

INSTITUTO TECNOLÓGICO DE BUENOS AIRES

22.31 - DISEÑO DE EQUIPOS ELECTRÓNICOS

19 DE FEBRERO DE 2024

---

## Proyecto final de Ingeniería Electrónica

Controller Area Network Data Logger Electronics  
**CANDLE**

---

Alumnos:

Matías BERGERMAN (60831)  
Xi LIN (59780)  
Pablo SMOLKIN (59523)  
María Luz STEWART HARRIS (57676)

Profesores:

Manuel Esteban CARRILLO DEL PINO  
Walter Hugo ORCHESSI  
Ignacio GASPARINI  
Pablo Fabian REGAZZONI

## **2. Agradecimientos**

A nuestras familias, que nos enseñaron el valor del esfuerzo y la educación, y nos apoyaron en todo nuestro crecimiento.

A nuestros amigos, compañeros de vida, por acompañarnos en este trayecto.

Al ingeniero Nicolás Magliola y al doctor Pablo Cossutta por acompañarnos como tutores en el trayecto de la realización del proyecto.

Al ingeniero Diego Ismirlian y Sofía Ismirlian por compartir su experiencia y asistencia en diseño de PCB e integración de periféricos al microcontrolador.

Al doctor Miguel Aguirre y al ingeniero Lien Tori por su asistencia con las pruebas realizadas en el karting eléctrico dentro de las instalaciones del Centro de Investigación y Desarrollo de Electrónica Industrial (CIDEI).

A Jorge Cáceres y Gabriel Zapata por la asistencia en el pañol de electrónica y la predisposición a ayudar ante las diversas necesidades a lo largo del desarrollo del proyecto.

Y finalmente, al ITBA por la formación profesional e interpersonal recibida.

### **3. Índice**

#### **Índice**

<b>2. Agradecimientos</b>	<b>1</b>
<b>3. Índice</b>	<b>2</b>
<b>4. Acrónimos y Definiciones</b>	<b>5</b>
<b>5. Resumen</b>	<b>6</b>
<b>6. Introducción</b>	<b>7</b>
6.1. Antecedentes . . . . .	7
6.2. Contexto del proyecto . . . . .	7
<b>7. Objetivos</b>	<b>8</b>
7.1. Finalidad del proyecto . . . . .	8
7.2. Planteamiento del problema a resolver . . . . .	8
7.3. Alcance . . . . .	8
<b>8. Definición de Producto</b>	<b>9</b>
8.1. Requerimientos de Cliente . . . . .	9
8.1.1. Relevamiento de Datos . . . . .	10
8.1.2. Casa de calidad . . . . .	11
8.2. Diagrama Funcional de Interfaces . . . . .	11
8.3. Especificaciones de diseño . . . . .	12
8.3.1. Especificaciones Funcionales . . . . .	12
8.3.2. Especificaciones de Interfaz . . . . .	12
8.3.3. Especificaciones de Performance . . . . .	14
8.3.4. Especificaciones de Implementación . . . . .	14
<b>9. Plan de Validación</b>	<b>15</b>
9.1. Diseño de Bancos de Pruebas . . . . .	15
9.1.1. Banco de pruebas N°1 . . . . .	15
9.1.2. Banco de pruebas N°2 . . . . .	18
9.1.3. Banco de pruebas N°3 . . . . .	21
9.1.4. Banco de pruebas N°4 . . . . .	21
9.1.5. Banco de pruebas N°5 . . . . .	21
9.2. Especificaciones de Tests . . . . .	22
9.2.1. Tests de interfaz . . . . .	22
9.2.2. Tests funcionales . . . . .	25
9.2.3. Tests de performance . . . . .	27
9.2.4. Tests de implementación . . . . .	28
9.3. Matriz de trazabilidad . . . . .	28
9.4. Plan de Tests . . . . .	28
<b>10. Análisis de factibilidad</b>	<b>30</b>

10.1. Factibilidad tecnológica . . . . .	30
10.1.1. Propuestas alternativas de diseño . . . . .	30
10.1.2. DFMEA . . . . .	34
10.2. Factibilidad de tiempos . . . . .	36
10.2.1. Planificación . . . . .	36
10.2.2. Programación . . . . .	38
10.3. Factibilidad económica . . . . .	38
10.3.1. Estudio de Mercado . . . . .	38
10.3.2. Modelo de negocios . . . . .	39
10.3.3. Costo de producción . . . . .	39
10.3.4. Flujo de fondos anual . . . . .	39
10.4. Factibilidad legal y de responsabilidad civil . . . . .	43
10.4.1. Leyes, Normas y Regulaciones . . . . .	43
<b>11. Ingeniería de detalle</b>	<b>45</b>
11.1. Hardware . . . . .	45
11.1.1. Diagrama de bloques . . . . .	45
11.1.2. Descripción de cada módulo . . . . .	45
11.1.3. Selección y cálculo de elementos circuitales de módulos . . . . .	53
11.1.4. Plan de pruebas de módulos . . . . .	57
11.2. Software . . . . .	60
11.2.1. Diagramas de bloques y estados . . . . .	60
11.2.2. Descripción de subrutinas . . . . .	61
11.2.3. Listados comentados del código . . . . .	64
11.2.4. Plan de prueba de módulos y de depuración del software . . . . .	70
<b>12. Construcción del prototipo</b>	<b>74</b>
12.1. Definición de los módulos . . . . .	74
12.2. Diseño de los circuitos impresos . . . . .	74
12.3. Diseño mecánico . . . . .	77
<b>13. Validación del prototipo</b>	<b>80</b>
13.1. Plan y protocolos especiales de medición . . . . .	80
13.2. Conexiones para pruebas . . . . .	80
13.3. Resultados . . . . .	86
13.3.1. Resultados de pruebas de módulos . . . . .	86
13.3.2. Resultados tests de validación . . . . .	88
13.3.3. Resultados de pruebas de integración prototipo - karting . . . . .	91
<b>14. Estudios de confiabilidad de hardware y de software</b>	<b>92</b>
14.1. Estudio de confiabilidad de hardware . . . . .	92
14.2. Estudio de confiabilidad de software . . . . .	95
<b>15. Conclusiones</b>	<b>97</b>
15.1. Objetivos Alcanzados . . . . .	97
15.2. Recomendaciones para futuros diseños . . . . .	97
<b>16. Referencias</b>	<b>98</b>

<b>17. Anexos</b>	<b>99</b>
17.1. Esquemáticos . . . . .	99
17.2. Planos de PCB . . . . .	99
17.3. Listado de Partes y Componentes (BOM) . . . . .	99
17.4. Códigos de Software . . . . .	106

## 4. Acrónimos y Definiciones

Acrónimo	Descripción
ADC	Convertidor analógico-digital ( <i>Analog to Digital Converter</i> )
CAN	Red de Area del Controlador ( <i>Controller Area Network</i> )
CIDEI	Centro de Investigación y Desarrollo de Electrónica Industrial
CSV	Valores Separados por Coma ( <i>Comma Separated Values</i> )
DUT	Dispositivo Bajo Prueba ( <i>Device Under Test</i> )
EMI	Interferencia Electromagnética ( <i>Electromagnetic Interference</i> )
FIFO	Entra primero, sale primero ( <i>First In - First Out</i> )
GND	Tierra ( <i>Ground</i> )
GPIO	Entrada-salida de propósito general ( <i>General Purpose Input-Output</i> )
LED	Diodo emisor de luz ( <i>Light Emitting Diode</i> )
MSD	Dispositivo de almacenamiento masivo ( <i>Mass Storage Device</i> )
PC	Computadora Personal ( <i>Personal Computer</i> )
RTC	Reloj de tiempo real ( <i>Real Time Clock</i> )
RTOS	Sistema operativo de tiempo real ( <i>Real Time Operating System</i> )
SD	<i>Secure Digital</i>
SDK	Kit de desarrollo de software ( <i>Software Development Kit</i> )
TBD	A Ser Determinado ( <i>To Be Determined</i> )
USB	Bus Universal en Serie ( <i>Universal Serial Bus</i> )
VDC	Volts de Corriente Continua ( <i>Volts Direct Current</i> )

**Conector DE-9:** D-subminiatura es un tipo común de conectores eléctricos. Reciben su nombre del característico protector de metal del conector con forma de D. En cuanto a la nomenclatura de este grupo de conectores, la “D” se utiliza como prefijo de toda la categoría, seguida de una letra (A, B, C, D, o E) que indica el tamaño de la carcasa, luego un número que indica la cantidad de terminales, y finalmente la letras P/M o S/F (*Pins/Macho* o *Sockets/Hembra*).

Para el lector familiarizado con este tipo de conectores, cabe aclarar que el conector DE-9 es comúnmente llamado “DB-9” de forma errónea, dado que la categoría de tamaño de carcasa B no cuenta con una opción estándar con 9 terminales. La siguiente tabla muestra los distintos conectores de la familia.

