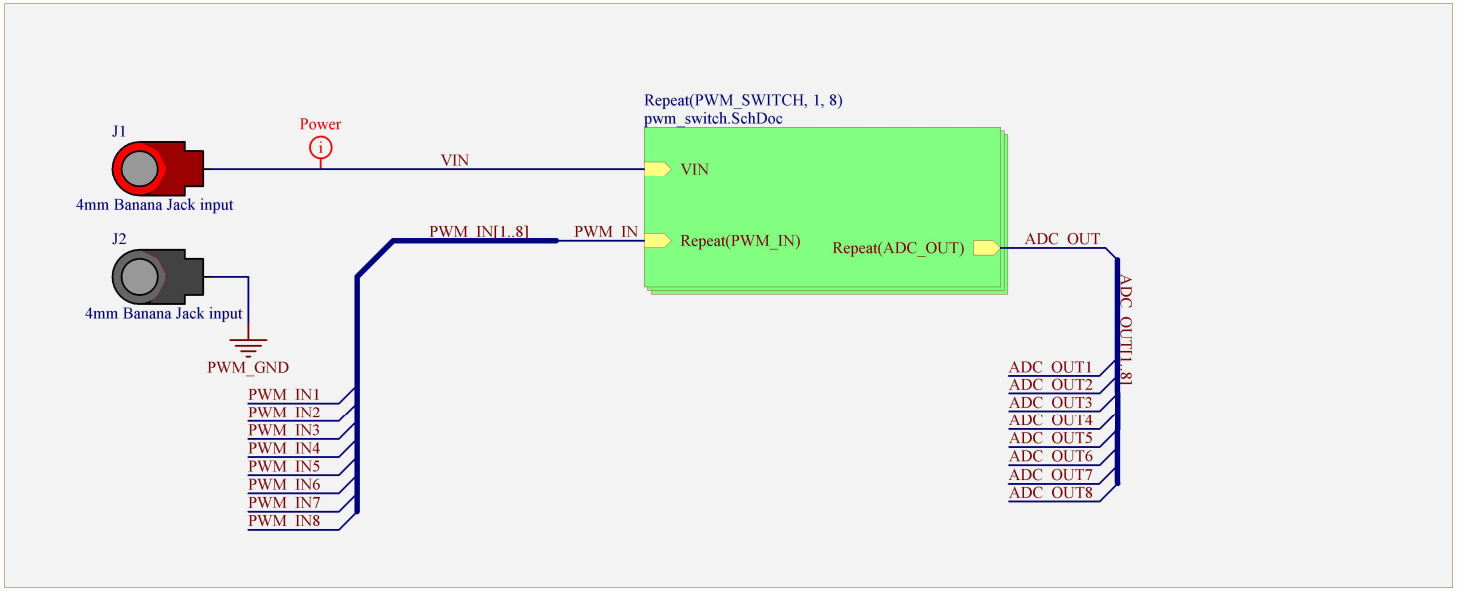


Figura 4.8 – *Modelo 3D . Vista isométrica.*

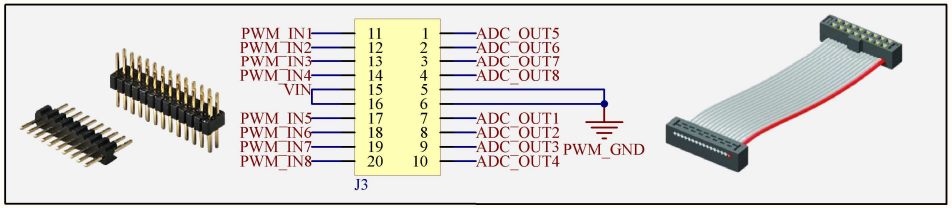
High Smart Switching PCB



MOUNTING HOLES

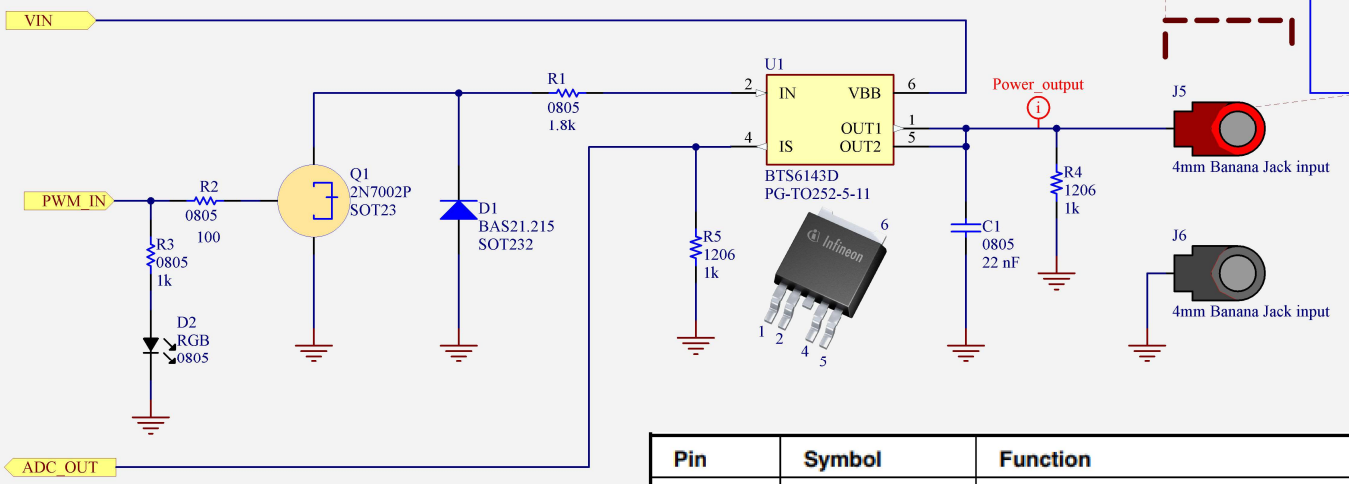
MH1 MH2 MH3 MH4

MOUNTING HOLES 7.2 DIAMETER, 3.2 MM DRILL PLATE



Designer's signature	Sheet title: **main.SchDoc**		
	Project title: **ConnectorBox.PrjPcb**		
Supervisor's signature	Designer: **Álvaro Mazuecos Nogales**		
	Date: **12/09/2022**	Revision: **1.2**	Sheet: 1 of 2

High Smart Switching Circuit based on BTS6143D



Component info

4mm Banana conector
->50 V DC, 30A

Design Warning

A esta borna deben circular al menos 6 A. Asegurarse en el rutado de la PCB.

Design Warning

Ton for switch is 500us max (2000 Hz) may cause trouble in high frquency low duty cycle situations !!

Pin	Symbol		Function
1	OUT	O	**Output;** output to the load; pin 1 and 5 must be externally shorted* .
2	IN	I	**Input;** activates the power switch if shorted to ground.
Tab/(3)	Vbb	+	**Supply Voltage;** positive power supply voltage; tab and pin3 are internally shorted.
4	IS	S	**Sense Output;** Diagnostic feedback; provides at normal operation a sense current proportional to the load current; in case of overload, overtemperature and/or short circuit a defined current is provided (see Truth Table on page 8)
5	OUT	O	**Output;** output to the load; pin 1 and 5 must be externally shorted* .

Table from DATASHEET

Designer's signature	Sheet title: **pwm_switch.SchDoc**		*GranaSAT*
	Project title: **ConnectorBox.PrjPcb**		
Supervisor's signature	Designer: **Álvaro Mazuecos Nogales**		
	Date: **12/09/2022**	Revision: **1.2**	Sheet: 2 of 2

4.2.3. Fabricación

Para la fabricación y montaje de las placas se ha seguido el esquema de la figura 4.9. Ambas placas se han fabricado a través del fabricante JLPCB, también se han pedido las plantillas(*stencils*) de cada placa para su posterior montaje. La compra de los componentes se ha realizado en LCSC, ver BOM en B.

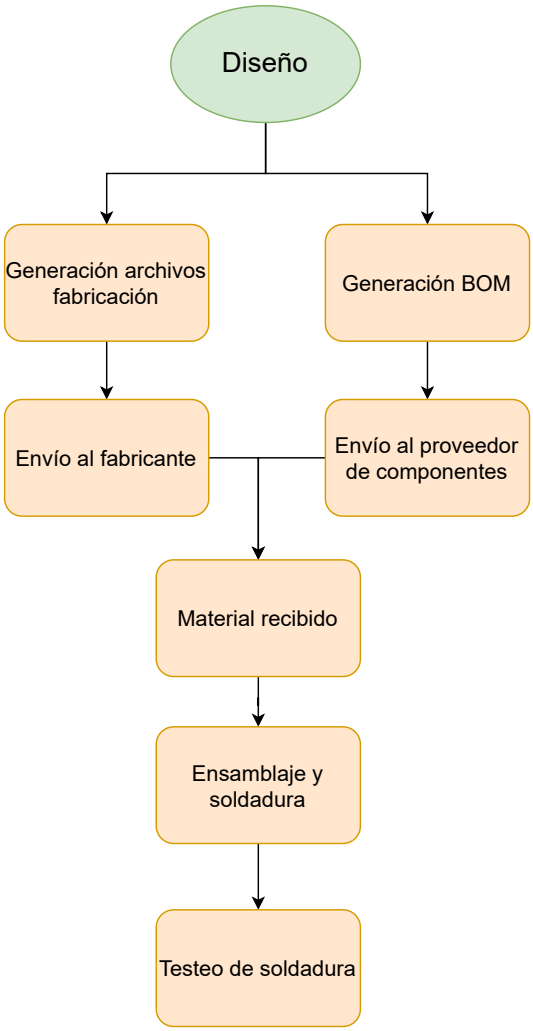


Figura 4.9 – *Metodología fabricación*

Para el proceso de montaje de las placas se van a usar las *stencils*.Se coloca la plantilla sobre su respectiva placa, ajustando todos los *pads* para que queden bien alineados, se fija para que no se mueva cuando se aplique la pasta de soldar. Con la ayuda de una espátula,se añade la pasta de soldar sobre todos los *pads*, una vez se añadido toda la pasta,figura 4.10, se procede a la colocación de los componentes.

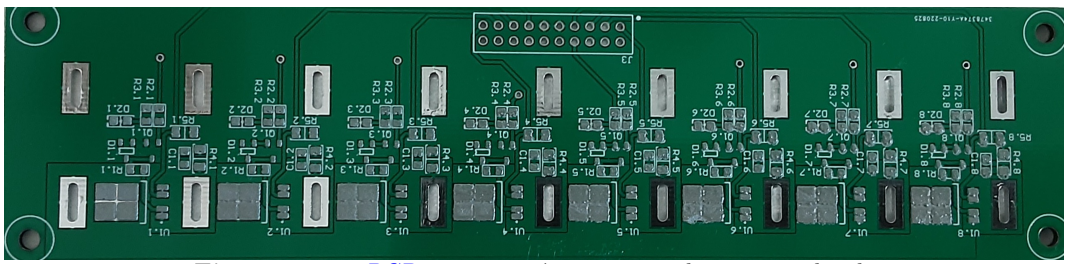


Figura 4.10 – *PCB conmutación con pasta de estaño aplicada*

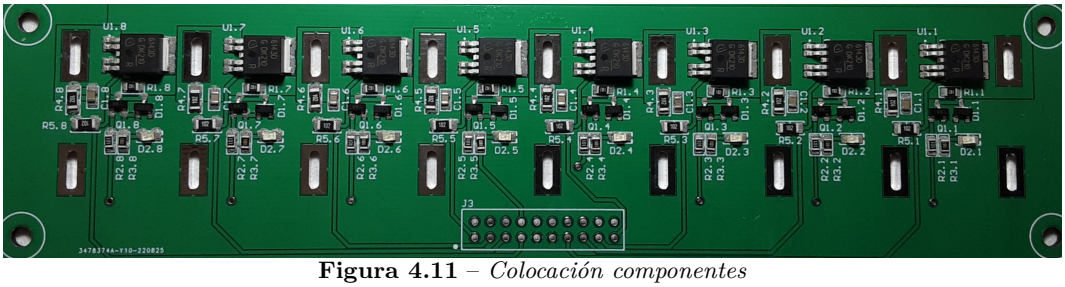


Figura 4.11 – *Colocación componentes*

Después de tener todos los componentes colocados en la placa como en la figura 4.11, se introduce en el horno para soldar los componentes a la placa por convección. El horno usado es el que se puede ver en la figura 4.12, como este horno no tiene ningún ciclo de temperatura programable (es un horno de cocina), inspeccionaremos de manera visual la soldadura. Se pone el horno a 250°C y activamos las resistencias inferiores y superiores y cuando se observe que el estaño empieza a fusionarse se espera hasta que esté completamente , se apaga el horno y abre la compuerta para que se enfríe.