Normal density		High density		Double density	
Name	Pin layout	Name	Pin layout	Name	Pin layout
DA-15	8-7	DA-26	9-9-8	DA-31	10-11-10
DB-25	13-12	DB-44	15-15-14	DB-52	17-18-17
DC-37	19-18	DC-62	21-21-20	DC-79	26-27-26
DD-50	17-16-17	DD-78	20-19-20-19	DD-100	26-25-24-25
DE-09	5-4	DE-15	5-5-5	DE-19	6-7-6

## **5. Resumen**

En este proyecto final de carrera de Ingeniería Electrónica se plantean las problemáticas existentes en el mercado actual y la solución con el producto presentado: un módulo para el registro de mensajes transmitidos a través de un bus CAN para vehículos experimentales.

Los requisitos y especificaciones del producto se examinan teniendo en cuenta las diversas necesidades de los clientes, que abarcan a centros de investigación y desarrollo que llevan a cabo proyectos relacionados con vehículos experimentales.

Se desarrolla un análisis exhaustivo de la viabilidad tecnológica, económica, temporal y legal del proyecto. Este análisis implica el diseño de un modelo de negocios y la evaluación de la rentabilidad del sistema a desarrollar. Se toman en cuenta los plazos estimados para la realización de cada tarea, partiendo por la definición inicial del problema hasta las fases de validación y elaboración de documentación final.

Luego, en el proceso de análisis, se procede a desglosar los diversos módulos que integran el sistema, abarcando tanto los elementos físicos como los programas informáticos que lo constituyen. En relación al hardware, se examinan detalladamente los componentes esenciales destacando su función dentro del conjunto y su interacción con otros elementos. Asimismo, se explora la funcionalidad de los módulos de software, resaltando su importancia en el control y gestión eficiente del sistema. Cada módulo, ya sea de hardware o software, se somete a un riguroso proceso de pruebas de validación para garantizar su correcto funcionamiento y conformidad con los requisitos del proyecto, asegurando así la calidad y fiabilidad del sistema en su totalidad.

Una vez completada la manufactura de un prototipo, se realizan pruebas exhaustivas para garantizar su conformidad con las necesidades del cliente. Los resultados obtenidos se analizan de forma extensa para identificar áreas de mejora. Este proceso de evaluación no solo valida el diseño actual, sino que también informa sobre posibles ajustes futuros para optimizar aún más el producto y mantener su relevancia en un entorno competitivo.

## **6. Introducción**

### **6.1. Antecedentes**

En el ámbito del trabajo de investigación y desarrollo de vehículos eléctricos, es de suma relevancia el entendimiento detallado de su desempeño en una variedad de condiciones, así como la capacidad de identificar y analizar los eventos que preceden a la ocurrencia de fallas, con el propósito fundamental de impulsar mejoras futuras.

A pesar de la existencia de productos comerciales que ofrecen servicios de registro de datos, es esencial reconocer que muchos de estos productos pueden proporcionar características que, si bien son avanzadas, pueden ser superfluas para los requerimientos específicos del proyecto en cuestión resultando en un gasto adicional innecesario.

### **6.2. Contexto del proyecto**

El desarrollo de este proyecto se ve condicionado en parte por la escasez global de componentes electrónicos que atraviesa a todos los ambientes de innovación tecnológica desde hace algunos años. Además, Argentina, el país donde se encuentra el grupo de trabajo, también sufre algunas limitaciones en cuanto a disponibilidad de recursos para el diseño electrónico.

El proyecto se realiza en conjunto con el CIDEI (*Centro de investigación y desarrollo de electrónica industrial*), el cual cumple el rol simultáneo de principal cliente. Este centro, perteneciente al ITBA al igual que los autores, proveerá ciertos materiales necesarios para el desarrollo, así como la tutoría de dos de sus miembros: el Ing. Nicolás Magliola y el Esp. Dr. Ing. Pablo Martín Cossutta.

Este proyecto se desarrolla con un prototipo del karting para la verificación de funcionalidad del producto presentado.

## **7. Objetivos**

### **7.1. Finalidad del proyecto**

La finalidad de este proyecto es el de proveer al CIDEI un *data logger* de mensajes transmitidos a través de un bus CAN para ser integrado al karting eléctrico que poseen. Este producto permitirá al usuario extraer información del vehículo para su posterior análisis de desempeño y diagnóstico de problemas durante el proceso de depuración de sus desarrollos.

### **7.2. Planteamiento del problema a resolver**

El desafío de diseño del producto se encuentra principalmente en la velocidad y la confiabilidad de la adquisición de datos y la “integración” con el vehículo, ya que las condiciones de uso de este data logger incluye el registro de los casos de falla del vehículo. Es por esto que este trabajo tendrá como foco los siguientes aspectos fundamentales: diseño para funcionamiento en condiciones de falla del vehículo, integración y validación del producto. En particular, las pruebas de validación serán de extrema relevancia para asegurar un producto de la calidad que se requiere.

### **7.3. Alcance**

Este proyecto contempla el diseño de un CAN data logger para ser utilizado en el karting, con los análisis relevantes para asegurar la viabilidad tecnológica, temporal, financiera y legal del producto. La verificación de la calidad del diseño estará basada en un único prototipo no comercial, que no deberá cumplir con todos los requerimientos de producto final.

El dispositivo no estará preparado para ser utilizado en vehículos comerciales. El alcance de este producto está limitado a la adquisición y almacenamiento del bus CAN del vehículo, no está contemplado la interpretación de estos datos.

## 8. Definición de Producto

### 8.1. Requerimientos de Cliente

ID	Requerimiento
REQ-LOG-1	Deberá poder recibir y almacenar correctamente mensajes CAN del estándar ISO 11898-2 [1] de velocidad 125 kbit/s de tipo DATA.
REQ-LOG-2	Deberá soportar el protocolo CAN en el modo no-extendido (CAN 2.0A).
REQ-LOG-3	Deberá poder registrar correctamente mensajes CAN con una separación temporal entre mensajes igual o mayor a 2 ms.
REQ-LOG-4	El almacenamiento deberá ocurrir en una tarjeta de tipo SD (deberá ser de tipo SDXC o SDUC), con el fin de ser extraíble en estado apagado.
REQ-LOG-5	Los datos deberán guardarse en archivos de formato CSV.
REQ-LOG-6	Para cada mensaje, se debe guardar: <ol style="list-style-type: none"><li>1. El ID del mensaje representando su valor hexadecimal mediante caracteres ASCII.</li><li>2. La cantidad de bytes del mensaje (<i>data length</i>) representando su valor hexadecimal mediante caracteres ASCII.</li><li>3. Cada byte de dato del mensaje (B0 a B7) representando su valor hexadecimal mediante caracteres ASCII.</li><li>4. Fecha y hora en formato YYYY,MM,DD, hh, mm, ss, ms, ms, ms.</li></ol>
REQ-LOG-7	Los archivos CSV almacenados en la tarjeta SD se deberán poder acceder desde una computadora mediante una conexión cableada.

Tabla 8.1: Requerimientos de registro y almacenamiento.

ID	Requerimiento
REQ-POW-1	Se deberá poder alimentar con una tensión entre 10 y 30 volts, con un ripple de 1 volt.
REQ-POW-2	Deberá tener protección contra sobre-tensión de hasta 50 volts, y tensión inversa de hasta 30 volts.
REQ-POW-3	Deberá consumir una media de menos de 1W mientras recibe alimentación externa.
REQ-POW-4	Deberá consumir un pico máximo de 3W mientras recibe alimentación externa.
REQ-POW-5	Deberá mantener su fecha y hora interna por 1 año sin alimentación externa (a temperatura 25 °C).
REQ-POW-6	Deberá poder guardar correctamente en el medio de almacenamiento todos los datos pendientes por guardarse, en caso de pérdida de alimentación.

Tabla 8.2: Requerimientos de alimentación y autonomía.

ID	Requerimiento																				
REQ-CON-1	Deberá tener un medio sencillo de conexión cableada con la computadora para el acceso a los datos de la tarjeta SD.																				
REQ-CON-2	Deberá tener dos conectores DE-9 en paralelo para la conexión al bus CAN. La distribución de pines deberá ser la siguiente (donde NC representa la ausencia de una conexión): <table style="margin-left: 20px; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Pin DE-9</th> <th style="text-align: center;">Conexión</th> </tr> </thead> <tbody> <tr><td style="text-align: center;">1</td><td style="text-align: center;">NC</td></tr> <tr><td style="text-align: center;">2</td><td style="text-align: center;">CAN_LOW</td></tr> <tr><td style="text-align: center;">3</td><td style="text-align: center;">GROUND</td></tr> <tr><td style="text-align: center;">4</td><td style="text-align: center;">NC</td></tr> <tr><td style="text-align: center;">5</td><td style="text-align: center;">NC</td></tr> <tr><td style="text-align: center;">6</td><td style="text-align: center;">NC</td></tr> <tr><td style="text-align: center;">7</td><td style="text-align: center;">CAN_HIGH</td></tr> <tr><td style="text-align: center;">8</td><td style="text-align: center;">NC</td></tr> <tr><td style="text-align: center;">9</td><td style="text-align: center;">POWER</td></tr> </tbody> </table>	Pin DE-9	Conexión	1	NC	2	CAN_LOW	3	GROUND	4	NC	5	NC	6	NC	7	CAN_HIGH	8	NC	9	POWER
Pin DE-9	Conexión																				
1	NC																				
2	CAN_LOW																				
3	GROUND																				
4	NC																				
5	NC																				
6	NC																				
7	CAN_HIGH																				
8	NC																				
9	POWER																				

Tabla 8.3: Requerimientos de conexionado.

ID	Requerimiento
REQ-MEC-1	El equipo deberá tener un muy bajo peso y tamaño en relación al resto del vehículo.
REQ-MEC-2	El equipo deberá tener resistencia al polvo.
REQ-MEC-3	El equipo deberá ser de fácil sujeción al chasis del vehículo.
REQ-MEC-4	El equipo deberá soportar las vibraciones durante la operación del vehículo.

Tabla 8.4: Requerimientos mecánicos.

ID	Requerimiento
REQ-MISC-1	No deberá estar sujeto a normativas aplicables a equipos para vehículos comerciales.
REQ-MISC-2	No deberá incluir el software necesario para analizar los datos obtenidos.
REQ-MISC-3	Deberá tener un costo unitario menor a 50 dólares estadounidenses en materiales.
REQ-MISC-4	Deberá operar normalmente cuando la temperatura ambiente se encuentre entre $-20^{\circ}\text{C}$ y $60^{\circ}\text{C}$ .

Tabla 8.5: Otros requerimientos.

### 8.1.1. Relevamiento de Datos

La información referente a los requerimientos del cliente fue obtenida a partir de reuniones entre el equipo de desarrollo del proyecto, el Ing. Nicolás Magliola y el Esp. Dr. Ing. Pablo Martín Cossutta, los cuales pertenecen al CIDEI. Estas reuniones se llevaron a cabo los días 18/03, 08/04, 13/04 y 20/04 del año 2022.

De acuerdo a las recomendaciones recibidas por el grupo de investigación del ITBA, es que se definen los requerimientos del producto, dado su amplio conocimiento respecto a las necesidades propias y del mercado referentes a vehículos experimentales.

### 8.1.2. Casa de calidad

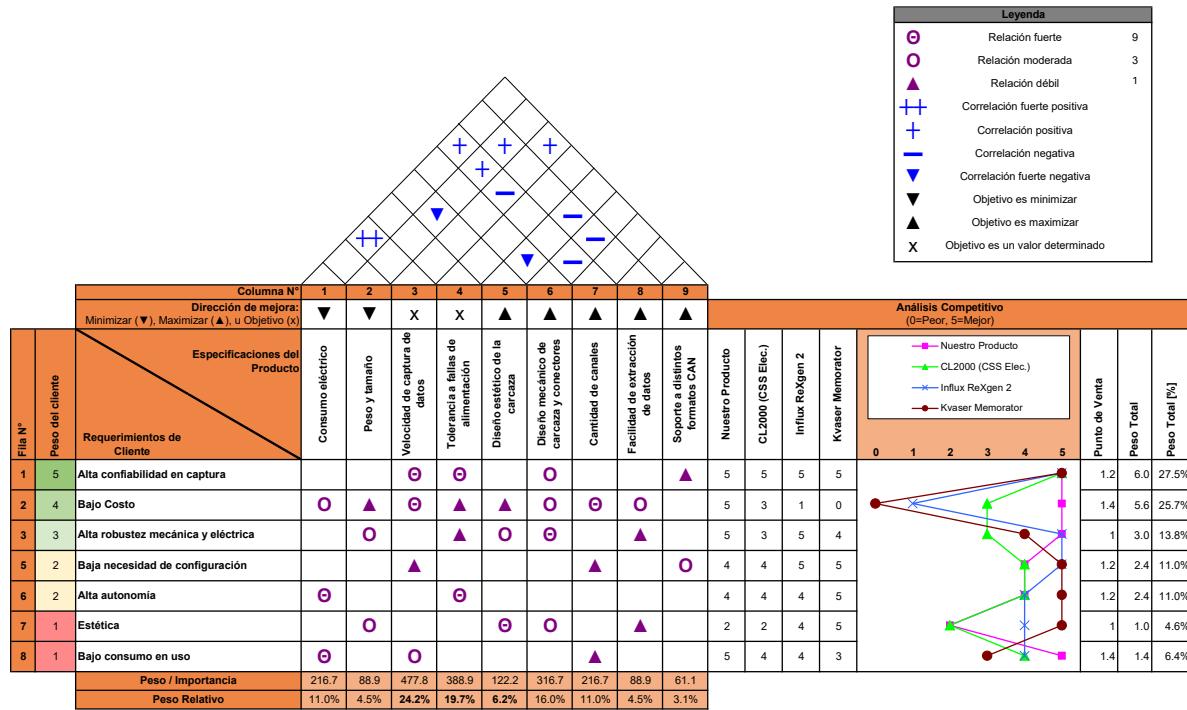


Figura 8.1: Casa de Calidad del producto.

### 8.2. Diagrama Funcional de Interfaces

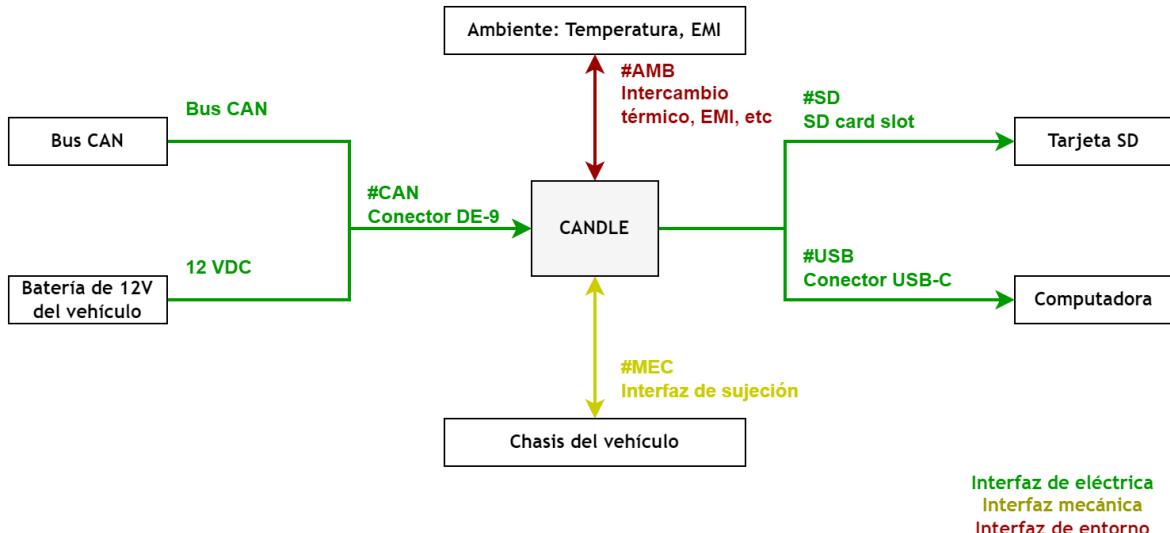


Figura 8.2: Diagrama funcional de interfaces.

### 8.3. Especificaciones de diseño

#### 8.3.1. Especificaciones Funcionales

ID	Especificación	Origen
FUN-LOG-01	Deberá poder recibir y almacenar correctamente mensajes CAN del estándar ISO 11898-2 a velocidad 125 kbit/s de tipo DATA.	REQ-LOG-1
FUN-LOG-02	Deberá soportar <i>data frames</i> con cualquier cantidad de bytes dentro de las especificadas en el estándar ISO 11898-2 (entre 0 y 8 bytes).	REQ-LOG-1
FUN-LOG-03	Deberá soportar el protocolo CAN en el modo no-extendido (CAN 2.0A).	REQ-LOG-2
FUN-LOG-04	Deberá poder registrar correctamente mensajes CAN con una separación temporal entre mensajes igual o mayor a 2 ms.	REQ-LOG-3
FUN-LOG-05	Los datos deberán guardarse en archivos de formato CSV.	REQ-LOG-5
FUN-LOG-06	Para cada mensaje, se debe guardar: <ol style="list-style-type: none"> <li>El ID del mensaje representando su valor hexadecimal mediante caracteres ASCII.</li> <li>La cantidad de bytes del mensaje (<i>data length</i>) representando su valor hexadecimal mediante caracteres ASCII.</li> <li>Cada byte de dato del mensaje (B0 a B7) representando su valor hexadecimal mediante caracteres ASCII.</li> <li>Fecha y hora en formato YYYY,MM,DD,hh,mm,ss,msmsms.</li> </ol>	REQ-LOG-6
FUN-LOG-07	Los archivos CSV almacenados en la tarjeta SD se deberán poder acceder desde una computadora mediante una conexión cableada.	REQ-LOG-7
FUN-POW-01	Deberá poder guardar correctamente en el medio de almacenamiento todos los datos pendientes por guardarse en caso de pérdida de alimentación.	REQ-POW-6
FUN-MISC-01	No deberá estar sujeto a normativas aplicables a equipos para vehículos comerciales.	REQ-MISC-1
FUN-MISC-02	No deberá incluir el software necesario para analizar los datos obtenidos.	REQ-MISC-2

Tabla 8.6: Especificaciones funcionales

#### 8.3.2. Especificaciones de Interfaz

ID	Especificación	Origen
INT-VIN-01	Se deberá poder alimentar con una tensión entre 10 y 30 volts, con un ripple de 1 volt.	REQ-POW-1
INT-VIN-02	Deberá tener protección contra sobre-tensión de hasta 50 volts, y tensión inversa de hasta 30 volts.	REQ-POW-2

Tabla 8.7: Especificaciones de interfaz de fuente de alimentación.