(a) *Horno convección* **(b)**

Figura 4.12 – *Soldadura por convección*

Para la placa de conmutación la soldadura quedó perfecta,en cambio en la placa de control, algunos pines del MCU,se puede ver en la figura 4.13,quedaron cortocircuitados por exceso de estaño, pero se pudo quitar el exceso de estaño con el soldador sin problema

Figura 4.13 – *Exceso estaño MCU*

4.3. Diseño firmware

El firmware con el controlador CH559L consiste en el desarrollo de las siguientes funcionalidades :

- Algoritmo para generar las señales cuadradas de los diferentes ciclos. Los datos de estas señales, proporcionados por el cliente, se almacenan dentro del código.
- La comunicación con la pantalla para mostrar el menú y el *rotary encoder* para la selección del ciclo a generar por parte del ususario.
- Comunicación USB para controlar la generación de señales.

- Proceso de las lecturas de las entradas analógicos, que provienen de la placa con los interruptores.

```
#include "LightingSystem.h"
void main()
{
    Setup();
    USBInit();
    while (1)
    {
        SerialReadline();
        isPressEnterBootloader();
        handleChangeRotary();
        handleSignalStart();
        handleGetNextOutputs();
        handleRefreshOutputs();
    }
}
```

Listado 4.1 – *main.c*

4.3.1. Entorno desarrollo y estructura

Como se comenta en la sección 3.1.1, para el desarrollo del firmware y programación del MCU se hace uso del framework PlatformIO. A través de una extensión en VSCode se puede poner en marcha el proyecto de forma rápida, a través de un archivo de configuración **platformio.ini** seleccionamos la plataforma y microcontrolador a programar. Por suerte, **PlatformIO** tiene soporte para Intel MCS-51 (8051). Sin embargo, no tenemos un proceso para quemar nuestro código, para ello se hará uso de un *script* realizado en Python(añadir a referencias).

4.3.2. Estructura de memoria CH559L

En la figura 4.14 se muestra la estructura de memoria del microcontrolador CH559. El código se almacenará en el espacio de memoria reservado para éste, cualquier variable o puntero que se quiera almacenar en este espacio hay que añadirle **___code** en la declaración, el compilador interpretará la variable como una constante. Para las variables volátiles, que se sometan a cambios constantes se les asignará al espacio de memoria interno, si se usa *–model-small* [1] para compilar el código, las variables declaradas por defecto se asignarán en este bloque. Sin embargo, el código se ha compilado con el argumento *–model-large* así las variables declaradas sin declararles ningún espacio de memoria irán al espacio RAM externo. Es importante que al *linker* se le indique este argumento también, sino dará un error.

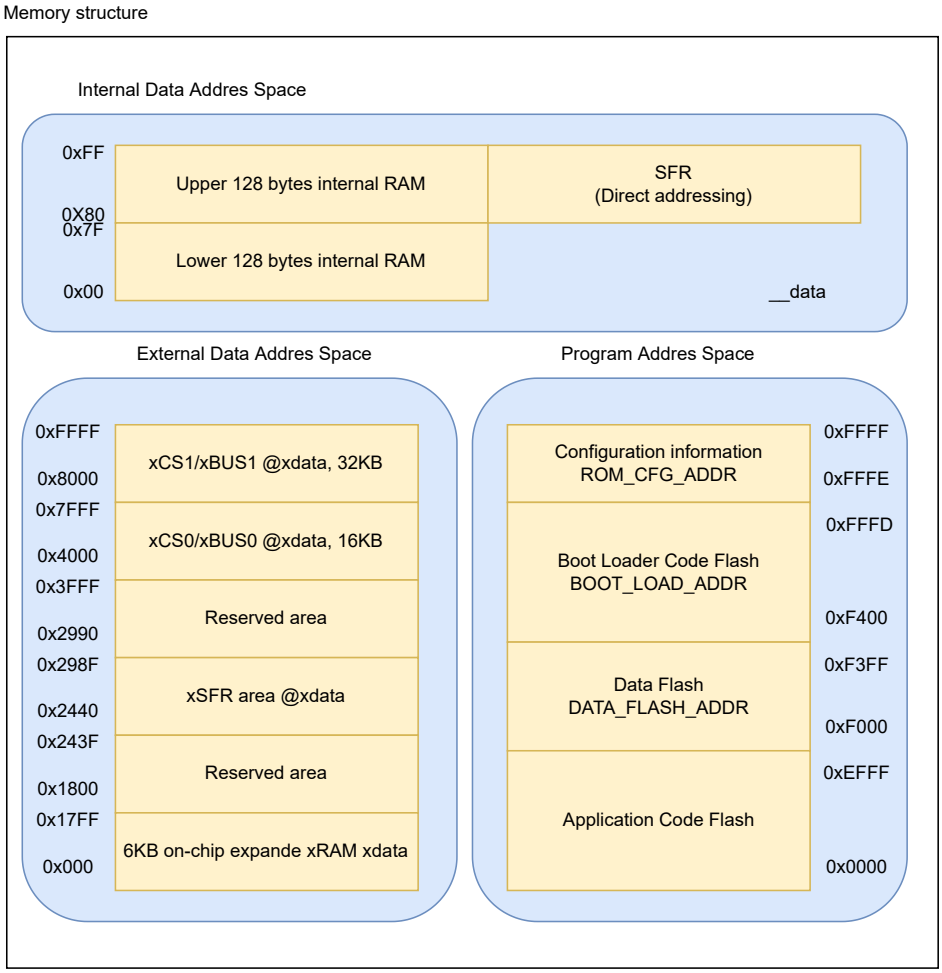


Figura 4.14 – *Estructura memoria CH559*

4.3.3. Flujo programa

Las siguientes fases describen el funcionamiento del programa. Se van a describir las interrupciones utilizadas en el programa, el algoritmo de generación de las señales, la escritura y lectura de la dataflash, valores analógicos y la comunicación USB.

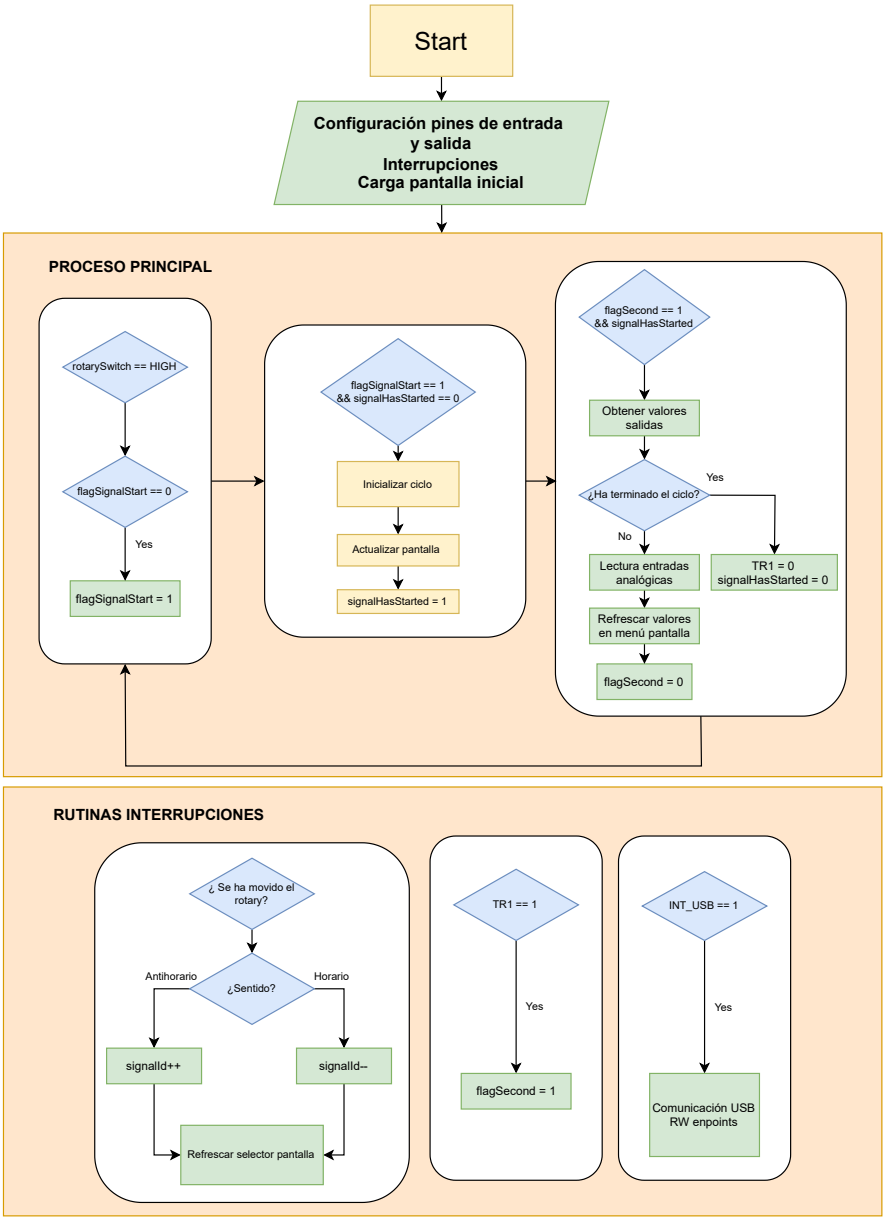


Figura 4.15 – *Flujo programa*

4.3.3.1. *Bootloader*

Para poder quemar la aplicación en la placa, hay que iniciar el *bootloader* del MCU, para ello el pin 46 del microcontrolador tiene que estar a GND cuando es alimentado. Una vez se ha iniciado el *bootloader*, hay dos métodos para quemar el código : por UART o USB. En *Windows* es necesario instalar Zadig y cambiar el *driver* a *libusb-win32*. Para subir el código UART es necesario un conversor *USB* a *TTL* como el de la figura 4.17 y conectarlo a los pines 0.2 (RXD) y 0.3 (TXD) del MCU.

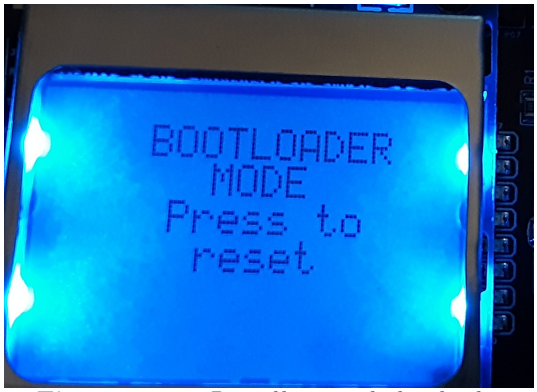
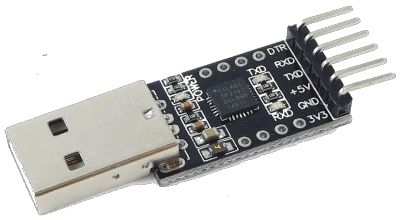


Figura 4.16 – *Pantalla entrada bootloader*

Para no tener que ejecutar el *bootloader* de la manera comentada arriba, se ha habilitado poder ejecutarlo desde la aplicación haciendo saltar la rutina del programa a la dirección de memoria en la que empieza a ejecutarse el bootloader y aparecerá un aviso de que se ha entrado en *bootloader*, como se ve en la figura 4.16.


```
Port COM6 57600 baud open.
Attempting to start the bootloader via the DTR line...
Found CH559
Flash size: 64 blocks, 65536 bytes.
Reserved for application: 60 blocks, 61440 bytes.
Bootloader version: 2.40
Flash Erased
Filesize: 20742 bytes
Flash size: 64 blocks, 65536 bytes.
Reserved for application: 60 blocks, 61440 bytes.
Bootloader version: 2.40
Flash Erased
Filesize: 20742 bytes
[====================] 100% Writing success
[====================] 100% Verify success
Closing COM6 port.
```

(a) *Log subida código por UART*



(b) *Conversor USB a TTL*

Figura 4.17 – *Subida código por UART*

4.3.3.2. Interrupciones

El microcontrolador CH559L tiene 14 interrupciones disponibles, 6 de ellas compatibles con el estándar MCS51 . En la tabla 4.1 aparecen las interrupciones usadas , su prioridad y la funcionalidad asociada.

Interrupción	N interrupción	Prioridad	Descripción
INT_NO_TMR0	1	1	Activa una *flag* que activará las salidas a su estado calculado anteriormente
INT_NO_GPIO	8	2	Procesa el movimiento del *rotary* para actualizar el estado
INT_NO_USB	12	3	Atiende la comunicación USB.