ID	Especificación	Origen
INT-PC-01	Deberá almacenar los datos en una tarjeta SD extraíble de tipo SDXC o SDUC.	REQ-LOG-4
INT-PC-02	Deberá tener un conector USB Tipo-C a PC para la extracción de datos.	REQ-CON-1
INT-PC-03	La conexión con la PC deberá implementar el estándar USB 2.0.	REQ-CON-1
INT-PC-04	La clase de dispositivo USB que deberá implementar el producto es Dispositivo de Almacenamiento Masivo (MSD por sus siglas en inglés, <i>Mass Storage Device</i> )	REQ-LOG-7

Tabla 8.8: Especificaciones de interfaz de computadora

ID	Especificación	Origen																				
INT-CAN-01	<p>Deberá tener dos conectores DE-9 en paralelo para la conexión al bus CAN. La distribución de pines deberá ser la siguiente (donde NC representa la ausencia de una conexión):</p> <table style="margin-left: 20px;"> <thead> <tr> <th>Pin DE-9</th> <th>Conexión</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>NC</td> </tr> <tr> <td>2</td> <td>CAN_LOW</td> </tr> <tr> <td>3</td> <td>GROUND</td> </tr> <tr> <td>4</td> <td>NC</td> </tr> <tr> <td>5</td> <td>NC</td> </tr> <tr> <td>6</td> <td>NC</td> </tr> <tr> <td>7</td> <td>CAN_HIGH</td> </tr> <tr> <td>8</td> <td>NC</td> </tr> <tr> <td>9</td> <td>POWER</td> </tr> </tbody> </table>	Pin DE-9	Conexión	1	NC	2	CAN_LOW	3	GROUND	4	NC	5	NC	6	NC	7	CAN_HIGH	8	NC	9	POWER	REQ-CON-2
Pin DE-9	Conexión																					
1	NC																					
2	CAN_LOW																					
3	GROUND																					
4	NC																					
5	NC																					
6	NC																					
7	CAN_HIGH																					
8	NC																					
9	POWER																					

Tabla 8.9: Especificaciones de interfaz de bus CAN

ID	Especificación	Origen
INT-MEC-01	Deberá cumplir un grado de protección IP5X.	REQ-MEC-2
INT-MEC-02	El equipo deberá poder sujetarse con cuatro tornillos con un diámetro de 1/8 de pulgada, distanciados entre sí no más de 15 cm y no menos de 5 cm.	REQ-MEC-3
INT-MEC-03	El equipo deberá tolerar vibraciones aleatorias mecánicas de hasta $20m/s^2$ de aceleración RMS y de hasta $0,4g^2/Hz$ de densidad espectral de aceleración entre 50-1500 Hz.	REQ-MEC-4

Tabla 8.10: Especificaciones de Interfaz Mecánica

### 8.3.3. Especificaciones de Performance

ID	Especificación	Origen
PER-POW-01	Deberá consumir una media de menos de 1W mientras recibe alimentación externa.	REQ-POW-3
PER-POW-02	Deberá consumir un máximo de 3W pico mientras recibe alimentación externa.	REQ-POW-4
PER-POW-03	Deberá mantener su fecha y hora interna por al menos 1 año sin alimentación externa (a temperatura 25 °C).	REQ-POW-5

Tabla 8.11: Especificaciones de Performance

### 8.3.4. Especificaciones de Implementación

ID	Especificación	Origen
IMP-DIM-01	Deberá tener un peso menor a 500g.	REQ-MEC-1
IMP-DIM-02	El dispositivo no deberá exceder las siguientes dimensiones: Largo $\leq$ 20 cm Ancho $\leq$ 20 cm Alto $\leq$ 10 cm	REQ-MEC-1

Tabla 8.12: Especificaciones Dimensionales y de Peso

ID	Especificación	Origen
IMP-OPE-01	El dispositivo deberá poder operar normalmente cuando la temperatura ambiente sea $-20^{\circ}\text{C} < T_{AMB} < 60^{\circ}\text{C}$	REQ-MISC-4

Tabla 8.13: Especificaciones de operación

ID	Especificación	Origen
IMP-COS-01	El costo de las partes que conforman el producto no deberá ser superior a 50 dólares estadounidenses.	REQ-MISC-3

Tabla 8.14: Especificaciones de costos

## 9. Plan de Validación

### 9.1. Diseño de Bancos de Pruebas

#### 9.1.1. Banco de pruebas N°1

##### 9.1.1.1 Diagrama

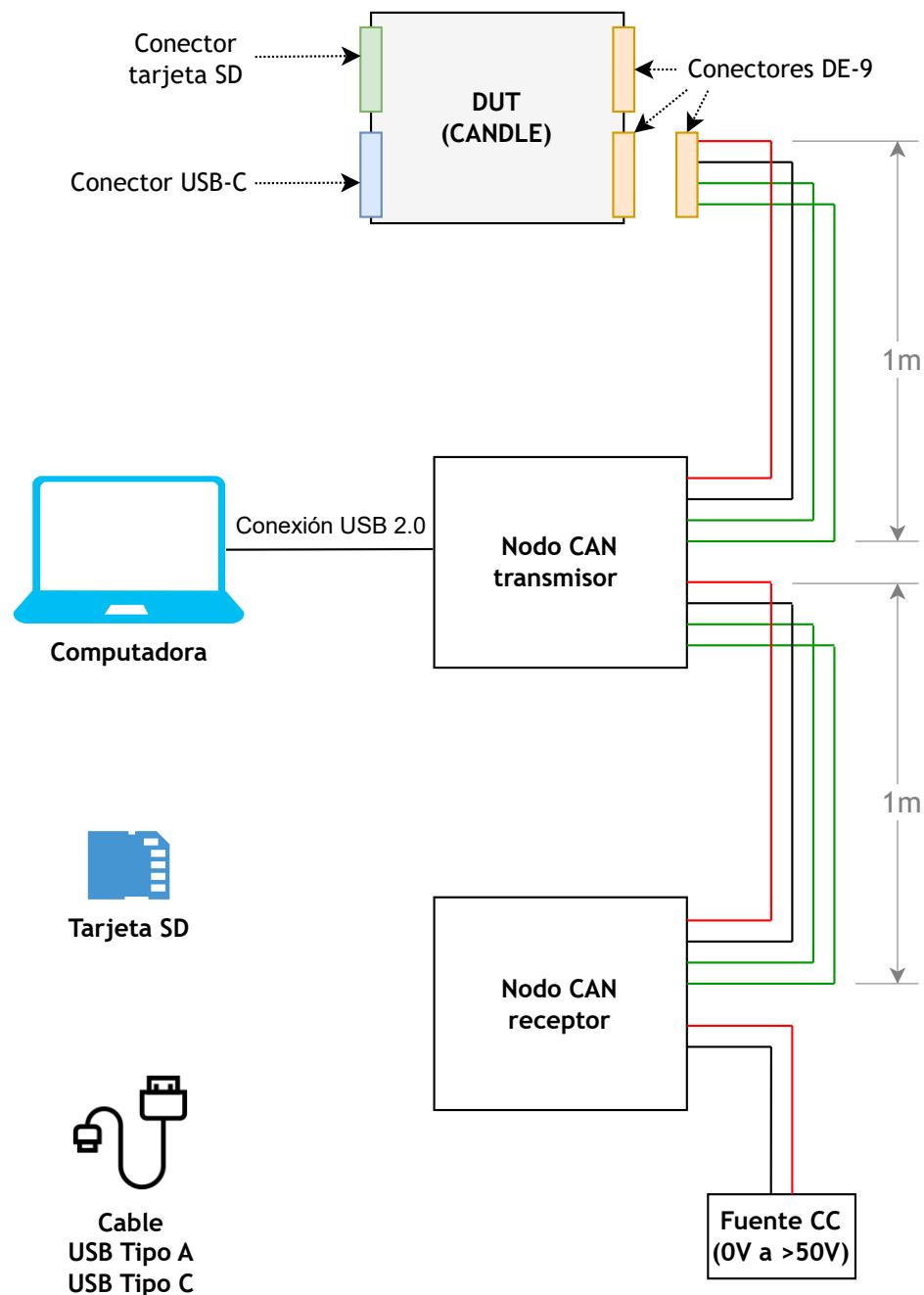


Figura 9.1: Banco de prueba N°1

### 9.1.1.2 Componentes

#### Nodo CAN transmisor

Dispositivo formado por un módulo controlador de CAN programable y un transceiver CAN. Cuenta con dos conectores macho polarizados para GND, CANL, CANH, y VCC conectados entre sí. Transmite mensajes de acuerdo al protocolo CAN 2.0 del estándar ISO 11898-2 a 125kHz. Su transmisión puede ser programada por la computadora, pudiendo controlar los siguientes parámetros:

- Cantidad de mensajes. Incluye la posibilidad de mandar una cantidad ilimitada.
- Tiempo entre mensajes.
- IDs de los mensajes. Puede ser diferente para cada mensaje.
- Cantidad de bytes de datos de los mensajes. Puede ser diferente para cada mensaje.
- Contenido de los bytes de datos de los mensajes. Puede ser diferente para cada mensaje.