Tabla 4.1 – *Descripción interrupciones utilizadas*

La interrupción principal es la asociada al *timer* ya que se necesita la mayor precisión posible sobre el tiempo transcurrido. En la arquitectura encontramos con un registro dos registro el registro **TIMER0**, que se compone de dos bytes **TIMER0_MSB** y **TIMER0_LSB**, cuando se inicia el temporizador este registro incrementa y cuando se produce un desbordamiento ocurre la interrupción que ejecutará la rutina asociada, como se ve en 4.3.3.2 la función llamada en cada interrupción es **Timer0_ISR** que inicializa los registro del temporizador a los valores asignados en **Timer0_init**, para llamar a una función en un intervalo de tiempo determinado, se incrementa un contador que llamará a una función asociada en **Timer0_AttachInterrupt** llegado a ese tiempo. Para ajustar el intervalo en el que se produce la interrupción hay que inicializar los registros **TIMER0_MSB** y **TIMER0_LSB** a un valor determinado cuando se produce la interrupción [4].

```
/*Value to get interruption each milliseconds*/
uint16_t TimerDelay = (uint16_t)65535 - (1083/(1.085));
uint8_t  TIMER0_MSB  ;
uint8_t  TIMER0_LSB  ;
uint8_t  TIMER1_MSB  ;
uint8_t  TIMER1_LSB  ;

uint16_t interval_time=0;
uint16_t contador_ms=0;
isrPointerFunction Timer0_CallBack = NULL;
isrPointerFunction Timer1_CallBack = NULL;

void Timer0_init()
{
  TMOD |= 0x01;                          /* Timer0 mode1 */
    TIMER0_MSB   = (TimerDelay>>8) & 0xFF;        /**< Get MSB BYTE
 */
    TIMER0_LSB   = (TimerDelay)    & 0xFF;        /**< Get LSB BYTE
 */
  ET0  = 1;                                      /* Start timer0 */
}

void Timer0_AttachInterrupt(uint16_t millis,isrPointerFunction
 func_a_llamar){
    Timer0_CallBack = func_a_llamar;
    interval_time   = millis;
}
void Timer0_ISR() __interrupt 1
{
    TH0 = TIMER0_MSB;         /* 1ms timer value */
  TL0 = TIMER0_LSB;
    contador_ms++;  /* Increment milliseconds counter */
    /*When it reaches interval_time run callback function an reset
 contador_ms*/
    if(interval_time < contador_ms)
    {
        if(Timer0_CallBack!=NULL)
        {
            Timer0_CallBack();
        }
        contador_ms = 0;
```

```
    }
}
```

Listado 4.2 – *Funciones de configuración interrupción asociada TIMER0 pertenecen a timer.c*

4.3.3.3. USB

Uno de los requisitos 2.2, es que el dispositivo sea reconocido como un dispositivo CDC, así cuando lo conectemos a un ordenador será reconocido como un puerto serie, lo que facilitará la comunicación y lectura de datos.

Descriptor	Valor
VIP	C750
PID	1209
Produce String	Automotive Lighting System
Serial String	SN000001
CDC	CDC Serial
Manufactor	GranaSAT

Tabla 4.2 – *Descriptores USB*

Configurar esto ha sido uno de los mayores retos de este proyecto, aunque el fabricante da algunos ejemplos sobre la comunicación USB, no llega a proporcionar una librería compacta para esta parte y además el código que proporciona es para el compilador Keil, la sintaxis y tipos de variables de datos cambian entre SDCC y éste, pero con la ayuda de varios repositorios de la comunidad [3] y adaptando el código a este compilador se ha conseguido el objetivo. En la figura 4.18, se pueden ver los registro al conectar la placa por USB, como se observa aparecen los campos de los descriptores que aparecen en la tabla 4.2.

```
[  762.028209] usb 1-4.3: new full-speed USB device number 7 using xhci_hcd
[  762.231001] usb 1-4.3: New USB device found, idVendor=1209, idProduct=c750, bcdDevice= 1.00
[  762.231005] usb 1-4.3: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[  762.231006] usb 1-4.3: Product: Automotive Lighting System
[  762.231007] usb 1-4.3: Manufacturer: GranaSAT
[  762.231008] usb 1-4.3: SerialNumber: SN000001
[  762.255303] cdc_acm 1-4.3:1.0: ttyACM0: USB ACM device
```

Figura 4.18 – *Log comando desmeg en Linux*

4.3.3.4. Generación ciclos

Cada ciclo está formado por una o varias señales diferentes concatenadas. En memoria se organizan los ciclos por arrays,tabla 4.3, donde cada elemento del es una señal con la estructura de datos de la tabla 4.4. Cada ciclo va asociado a un determinado canal , hay 8 disponibles. Se podrá seleccionar pulsando el interruptor del *rotary* como se muestra en la figura 4.19.

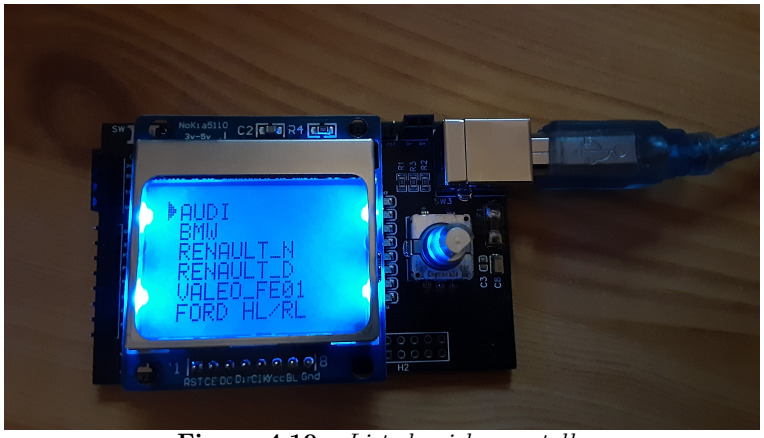


Figura 4.19 – *Listado ciclos pantalla*

Las entradas al proceso de generación de las señales de un ciclo son la dirección de memoria donde comienza el ciclo y el número de señales que lo componen, a partir de aquí se va procesando cada señal actualizando los estados de las salidas en cada intervalo de tiempo establecido. Las variables de tiempo de las señales están en segundos.

La estructura de datos de un ciclo

ID (2 Bytes)	S0 (12 Bytes)	S1 (12 Bytes)	S2(12 Bytes)	...	SN (12 Bytes)

Tabla 4.3 – *Estructura de un ciclo*

Nombre	NBytes	Datatype	Descripción
initValue	2	*unsigned short*	Estado señal durante el retardo
timeDelay	2	*unsigned short*	Retardo señal
onValue	2	*unsigned short*	Valor inicial señal
timeOn	2	*unsigned short*	Tiempo en *onValue*
timePeriod	2	*unsigned short*	Periodo señal
nCycles	2	*unsigned short*	Número de ciclos

Tabla 4.4 – *Estructura señal*

```
2 __code unsigned short C2_0[36] =
    {
      0, 120, 1, 5, 10, 3,
5     0,  30, 1, 5, 10, 3,
      0, 150, 1, 5, 10, 3,
      0,  30, 1, 5, 10, 3,
8     0,  60, 1, 5, 10, 3,
      0,  30, 1, 5, 10, 3
      };
```

Listado 4.3 – *Ejemplo de ciclo en C*

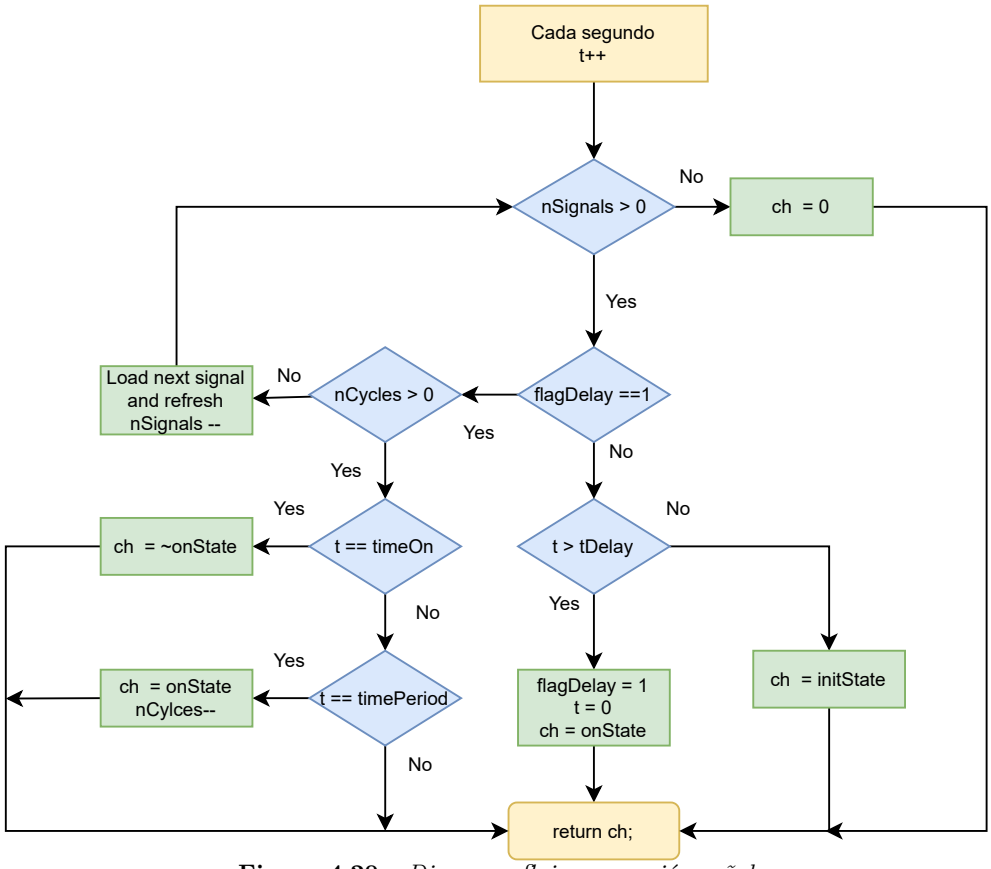


Figura 4.20 – *Diagrama flujo generación señal*

4.3.3.5. Dataflash

El microcontrolador CH559, nos ofrece un *kilobyte* de memoria ROM, aparece como *Dataflash* en la figura 4.14. Este espacio de memoria será usado para guardar de forma no volátil información de configuración como puede ser el número de serie del producto, la fecha de fabricación. También se usa para permitir al usuario almacenar un conjunto de señales mediante USB, que posteriormente podrá generar seleccionándolo en el menú.

4.3.3.6. Lectura valores analógicos

CH559L proporciona 8 conversores analógicos digitales de aproximaciones sucesivas de 10 u 11 bits. Las principales características son :

- Resolución de 10 u 11 bits.
- Rango de 0 a 3.3 V.
- Máximo de 1MSPS (un millón de muestras por segundo).

Cuando se están generando los ciclos, en la pantalla se muestra el voltaje de cada canal.

Capítulo 5

Sistema de verificación y pruebas

En este capítulo se va a mostrar el sistema una vez montado en el laboratorio, se proporcionarán imágenes del entorno de trabajo empleado y la utilización de los diferentes elementos.

Una vez realizado el prototipo de ambas placas se pondrán en marcha para verificar el comportamiento de las diferentes partes de la electrónica. Se realiza la conexión de las salidas y entradas entre las respectivas placas.El sistema se alimenta a través de unos conectores banana en el producto real, para alimentar la placa se han soldado dos cables en los *pads* que corresponden a la alimentación del sistema y se procede a conectar una fuente de alimentación regulable y ajustar la para proporcionar un voltaje de 12V.

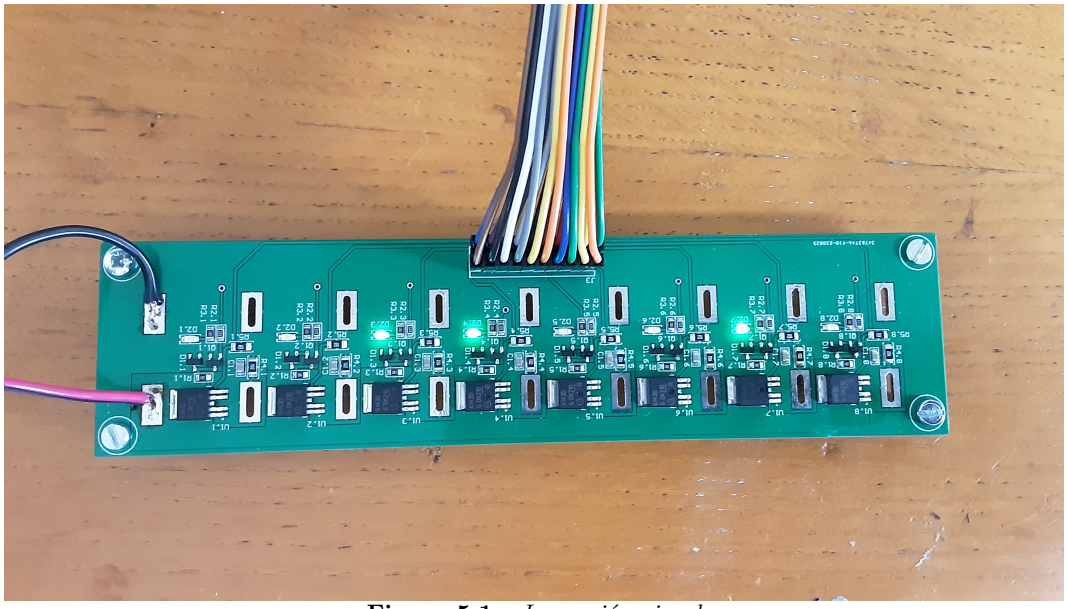


Figura 5.1 – *Inspección visual*

Posteriormente, se han soldado varios cables en un par de salidas, que irán conectados a las diferentes luces de los faros. En una misma salida, podemos tener conectados en paralelo las diferentes entradas de iluminación de un faro 5.3, así se podrá estimar el consumo total de éste.