#### Nodo CAN receptor

Dispositivo formado por un módulo controlador de CAN programable y un transceiver CAN. Cuenta con dos conectores macho polarizados para GND, CANL, CANH, y VCC conectados entre sí. Su única función es asertar el bit de *Acknowledge* en la trama de los mensajes CAN enviados por el nodo CAN transmisor.

#### Computadora

Computadora con al menos dos puertos USB Tipo A con protocolo USB 2.0 y con lector de tarjetas SD. Sus funciones son programar el nodo CAN transmisor para controlar los mensajes CAN que envía, y leer los logs almacenados en la tarjeta SD.

#### Fuente CC

Fuente de tensión continua regulable con salida de entre al menos 0V y 50V

#### Tarjeta SD

Tarjeta SD de tipo SDXC o SDUC utilizada para almacenar los logs capturados por el DUT.

#### Cable USB Tipo A — USB Tipo C

Cable con conectores USB Tipo A y USB Tipo C y cableado interno adecuado para soportar comunicación de tipo USB 2.0.

#### Cables de bus CAN

Dos tramos de 4 cables trenzados de  $1 \pm 0,2$  metros.

Color del cable	Señal
Negro	GND
Verde	CANL
Azul	CANH
Rojo	VCC

- El tramo A cuenta con un conector polarizado de 4 pines compatible con el pinout de los nodos CAN transmisor y receptor en un extremo, y un conector DE-9 en el otro extremo con el siguiente pinout:

Pin DE-9	Conexión
1	NC
2	CAN_LOW
3	GROUND
4	NC
5	NC
6	NC
7	CAN_HIGH
8	NC
9	POWER

- El tramo B cuenta con un conector polarizado de 4 pines compatible con el pinout de los nodos CAN transmisor y receptor en ambos extremos.

#### 9.1.1.3 Conexiones

- La computadora y el nodo CAN transmisor están conectados por un cable USB:
  - La conexión con la computadora es USB Tipo A.
  - La conexión con el nodo es MicroUSB.
- La fuente se conecta a los pines de uno de los conectores polarizados del nodo CAN receptor:
  - Se utiliza un cable negro para conectar GND de la fuente con el pin de GND del conector de nodo CAN receptor,
  - Se utiliza un cable rojo para conectar la salida de tensión positiva de la fuente con el pin de VCC del conector de nodo CAN receptor,
- El tramo B del bus CAN se conecta al nodo CAN transmisor y al nodo CAN receptor.
- El tramo A del bus CAN se conecta al nodo CAN transmisor.

#### 9.1.1.4 Setup

Antes de proceder con el test, se deben seguir los siguientes pasos:

1. Con la fuente CC desconectada y encendida, configurarla para entregar 12V.
2. Apagar la fuente CC.
3. Conectar la fuente CC al nodo CAN receptor.

### 9.1.2. Banco de pruebas N°2

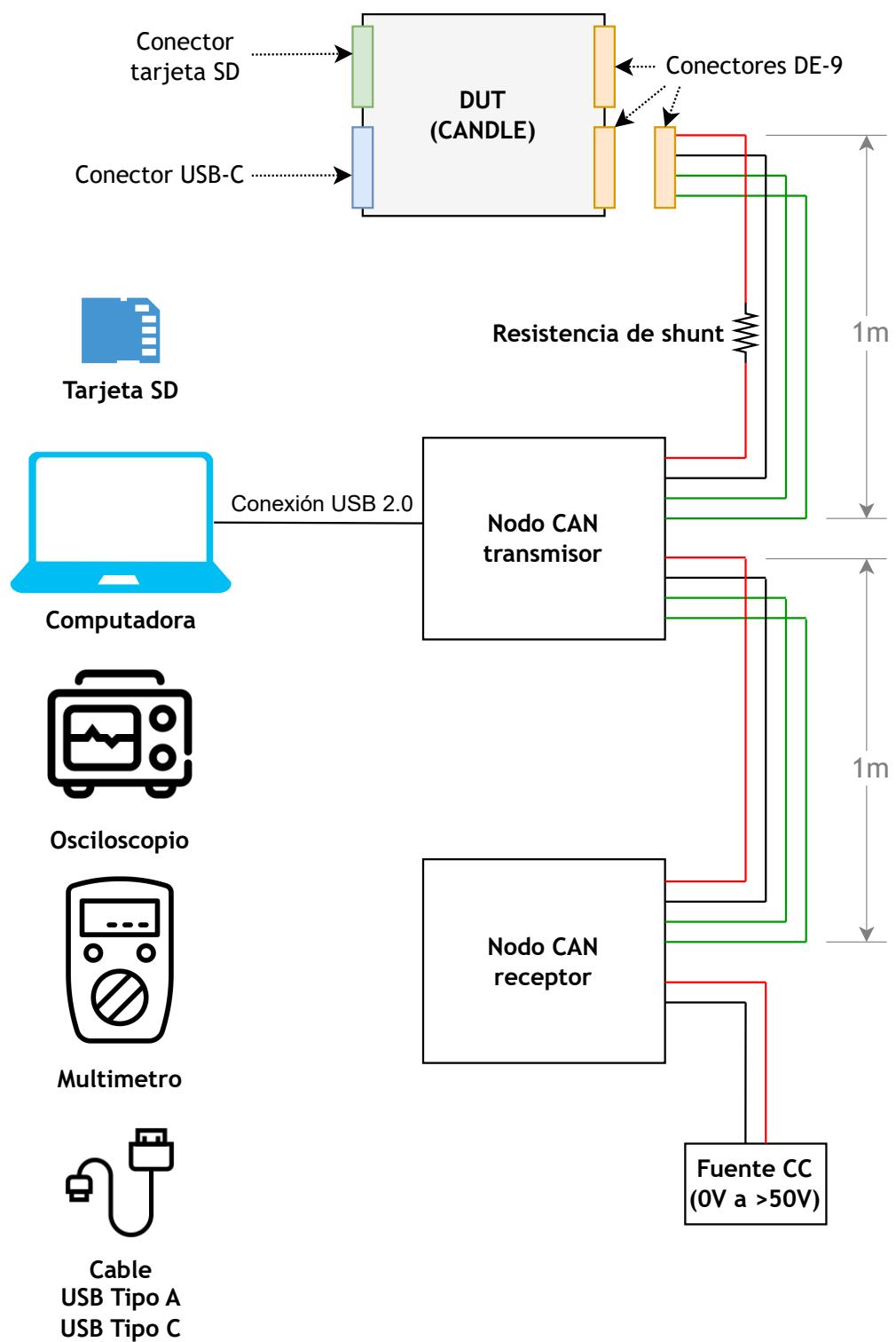


Figura 9.2: Banco de prueba N°2

#### 9.1.2.1 Componentes

##### Nodo CAN transmisor

Dispositivo formado por un módulo controlador de CAN programable y un transceiver CAN. Cuenta

con dos conectores macho polarizados para GND, CANL, CANH, y VCC conectados entre sí. Transmite mensajes de acuerdo al protocolo CAN 2.0 del estándar ISO 11898-2 a 125KHz. Su transmisión puede ser programada por la computadora, pudiendo controlar los siguientes parámetros:

- Cantidad de mensajes. Incluye la posibilidad de mandar una cantidad ilimitada.
- Tiempo entre mensajes.
- IDs de los mensajes. Puede ser diferente para cada mensaje.
- Cantidad de bytes de datos de los mensajes. Puede ser diferente para cada mensaje.
- Contenido de los bytes de datos de los mensajes. Puede ser diferente para cada mensaje.

#### Nodo CAN receptor

Dispositivo formado por un módulo controlador de CAN programable y un transceiver CAN. Cuenta con dos conectores macho polarizados para GND, CANL, CANH, y VCC conectados entre sí. Su única función es asertar el bit de *Acknowledge* en la trama de los mensajes CAN enviados por el nodo CAN transmisor.

#### Computadora

Computadora con al menos dos puertos USB Tipo A con protocolo USB 2.0 y con lector de tarjetas SD. Sus funciones son programar el nodo CAN transmisor para controlar los mensajes CAN que envía, y leer los logs almacenados en la tarjeta SD.

#### Fuente CC

Fuente de tensión continua regulable con salida de entre al menos 0V y 50V

#### Tarjeta SD

Tarjeta SD de tipo SDXC o SDUC utilizada para almacenar los logs capturados por el DUT.

#### Cable USB Tipo A — USB Tipo C

Cable con conectores USB Tipo A y USB Tipo C y cableado interno adecuado para soportar comunicación de tipo USB 2.0.

#### Cables de bus CAN

Dos tramos de 4 cables trenzados de  $1 \pm 0,2$  metros.

Color del cable	Señal
Negro	GND
Verde	CANL
Azul	CANH
Rojo	VCC

- El tramo A cuenta con un conector polarizado de 4 pines compatible con el pinout de los nodos CAN transmisor y receptor en un extremo, y un conector DE-9 en el otro extremo con el siguiente pinout:

Pin DE-9	Conexión
1	NC
2	CAN_LOW
3	GROUND
4	NC
5	NC
6	NC
7	CAN_HIGH
8	NC
9	POWER

Además, cuenta con una resistencia de shunt de menos de  $3\Omega$  y al menos 5W con las terminales expuestas en el cable rojo.

- El tramo B cuenta con un conector polarizado de 4 pines compatible con el pinout de los nodos CAN transmisor y receptor en ambos extremos.

**Osciloscopio** Osciloscopio de al menos dos canales con posibilidad de mostrar la resta entre dos señales.

**Multímetro** Multímetro con funcionalidad para medir resistencia con un error de  $\pm 1\Omega$  y para medir continuidad.

### 9.1.2.2 Conexiones

- La computadora y el nodo CAN transmisor están conectados por un cable USB:
  - La conexión con la computadora es USB Tipo A.
  - La conexión con el nodo es microUSB.
- La fuente se conecta a los pines de uno de los conectores polarizados del nodo CAN receptor:
  - Se utiliza un cable negro para conectar GND de la fuente con el pin de GND del conector de nodo CAN receptor,
  - Se utiliza un cable rojo para conectar la salida de tensión positiva de la fuente con el pin de VCC del conector de nodo CAN receptor,
- El tramo B del bus CAN se conecta al nodo CAN transmisor y al nodo CAN receptor.
- El tramo A del bus CAN se conecta al nodo CAN transmisor.

### 9.1.2.3 Setup

Antes de proceder con el test, se deben seguir los siguientes pasos:

1. Con la fuente CC desconectada y encendida, configurarla para entregar 12V.
2. Apagar la fuente CC.
3. Conectar la fuente CC al nodo CAN receptor.

### **9.1.3. Banco de pruebas N°3**

#### **9.1.3.1 Componentes**

- Balanza a batería con rango de medición de al menos 0.1kg a 3kg, y con resolución de al menos  $\pm 0,05\text{kg}$ .

#### **9.1.3.2 Conexiones**

- N/A

#### **9.1.3.3 Setup**

- Colocar la balanza sobre la superficie de trabajo, sin carga.
- Configurar el nivel de 0 de la balanza.

### **9.1.4. Banco de pruebas N°4**

Describo en sección 4.1.2.4 de [2] (norma que establece procedimientos para probar la tolerancia a condiciones ambientales de vehículos comerciales de equipos eléctricos y electrónicos).

### **9.1.5. Banco de pruebas N°5**

Describo en sección 12.1 de [16] (norma que establece procedimientos para probar la tolerancia a condiciones ambientales de vehículos comerciales de equipos eléctricos y electrónicos).

## 9.2. Especificaciones de Tests

### 9.2.1. Tests de interfaz

ID	Banco de pruebas	Procedimiento	Criterios de aceptación
TST-MSD-1	1	<ol style="list-style-type: none"> <li>1. Insertar la tarjeta SD en el lector de la computadora.</li> <li>2. En la tarjeta SD, borrar todos los archivos y crear un nuevo archivo vacío de nombre “TST-OK.txt”.</li> <li>3. En la computadora, listar los dispositivos de almacenamiento masivo.</li> <li>4. Insertar la tarjeta SD en el DUT.</li> <li>5. Conectar la computadora al DUT mediante el cable USB Tipo-A — USB Tipo-C, utilizando un puerto USB 2.0.</li> <li>6. En la computadora, volver listar los dispositivos de almacenamiento masivo.</li> </ol>	La segunda vez que se listan los dispositivos de almacenamiento masivo aparece uno que contiene un único archivo llamado “TST-OK.txt”

Tabla 9.1: Tests de validación de interfaz grupo MSD.

ID	Banco de pruebas	Procedimiento	Criterios de aceptación
TST-SD-1	1	<ol style="list-style-type: none"> <li>1. Insertar la tarjeta SD en el lector de la computadora y vaciarla.</li> <li>2. Insertar la tarjeta SD en el DUT.</li> <li>3. Programar el nodo CAN transmisor para el envío de 1 mensaje CAN de tipo dato de largo 1, contenido 0xFF y ID 0x44.</li> <li>4. Desconectar la fuente CC del nodo CAN receptor y encenderla.</li> <li>5. Configurar la fuente CC para entregar 12V.</li> <li>6. Apagar la fuente CC.</li> <li>7. Conectar la fuente al nodo CAN receptor.</li> <li>8. Conectar el bus CAN al DUT.</li> <li>9. Encender la fuente CC.</li> <li>10. Comenzar el envío del mensaje.</li> <li>11. Esperar 5 segundos.</li> <li>12. Apagar la fuente CC.</li> <li>13. Desconectar el bus CAN del DUT.</li> <li>14. Insertar la tarjeta SD en el lector de la computadora</li> </ol>	En la tarjeta SD existe un log.

Tabla 9.2: Tests de validación de interfaz grupo SD.

ID	Banco de pruebas	Procedimiento	Criterios de aceptación
TST-ALIM-1	1	<ol style="list-style-type: none"> <li>1. Insertar la tarjeta SD en el lector de la computadora y vaciarla.</li> <li>2. Insertar la tarjeta SD en el DUT.</li> <li>3. Programar el nodo CAN transmisor para el envío de 1 mensaje CAN de tipo dato de largo 1, contenido 0xFF y ID 0x44.</li> <li>4. Para las tensiones VCC = [10V, 20V, 30V]:             <ol style="list-style-type: none"> <li>a) Desconectar la fuente CC, encenderla, configurar la fuente CC para entregar VCC, y apagarla.</li> <li>b) Conectar la fuente al nodo CAN receptor.</li> <li>c) Conectar el bus CAN al DUT.</li> <li>d) Encender la fuente CC.</li> <li>e) Comenzar el envío del mensaje.</li> <li>f) Esperar 5 segundos.</li> <li>g) Apagar la fuente CC.</li> <li>h) Desconectar el bus CAN del DUT.</li> </ol> </li> <li>5. Conectar la computadora al DUT mediante el cable USB Tipo A — USB Tipo C</li> <li>6. Abrir la carpeta del dispositivo de almacenamiento masivo</li> </ol>	<p>En el nuevo dispositivo de almacenamiento masivo existen tres logs, y cada uno contiene el mensaje enviado y las horas de sus envíos.</p>
TST-ALIM-2	1	<ol style="list-style-type: none"> <li>1. Insertar la tarjeta SD en el lector de la computadora y vaciarla.</li> <li>2. Insertar la tarjeta SD en el DUT.</li> <li>3. Programar el nodo CAN transmisor para el envío de 1 mensaje CAN de tipo dato de largo 1, contenido 0xFF y ID 0x44.</li> <li>4. Desconectar la fuente CC, encenderla, configurar la fuente CC para entregar 0V</li> <li>5. Conectar la fuente al nodo CAN receptor.</li> <li>6. Conectar el bus CAN al DUT.</li> <li>7. Aumentar lentamente la tensión de la fuente CC hasta llegar a 50V.</li> <li>8. Bajar lentamente la tensión de la fuente CC hasta llegar a 12V.</li> <li>9. Comenzar el envío de mensajes CAN por parte del nodo CAN transmisor.</li> <li>10. Apagar la fuente CC.</li> <li>11. Desconectar el bus CAN del DUT.</li> <li>12. Conectar la computadora al DUT mediante el cable USB Tipo A — USB Tipo C</li> <li>13. Abrir la carpeta del dispositivo de almacenamiento masivo</li> </ol>	<p>En el nuevo dispositivo de almacenamiento masivo existe un log que contiene el mensaje enviado y la hora de su envío.</p>

Tabla 9.3: Tests de validación de interfaz grupo ALIMENTACIÓN (continuación).

ID	Banco de pruebas	Procedimiento	Criterios de aceptación
TST-ALIM-3	1	<ol style="list-style-type: none"> <li>1. Desconectar la fuente CC, encenderla, configurarla para entregar 30V y apagarla.</li> <li>2. Conectar el pin GND del conector DE-9 del DUT a la salida VCC de la fuente CC, y el pin VCC del conector DE-9 del DUT al GND de la fuente CC.</li> <li>3. Encender la fuente durante 10 segundos y luego apagarla.</li> <li>4. Insertar la tarjeta SD en el lector de la computadora y vaciarla.</li> <li>5. Insertar la tarjeta SD en el DUT.</li> <li>6. Programar el nodo CAN transmisor para el envío de 1 mensaje CAN de tipo dato de largo 1, contenido 0xFF y ID 0x44.</li> <li>7. Desconectar la fuente CC y encenderla, configurarla para entregar 12V y apagarla.</li> <li>8. Conectar la fuente al nodo CAN receptor en sentido correcto (GND con GND y VCC con VCC).</li> <li>9. Conectar el bus CAN al DUT.</li> <li>10. Encender la fuente CC.</li> <li>11. Comenzar el envío de mensajes CAN por parte del nodo CAN transmisor.</li> <li>12. Apagar la fuente CC.</li> <li>13. Desconectar el bus CAN del DUT.</li> <li>14. Conectar la computadora al DUT mediante el cable USB Tipo A — USB Tipo C</li> <li>15. Abrir la carpeta del dispositivo de almacenamiento masivo</li> </ol>	<p>Al conectar la fuente en inversa se debe encender el LED indicador, y en el dispositivo de almacenamiento masivo existe un log que contiene el mensaje enviado y la hora de su envío.</p>

Tabla 9.4: Tests de validación de interfaz grupo ALIMENTACIÓN (continuación).