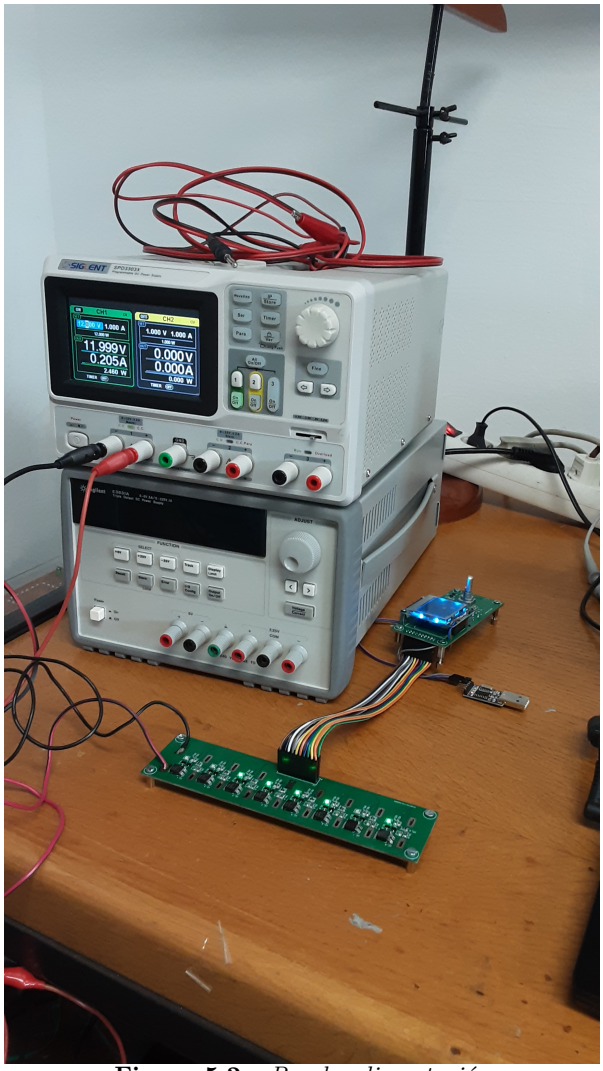


Figura 5.2 – *Prueba alimentación*

Por defecto, todas las salidas de conmutación tienen en la salida una resistencia de 1 $k\Omega$, si se encuentran todas las salidas en alto, se estima un consumo de 960 mA más lo consumido por la placa de control. Una vez alimentado el sistema , se procede a una subida de firmware para comprobar que el sistema se puede actualizar correctamente. Cuando la aplicación se haya cargado, se reinicia y aparecerán las señales para su selección.

Figura 5.3 – *PCB conectada faro.*

En las siguientes figuras 5.4 5.5, se muestran dos ejemplos de salidas generadas.

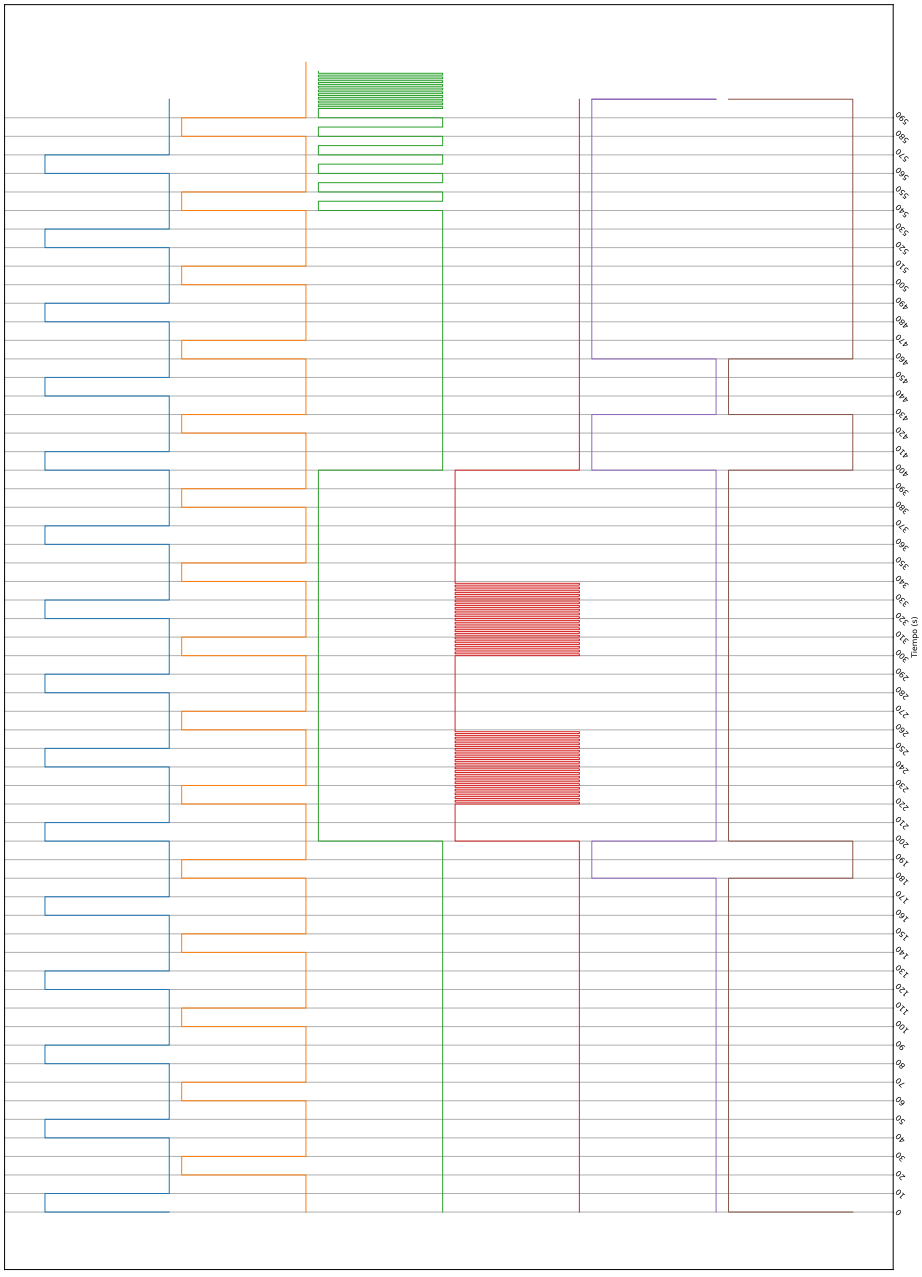


Figura 5.4 – *Ejemplo señal generada 1.*

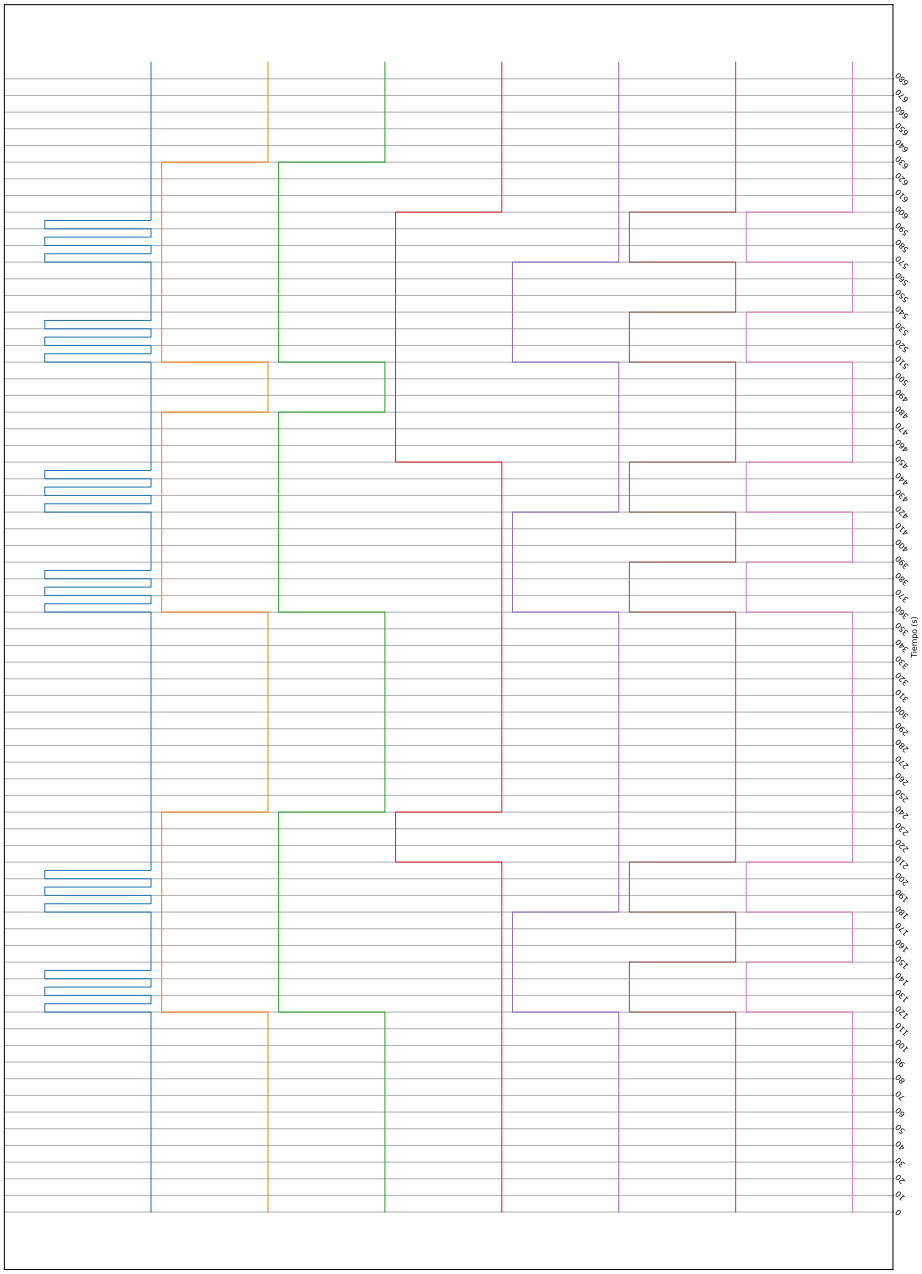


Figura 5.5 – *Ejemplo señal generada 2.*

Capítulo 6

Conclusiones y líneas futuras

6.1. Conclusiones

Durante el desarrollo de este proyecto se han presentado muchos retos que han ayudado al aprendizaje y desarrollo en varias disciplinas, ya que se combina software, hardware, el ensamblaje de varias placas con una carcasa. Sin duda, el desarrollo del firmware es lo que más problemas ha presentado, no por la funcionalidades a desarrollar, sino por el microcontrolador elegido para llevar acabo estas funcionalidades. La elección de un microcontrolador totalmente diferente a lo que estaba acostumbrado,como puede ser alguno de AVR o Microchip, con una arquitectura diferente, un entorno de trabajo diferente ha ocasionado en muchas ocasiones frustración, pero de cada problema afrontado, cada error y cabezazo contra el teclado se ha aprendido algo nuevo. En otro contexto, por ejemplo, en una empresa u otro proyecto no me atrevería a elegir un microcontrolador nuevo, ya que conllevaría un tiempo considerable en aprender la nueva tecnología y siempre los riesgos de error son mayores. Una de las cosas que tenía claro cuando elegí el proyecto, es que quería desarrollar en C, quería tener más contacto con la programación a bajo nivel, para aprender todo eso que no ves cuando programas con algún *framework* o librerías que ya tienen implementadas todas las funcionalidades que necesitas. Me encuentro satisfecho con la elección,aunque me haya costado, porque he cumplido con los objetivos propuestos.

Por otro lado, es la primera vez que me enfrento un producto con dos placas integradas en el mismo diseño mecánico, esto hace que durante el diseño de la placa se esté considerando en todo momento el diseño mecánico y como puede afectar, ya que cualquier despiste en alguna dimensión o error en la colocación de algún componente que interaccione con la parte mecánica puede hacer que tengas que volver a plantear gran parte del diseño.

6.2. Líneas futuras

En este apartado se van a plantear ciertas mejoras u otras líneas en las que enfocar el desarrollo de próximas versiones.

- Desarrollar un API integrada en el dispositivo con un protocolo de comunicación específico, para poder controlar todos los elementos y comunicarse con el dispositivo. Por lo tanto que sería posible interactuar con el dispositivo de diferentes maneras y en diferentes lenguajes de programación. Los ciclos podrían almacenarse en el dispositivo o generarse recibiendo los datos externamente. El desarrollo de esta API conllevaría no tener que ofrecer o desarrollar otra herramienta al cliente para comunicarse. El cliente podría decidir desarrollar la comunicación con el lenguaje que quisiera y poder crearse sus propios métodos e integraciones con el producto.

- Ampliar la memoria de almacenamiento, algún chip de memoria para poder almacenar ciclos y configuración necesaria.
- Se podría dotar al dispositivo de una capa física *ethernet*, esto conllevaría cambiar a un microcontrolador mucho más caro y sobredimensionado para el objetivo final, que es conmutar unos canales y leer valores, pero sería interesante proporcionar un módulo WiFi con el que poder comunicarse con un servidor donde se darían de alta los diferentes dispositivos y poder gestionarlos a través de un interfaz web o una aplicación de escritorio, así los usuarios podrían compartir los diferentes ciclos generados en una base de datos , monitorizar los dispositivos y tener un registro de todas las pruebas realizadas.

6

Bibliografía

[1] Dutta, S., and developers, S. Small device compiler, 2022.

[2] SolidWorks. Solidworks logo.

[3] Sun, D. ch55xduino.

[4] WCH. Datasheet ch559, 2014.

Apéndice A

Presupuesto proyecto

A.1. Hardware

A.1.0.1. Instrumentación

Elemento	**Precio [€]**
Analizador lógico	10
Estación soldadura	80
Osciloscipio	180
Multimetro	20
Horno convección	80
Elementos soldadura	50
Total	420

Tabla A.1 – *Coste total de la instrumentación y herramientas usadas para el desarrollo*

A.1.0.2. Fabricación

Elemento	**Precio [€]**
2 layer PCB	65
Stencils	20
Componentes SMD	100
Conectores	10
Carcasa	10
Total	205

Tabla A.2 – *Coste total fabricación y ensamblaje*

A.2. Recursos humanos

Elemento	**Precio [€]**
Ingeniero Junior	7500
Ingeniero Senior	6000
Total	13500

Tabla A.3 – *Coste recursos humanos*

Apéndice B

BOM

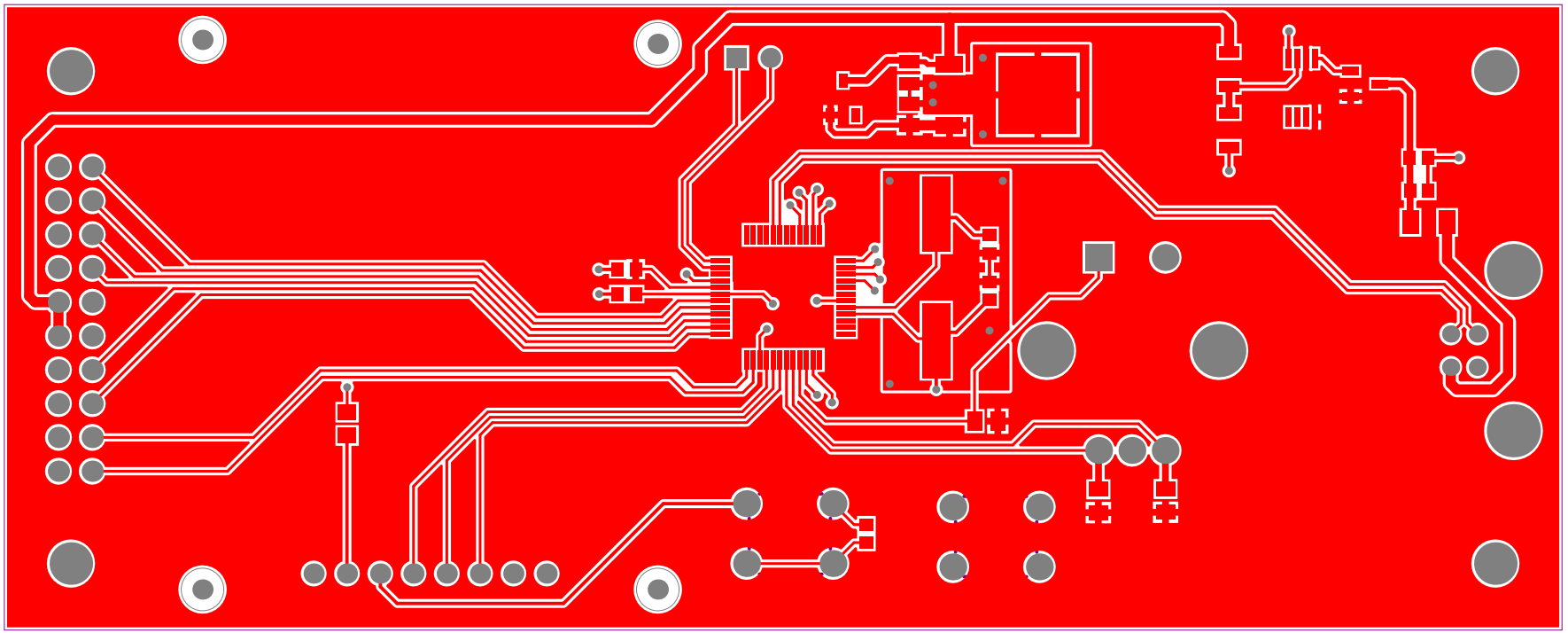
Quantity	Manufacture Part Number	Manufacturer(optional)	LCSC Part Number(optional)
8	BTS6143D	Infineon	C95232
16	HP06W2J0102T5E		C2759097
20	"BAS21,215"	Nexeria	C8661
20	"2N7002P,215"	Nexeria	C81488
10	ERA6AEB101V	PANASONIC	C445646
10	AR05DTCW1001	Viking Tech	C319814
10	LTST-C170TGKT	Lite-On	C364562
50	AS05W2J0182T5E	UNI-ROYAL	C966136
50	GCM21BL81E334KA37L	Murata Electronics	C161258
10	1-1825910-0	TE Connectivity	C2928193
10	LM358DGKR		C33696
10	12T.05000063AG12S1B02		C140474
20	0805X335K250CT		C296080
50	CC0805ZRY5V8BB104		C519930
50	TMK212BBJ106KGHT		C386084
20	ARG05BTC3300		C2984456
50	AR05BTC1002		C189482
20	FDN304P		C347509
2	PEC11R-4015F-S0024		C143791
5	U-USBBR04P-F001		C386743
10	CJ78M05		C9070
5	CH559L	WCH	C150548