ID	Banco de pruebas	Procedimiento	Criterios de aceptación
TST-DE9-1	2	<ol style="list-style-type: none"> <li>1. Medir la continuidad entre los pines asignados a GND de los dos conectores DE-9 del DUT.</li> <li>2. Medir la continuidad entre los pines asignados a CANL de los dos conectores DE-9 del DUT.</li> <li>3. Medir la continuidad entre los pines asignados a CANH de los dos conectores DE-9 del DUT.</li> <li>4. Medir la continuidad entre los pines asignados a VCC de los dos conectores DE-9 del DUT.</li> </ol>	Hay continuidad en las 4 mediciones.

Tabla 9.5: Tests de validación de interfaz grupo DE9.

#### 9.2.2. Tests funcionales

ID	dlc	B7	B6	B5	B4	B3	B2	B1	B0
0x223	8	0x01	0x23	0x45	0x67	0x89	0xAB	0xCD	0xEF
0x222	7	—	0x23	0x45	0x67	0x89	0xAB	0xCD	0xEF
0x221	6	—	—	0x45	0x67	0x89	0xAB	0xCD	0xEF
0x220	5	—	—	—	0x67	0x89	0xAB	0xCD	0xEF
0x21F	4	—	—	—	—	0x89	0xAB	0xCD	0xEF
0x21E	3	—	—	—	—	—	0xAB	0xCD	0xEF
0x21D	2	—	—	—	—	—	—	0xCD	0xEF
0x21C	1	—	—	—	—	—	—	—	0xEF
0x21B	0	—	—	—	—	—	—	—	—

Tabla 9.6: Mensajes CAN para pruebas funcionales

Nombre de columna	Formato
Year	YYYY
Month	MM
Day	DD
Hour	hh
Minute	mm
Second	ss
Milisecond	msmsms

Tabla 9.7: Formato de timestamp esperado

ID	Banco de pruebas	Procedimiento	Criterios de aceptación
TST-FUN-1	1	<ol style="list-style-type: none"> <li>1. Insertar la tarjeta SD en el lector de la computadora y vaciarla.</li> <li>2. Insertar la tarjeta SD en el DUT.</li> <li>3. Programar el nodo CAN transmisor para el envío de los mensajes de la tabla 9.6 con una separación de 10ms.</li> <li>4. Desconectar la fuente CC y encenderla.</li> <li>5. Configurar la fuente CC para entregar 12V.</li> <li>6. Apagar la fuente CC.</li> <li>7. Conectar la fuente al nodo CAN receptor.</li> <li>8. Conectar el bus CAN al DUT.</li> <li>9. Encender la fuente CC.</li> <li>10. Comenzar el envío de los mensajes.</li> <li>11. Esperar 5 segundos.</li> <li>12. Apagar la fuente CC.</li> <li>13. Desconectar el bus CAN del DUT.</li> <li>14. Conectar la computadora al DUT mediante el cable USB Tipo A — USB Tipo C</li> <li>15. Abrir la carpeta del dispositivo de almacenamiento masivo</li> </ol>	<p>En el nuevo dispositivo de almacenamiento masivo existe un log en formato CSV que contiene los mensajes enviados con sus IDs en hexadecimal, los data bytes en hexadecimal y las horas de sus envíos en el formato detallado en la tabla 9.7. Al final del CSV, aparece un mensaje indicando que se guardaron todos los mensajes antes del apagado.</p>
TST-FUN-2	1	<ol style="list-style-type: none"> <li>1. Insertar la tarjeta SD en el lector de la computadora y vaciarla.</li> <li>2. Insertar la tarjeta SD en el DUT.</li> <li>3. Programar el nodo CAN transmisor para el envío de los mensajes de la tabla 9.6 con una separación de 2ms.</li> <li>4. Desconectar la fuente CC y encenderla.</li> <li>5. Configurar la fuente CC para entregar 12V.</li> <li>6. Apagar la fuente CC.</li> <li>7. Conectar la fuente al nodo CAN receptor.</li> <li>8. Conectar el bus CAN al DUT.</li> <li>9. Encender la fuente CC.</li> <li>10. Comenzar el envío de los mensajes.</li> <li>11. Esperar 5 segundos.</li> <li>12. Apagar la fuente CC.</li> <li>13. Desconectar el bus CAN del DUT.</li> <li>14. Conectar la computadora al DUT mediante el cable USB Tipo A — USB Tipo C</li> <li>15. Abrir la carpeta del dispositivo de almacenamiento masivo</li> </ol>	<p>En el nuevo dispositivo de almacenamiento masivo existe un log en formato CSV que contiene los mensajes enviados con sus IDs en hexadecimal, los data bytes en hexadecimal y las horas de sus envíos en el formato detallado en la tabla 9.7. Al final del CSV, aparece un mensaje indicando que se guardaron todos los mensajes antes del apagado.</p>

Tabla 9.8: Tests de validación funcional.

### 9.2.3. Tests de performance

ID	Banco de pruebas	Procedimiento	Criterios de aceptación
TST-PER-1	2	<ol style="list-style-type: none"> <li>1. Con el multímetro, medir el valor exacto de la resistencia de shunt <math>R</math>.</li> <li>2. Conectar dos puntas del osciloscopio a los bornes de la resistencia de shunt.</li> <li>3. Configurar el osciloscopio para visualizar las señales de las dos puntas en el rango de 0V a 14V y activar el modo <i>math</i> para visualizar la caída de tensión en la resistencia de shunt.</li> <li>4. Empezar la captura con el osciloscopio.</li> <li>5. Insertar la tarjeta SD en el lector de la computadora y vaciarla.</li> <li>6. Insertar la tarjeta SD en el DUT.</li> <li>7. Programar el nodo CAN transmisor para el envío de mensajes con una separación de 10ms.</li> <li>8. Desconectar la fuente CC y encenderla.</li> <li>9. Configurar la fuente CC para entregar 12V.</li> <li>10. Apagar la fuente CC.</li> <li>11. Conectar la fuente al nodo CAN receptor.</li> <li>12. Conectar el bus CAN al DUT.</li> <li>13. Encender la fuente CC.</li> <li>14. Comenzar el envío de los mensajes.</li> <li>15. Esperar 2 segundos.</li> <li>16. Con la función <i>average</i>, obtener la tensión promedio en la resistencia de shunt <math>V_{avg}</math>.</li> <li>17. Apagar la fuente CC.</li> <li>18. Desconectar el bus CAN del DUT.</li> <li>19. Detener la captura con el osciloscopio</li> <li>20. Para todo el tiempo de captura del osciloscopio, obtener el máximo de la tensión en la resistencia de shunt <math>V_p</math>.</li> </ol>	<p>Se cumplen las siguientes condiciones:</p> <ul style="list-style-type: none"> <li>■ <math>3W &gt; 12V \cdot \frac{V_p}{R}</math></li> <li>■ <math>1W &gt; 12V \cdot \frac{V_{avg}}{R}</math></li> </ul>

Tabla 9.9: Tests de validación de performance.

ID	Banco de pruebas	Procedimiento	Criterios de aceptación
TST-MEC-1	4	Descrito en la sección 4.1.2.4.2 de [2]	Descrito en la sección 4.1.2.4.3 de [2].
TST-MEC-2	5	Descrito en la sección 12.2 de [16]	Descrito en la sección 12.3 de [16].

Tabla 9.10: Tests de validación de interfaz de grupo mecánica.

#### 9.2.4. Tests de implementación

ID	Banco de pruebas	Procedimiento	Criterios de aceptación
TST-IMP-1	3	1. Pesar el DUT.	El DUT pesa menos de 500g.

Tabla 9.11: Tests de validación de implementación.

#### 9.3. Matriz de trazabilidad

En la tabla 9.12 se puede ver la matriz de trazabilidad. Los requerimientos que no cuentan con un test asociado son validados mediante el proceso de diseño.

#### 9.4. Plan de Tests

La figura 9.3 muestra el orden en el cual se deben realizar los tests.