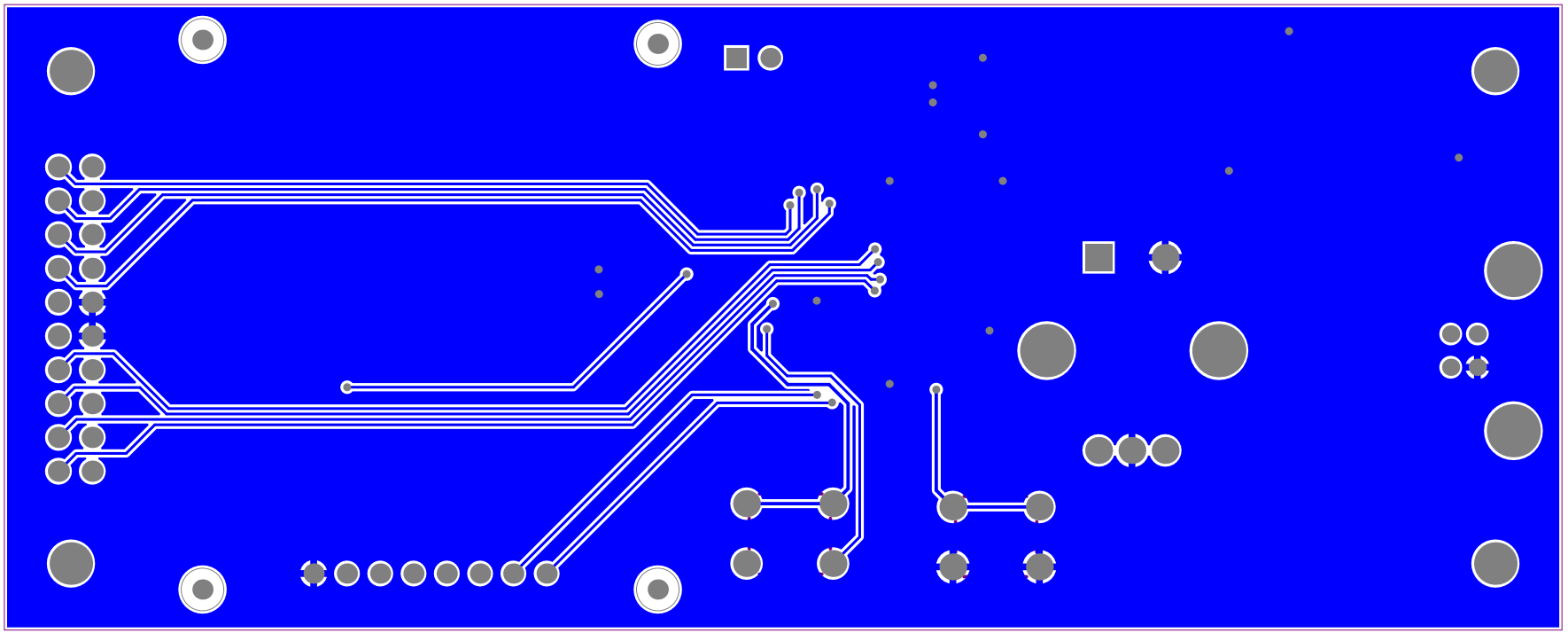
Tabla B.1 – *BOM*

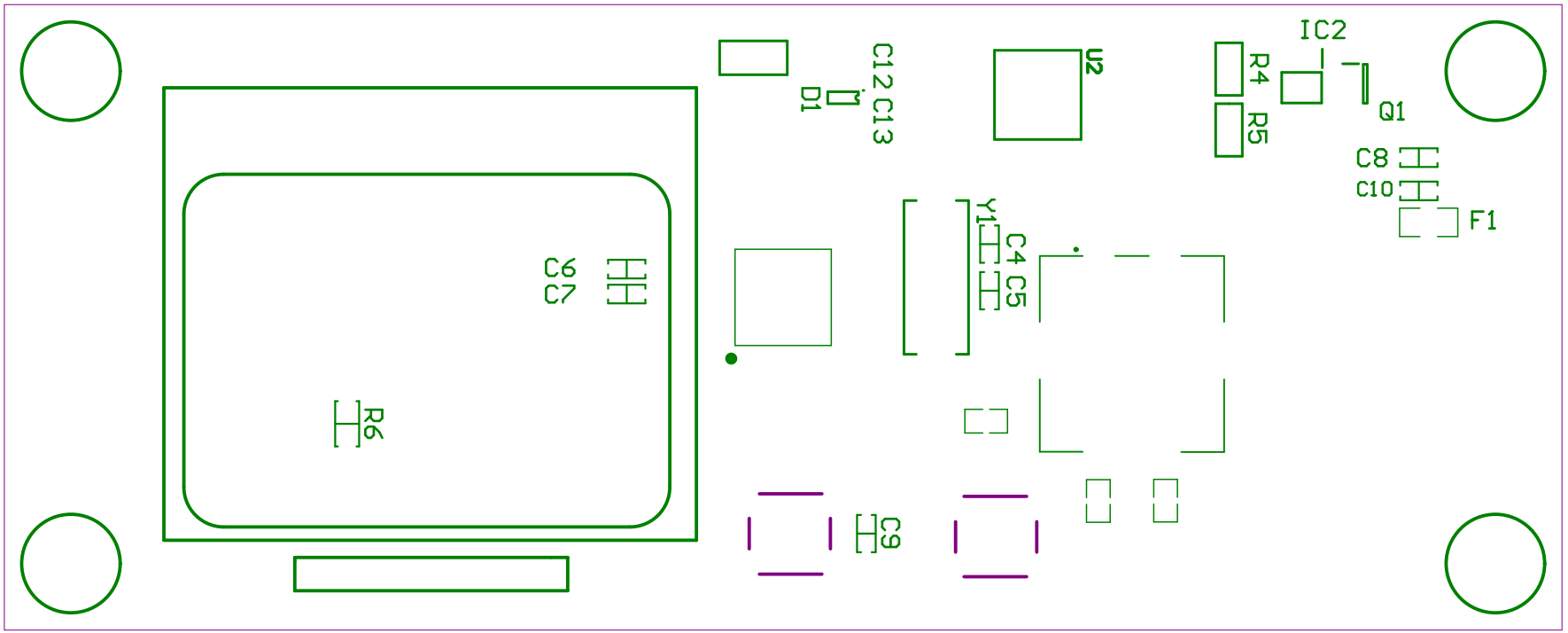
+

Apéndice C

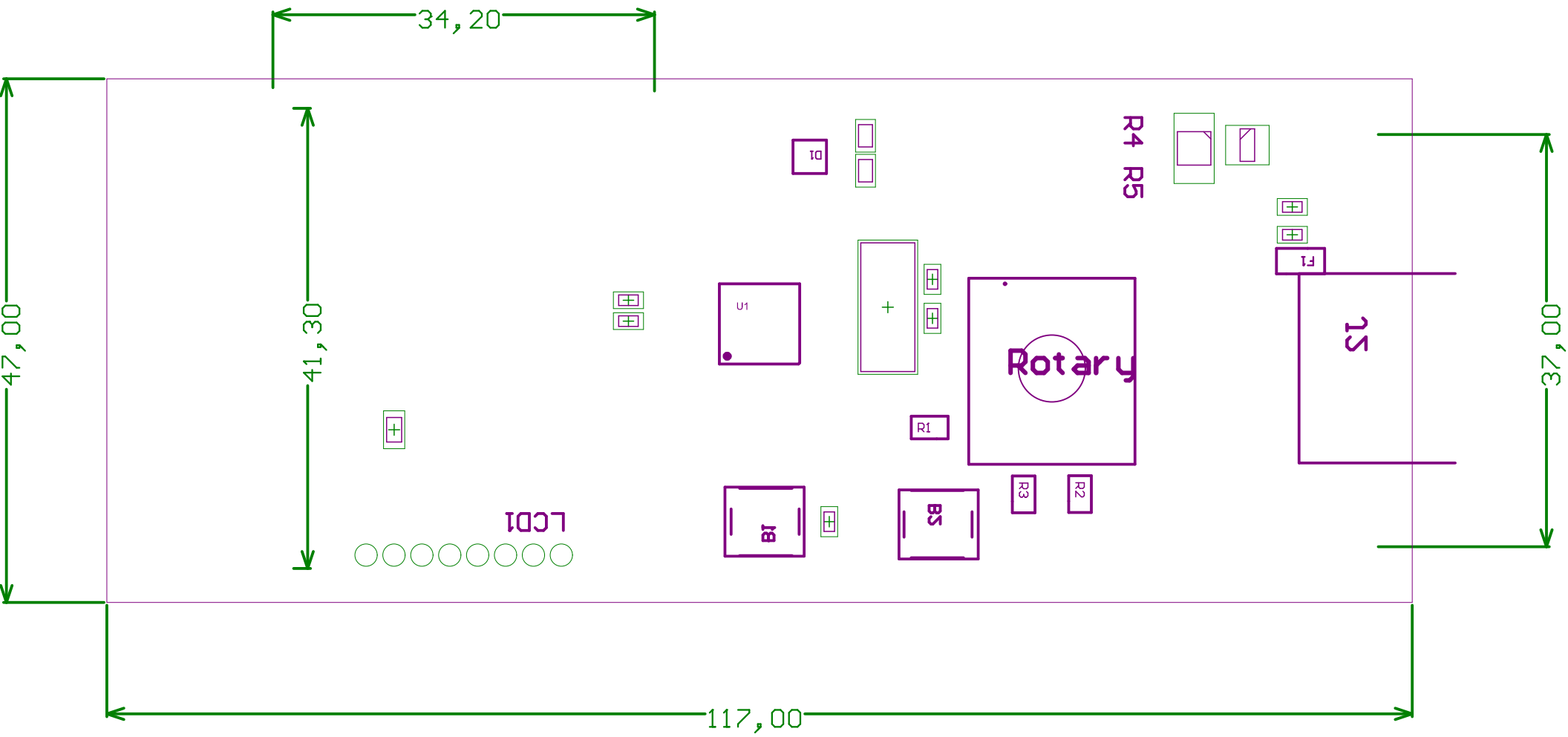
Archivos fabricación . Gerber

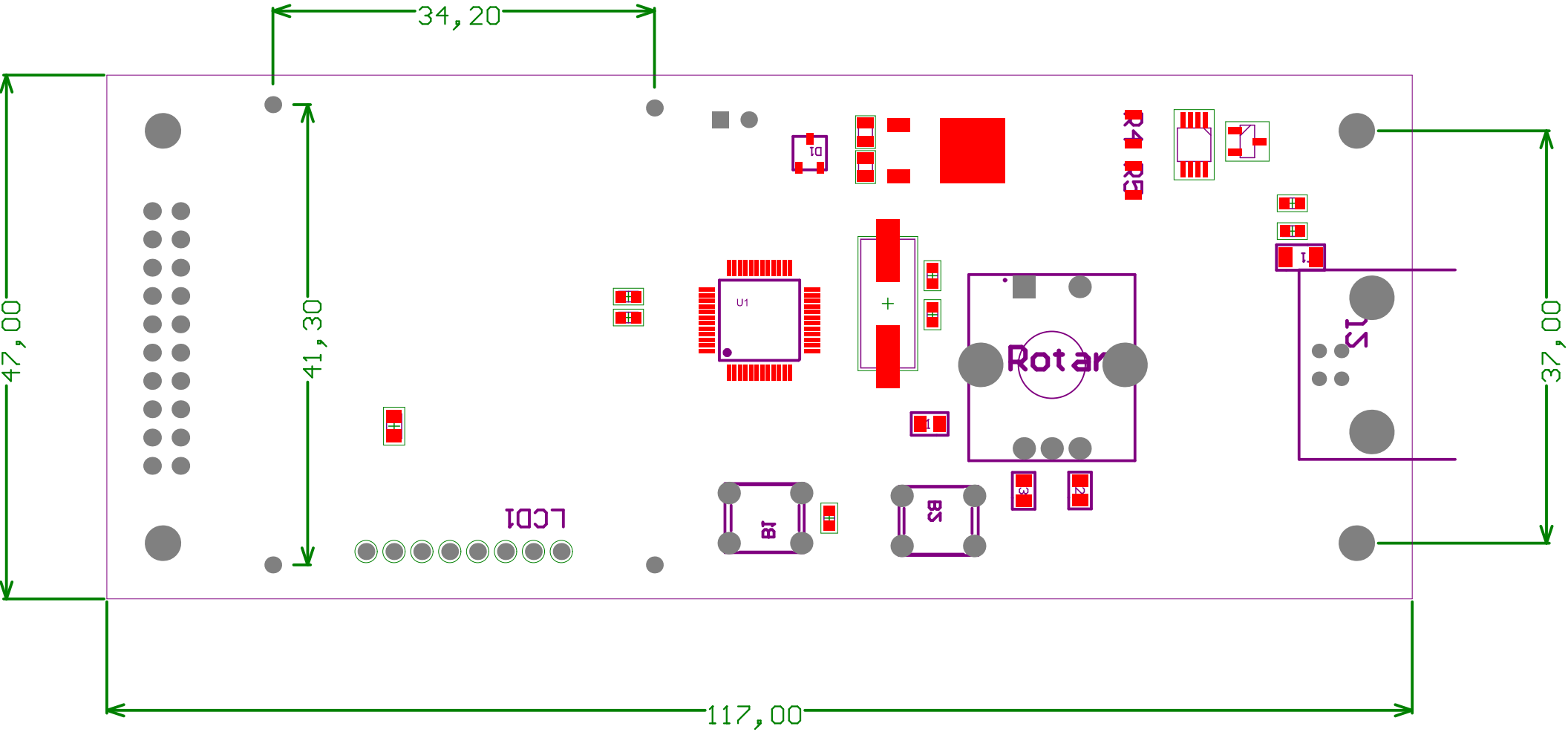


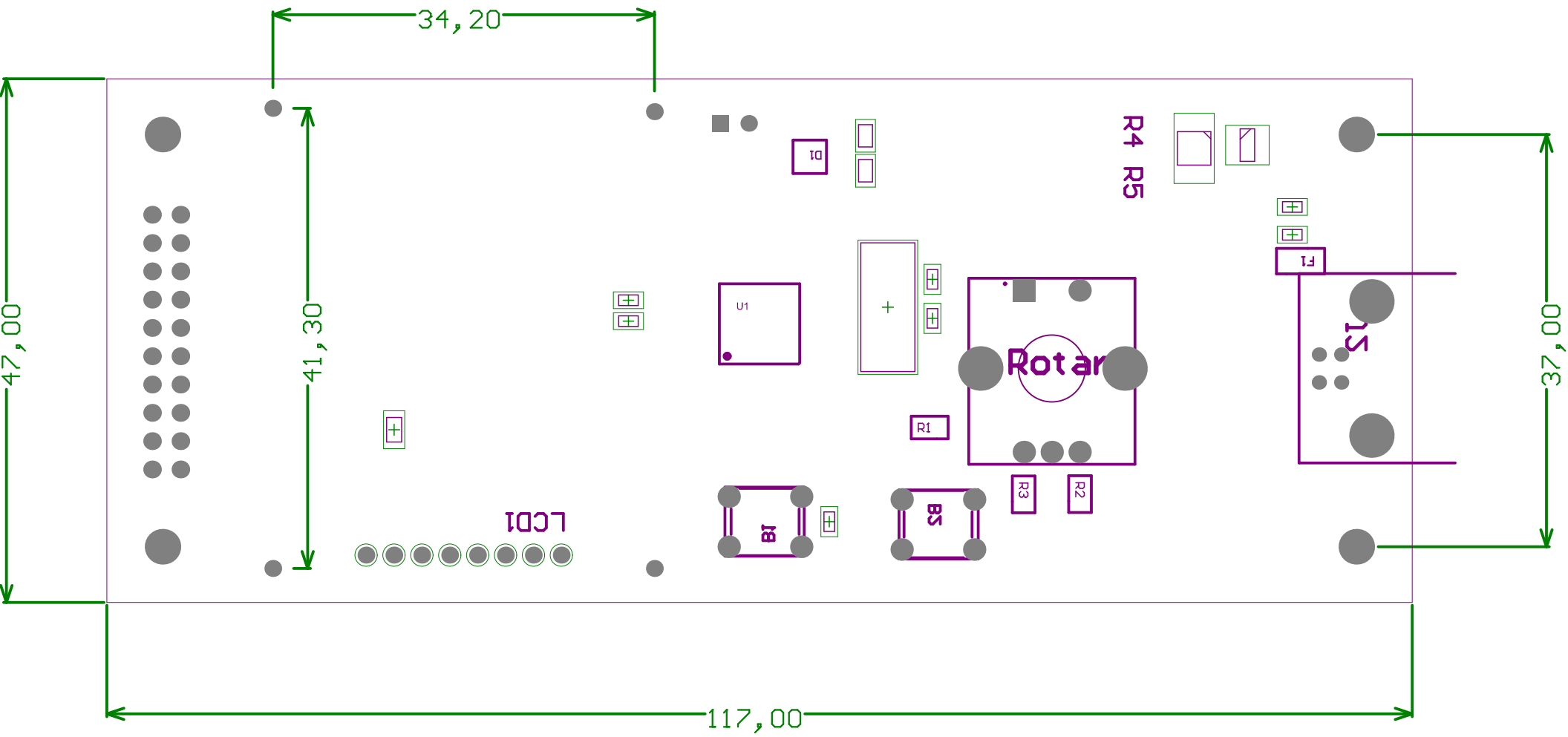


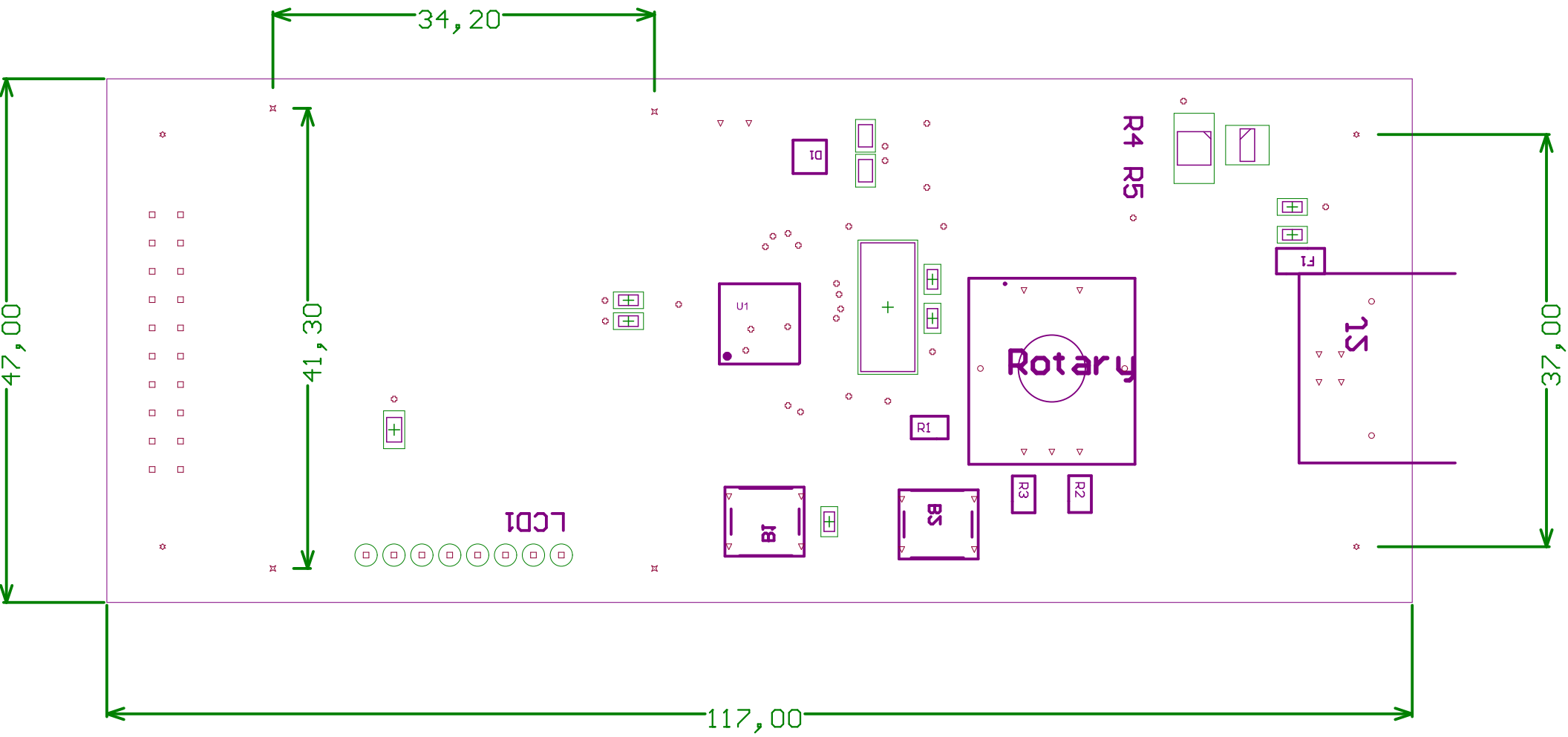

IC2
U2
R4
R5
Q1
C12
C13
D1
C8
C10
F1
Y1
C4
C5
C6
C7
R6
C9

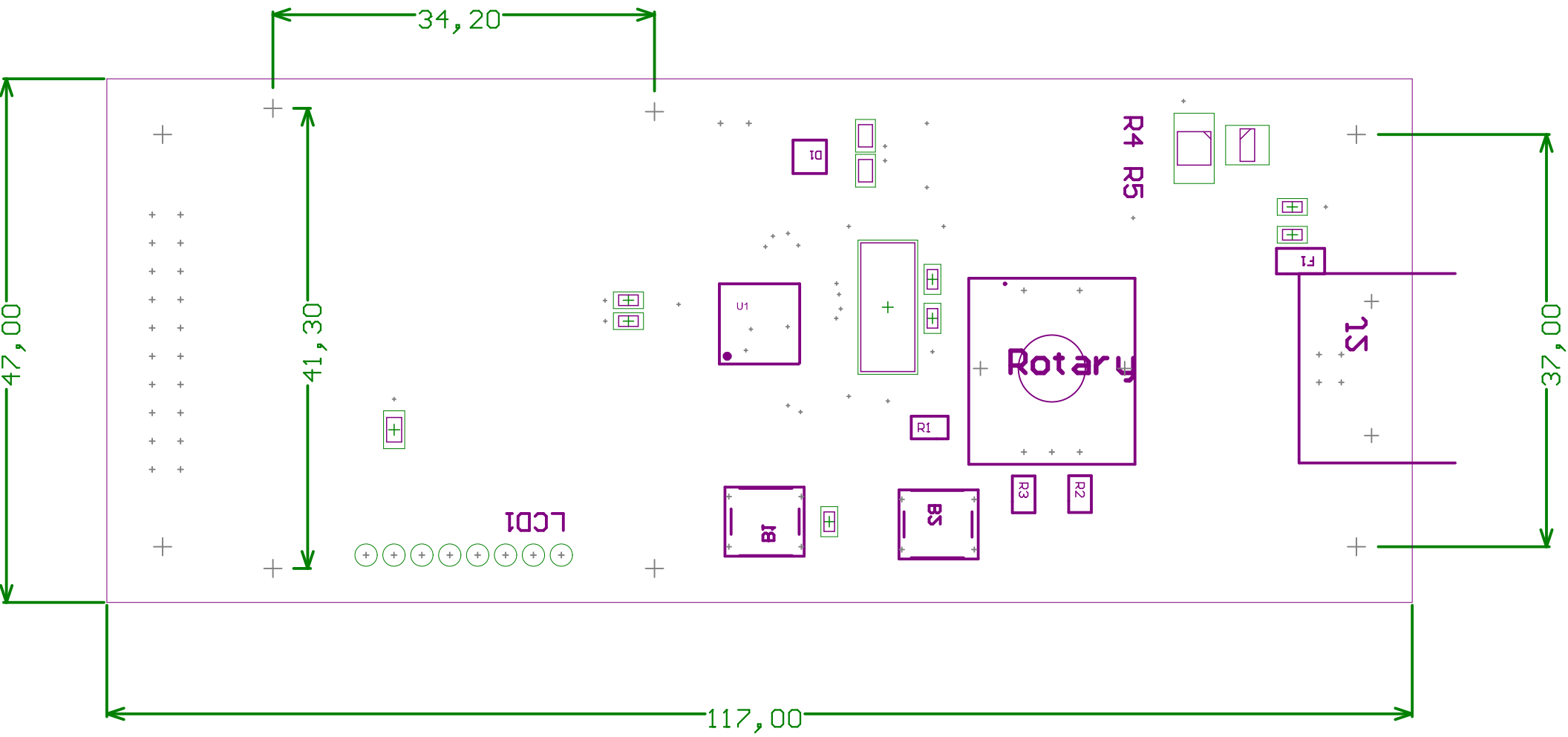
RX TX
P1
J2
B1
B2

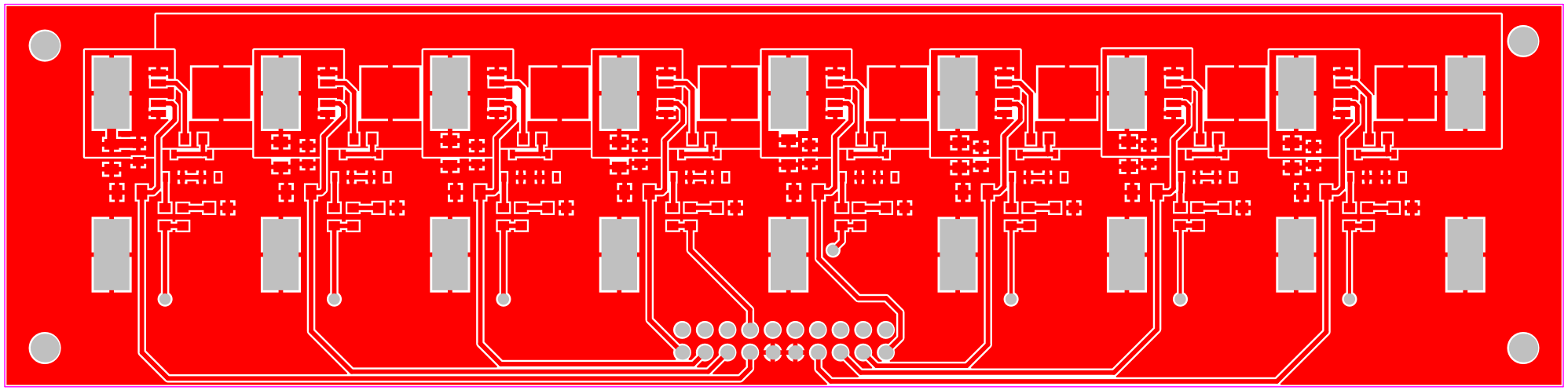

34,20
47,00
41,30
37,00
117,00
D1
R4 R5
U1
F1
J1
Rotary
R1
R3
R2
LCD1
B1
S1

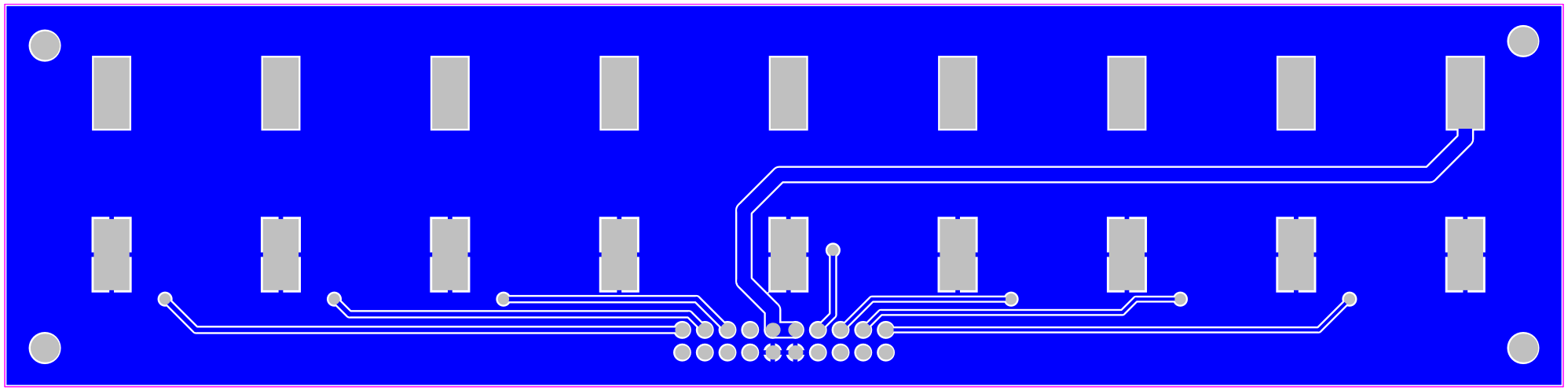

34,20
47,00
41,30
37,00
117,00
LCD1
U1
D1
R4
R5
Rotar
J2
B1
B2

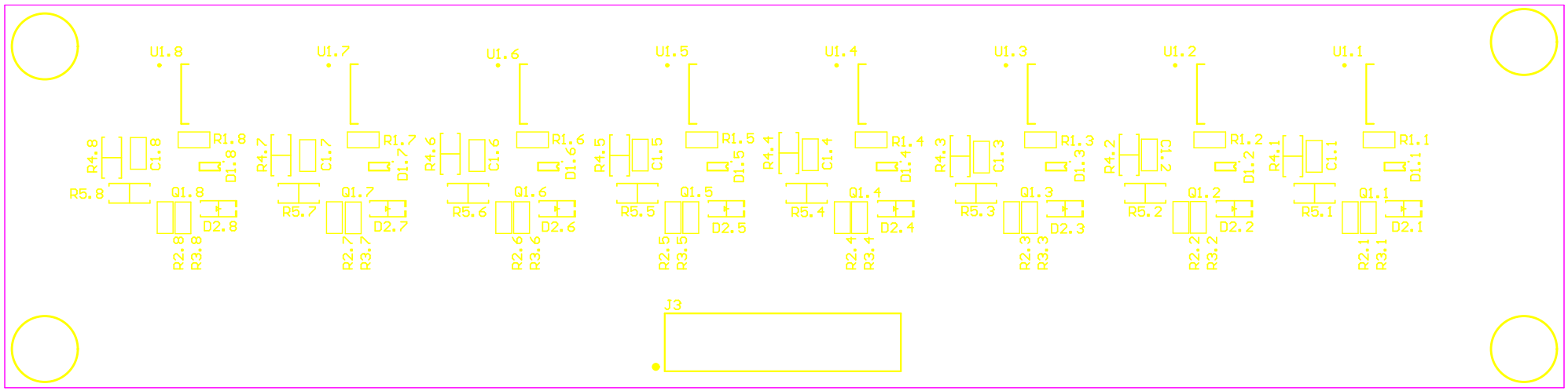

34,20
47,00
41,30
37,00
117,00
D1
R4
R5
U1
Rotar
R1
R3
R2
F1
LCD1
B1
B2


34,20
47,00
41,30
37,00
117,00
D1
U1
R4
R5
F1
J1
Rotary
R1
R3
R2
LCD1
B1
B2

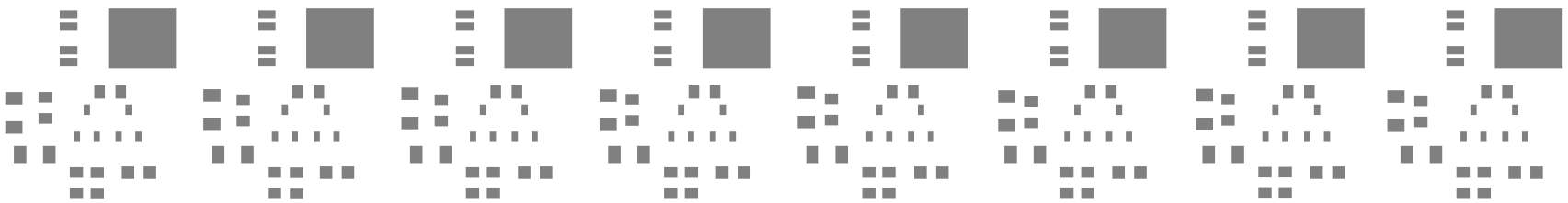

34,20
47,00
41,30
37,00
117,00
D1
U1
R4
R5
F1
J1
Rotary
R1
R3
R2
B1
B2
LCD1

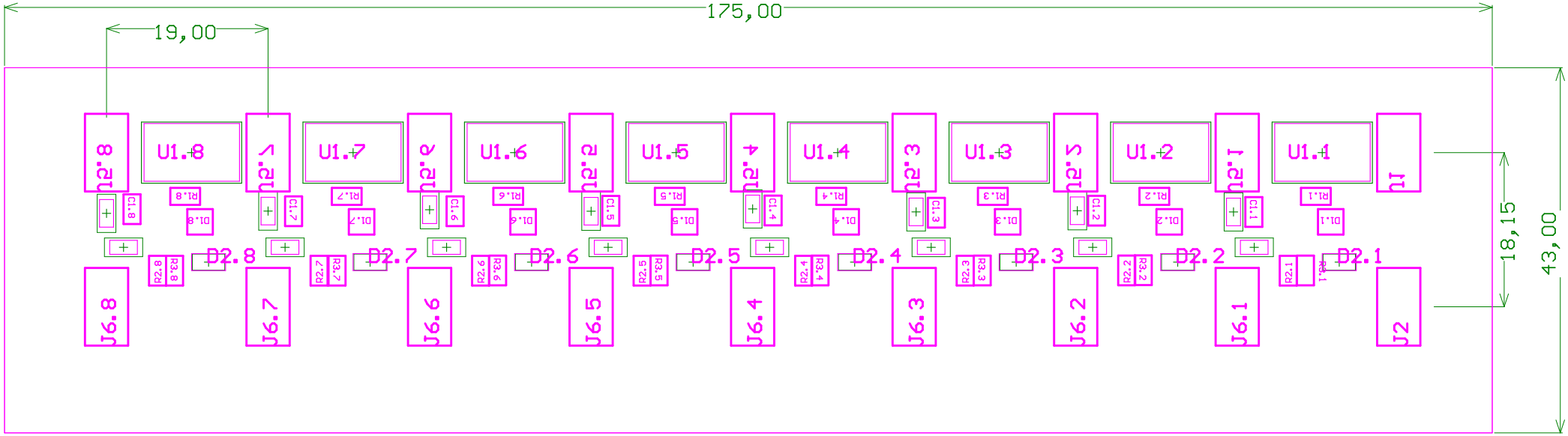


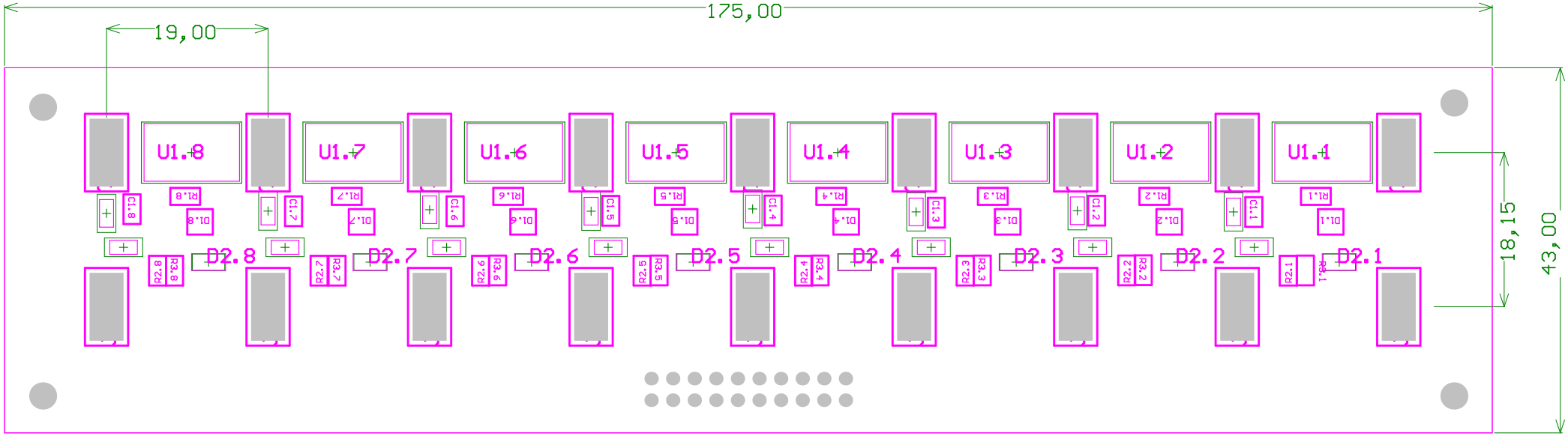


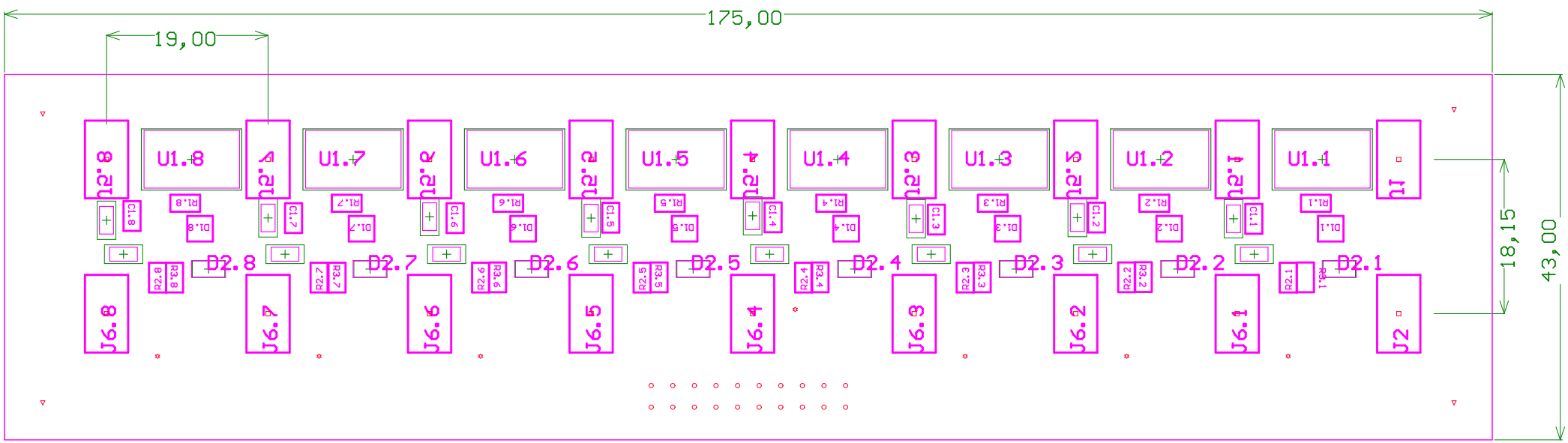

U1.8
U1.7
U1.6
U1.5
U1.4
U1.3
U1.2
U1.1
R1.8
R1.7
R1.6
R1.5
R1.4
R1.3
R1.2
R1.1
R4.8
R4.7
R4.6
R4.5
R4.4
R4.3
R4.2
R4.1
C1.8
C1.7
C1.6
C1.5
C1.4
C1.3
C1.2
C1.1
D1.8
D1.7
D1.6
D1.5
D1.4
D1.3
D1.2
D1.1
R5.8
R5.7
R5.6
R5.5
R5.4
R5.3
R5.2
R5.1
Q1.8
Q1.7
Q1.6
Q1.5
Q1.4
Q1.3
Q1.2
Q1.1
D2.8
D2.7
D2.6
D2.5
D2.4
D2.3
D2.2
D2.1
R2.8
R3.8
R2.7
R3.7
R2.6
R3.6
R2.5
R3.5
R2.4
R3.4
R2.3
R3.3
R2.2
R3.2
R2.1
R3.1
J3

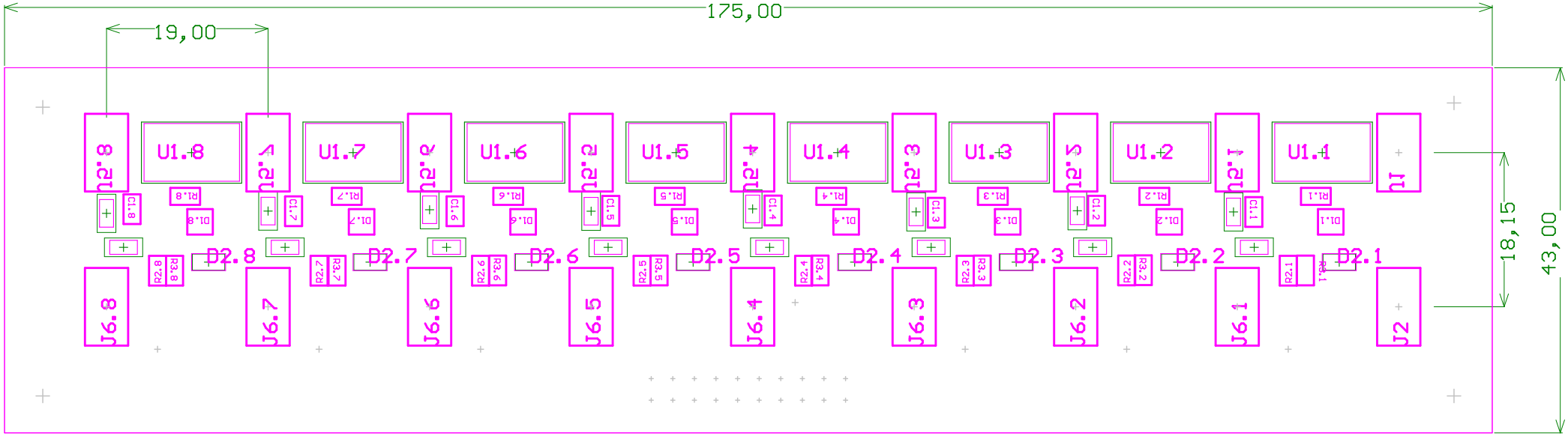
11	15.1	15.2	15.3	15.4	15.5	15.6	15.7	15.8
12	16.1	16.2	16.3	16.4	16.5	16.6	16.7	16.8




175,00
19,00
18,15
43,00
U1.8
U1.7
U1.6
U1.5
U1.4
U1.3
U1.2
U1.1
D2.8
D2.7
D2.6
D2.5
D2.4
D2.3
D2.2
D2.1
J6.8
J6.7
J6.6
J6.5
J6.4
J6.3
J6.2
J6.1
J2


175,00
19,00
18,15
43,00
U1.8
U1.7
U1.6
U1.5
U1.4
U1.3
U1.2
U1.1
D2.8
D2.7
D2.6
D2.5
D2.4
D2.3
D2.2
D2.1


175,00
19,00
18,15
43,00
U1.8
U1.7
U1.6
U1.5
U1.4
U1.3
U1.2
U1.1
D2.8
D2.7
D2.6
D2.5
D2.4
D2.3
D2.2
D2.1
J6.8
J6.7
J6.6
J6.5
J6.4
J6.3
J6.2
J6.1
J2


175,00
19,00
18,15
43,00
U1.8
U1.7
U1.6
U1.5
U1.4
U1.3
U1.2
U1.1
D2.8
D2.7
D2.6
D2.5
D2.4
D2.3
D2.2
D2.1
J6.8
J6.7
J6.6
J6.5
J6.4
J6.3
J6.2
J6.1
J2