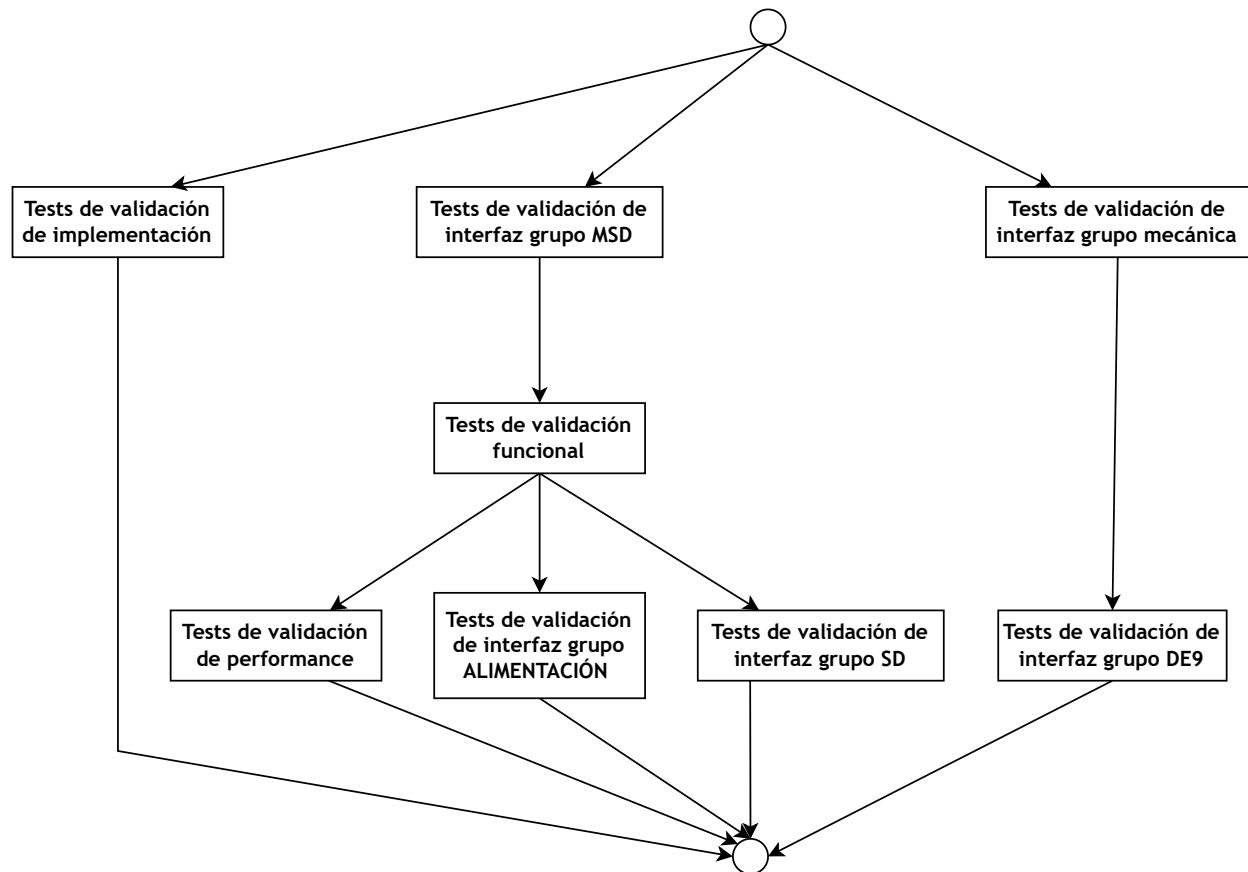


Figura 9.3: Plan de tests

Requerimiento	Especificación	Test
REQ-LOG-1	FUN-LOG-01 FUN-LOG-02	TST-FUN-1 TST-FUN-2
REQ-LOG-2	FUN-LOG-03	TST-FUN-1 TST-FUN-2
REQ-LOG-3	FUN-LOG-04	TST-FUN-1 TST-FUN-2
REQ-LOG-4	INT-PC-01	TST-FUN-1 TST-FUN-2
REQ-LOG-5	FUN-LOG-05	TST-FUN-1 TST-FUN-2
REQ-LOG-6	FUN-LOG-06	TST-FUN-1 TST-FUN-2
REQ-LOG-7	FUN-LOG-07 INT-PC-04	TST-MSD-1
REQ-POW-1	INT-VIN-01	TST-ALIM-1 TST-FUN-1 TST-FUN-2
REQ-POW-2	INT-VIN-02	TST-ALIM-2 TST-ALIM-3
REQ-POW-3	PER-POW-01	TST-PER-1
REQ-POW-4	PER-POW-02	TST-PER-1
REQ-POW-5	PER-POW-03	—
REQ-POW-6	FUN-POW-01	TST-FUN-1 TST-FUN-2
REQ-CON-1	INT-PC-02 INT-PC-03	TST-MSD-1
REQ-CON-2	INT-CAN-01	TST-DE9-1
REQ-MEC-1	IMP-DIM-01 IMP-DIM-02	TST-IMP-1
REQ-MEC-2	INT-MEC-01	TST-MEC-2
REQ-MEC-3	INT-MEC-02	—
REQ-MEC-4	INT-MEC-03	TST-MEC-1
REQ-MISC-1	FUN-MISC-01	—
REQ-MISC-2	FUN-MISC-02	—
REQ-MISC-3	IMP-COS-01	—
REQ-MISC-4	IMP-OPE-01	—

Tabla 9.12: Matriz de trazabilidad.

## 10. Análisis de factibilidad

Esta sección tiene como objetivo evaluar la factibilidad del proyecto en cuatro áreas: tecnológica, temporal, económica y legal-civil. En la presente sección se lleva a cabo un análisis de cada área, destacando los factores críticos y estableciendo las estrategias necesarias para lograr la ejecución exitosa del proyecto.

### 10.1. Factibilidad tecnológica

La figura 10.1 muestra un diagrama modular de interfaces del producto.

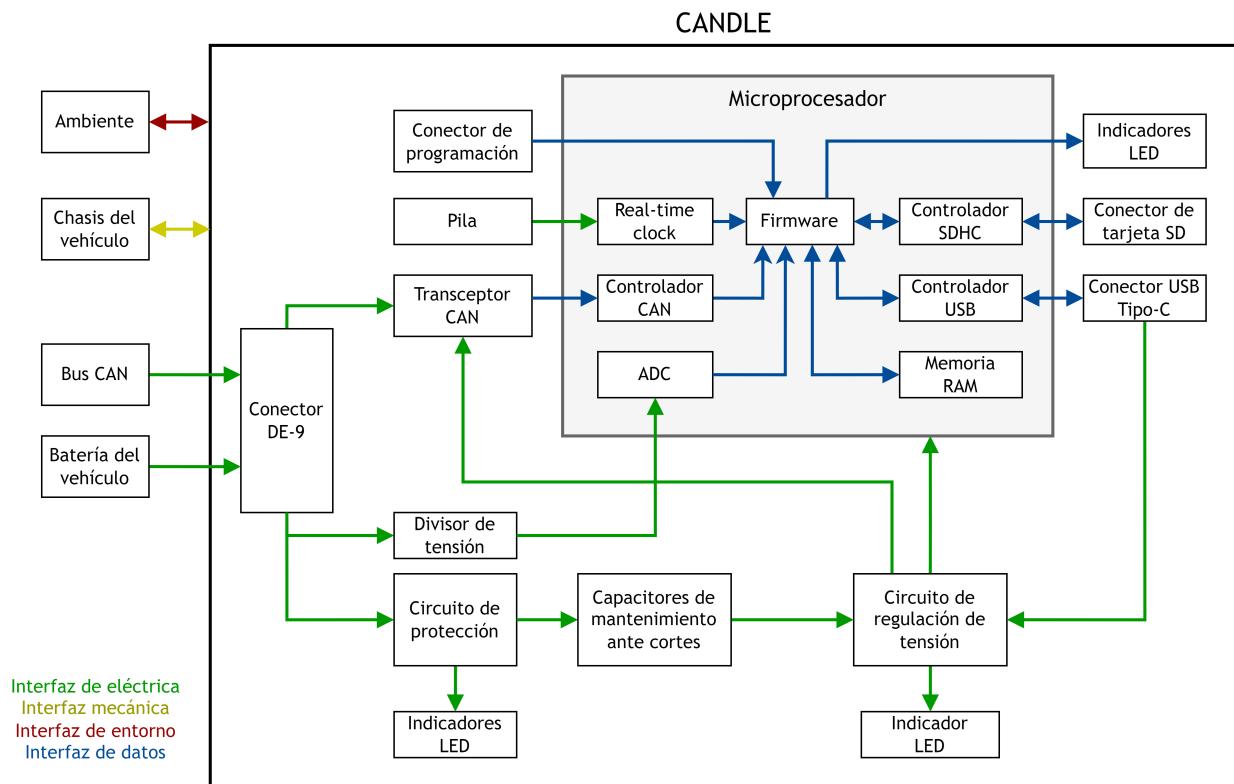


Figura 10.1: Diagrama de módulos de la solución

#### 10.1.1. Propuestas alternativas de diseño

##### 10.1.1.1 Microcontrolador

El microcontrolador es la unidad de procesamiento del sistema. Se encargará de realizar las siguientes tareas:

- Recepción de mensajes CAN.
- Comunicación con la tarjeta SD para el almacenamiento de datos.
- Comunicación con la computadora a través de un bus de tipo USB 2.0.
- Detección de cortes en la línea de alimentación (*blackout*).
- Control de los indicadores LED para visualizar el estado del sistema.
- Obtener la fecha y horario actual.

Para lograr implementar la funcionalidad requerida, el microcontrolador empleado debe poseer controladores periféricos integrados para todas las interfaces que serán utilizadas. Por otro lado, la implementación del firmware requiere de un *buffer* que se almacena en la memoria volátil del microcontrolador. Otras características relevantes serán la velocidad de *clock*, la memoria FLASH donde se almacenará el código del programa, y el costo.

Las características a comparar son:

- Velocidad de *clock*.
- Memoria RAM.
- Memoria FLASH.
- Periférico RTC.
- Modo de bajo consumo con mantenimiento del módulo RTC.
- Periférico SDHC.
- Periférico CAN.
- Periférico USB.
- Existencia de una placa de desarrollo provista por el fabricante.
- Existencia de bibliotecas provistas por el fabricante.
- Costo.

**Estimación del uso de memoria** Uno de los usos de la memoria RAM estará asociado al procesamiento de los mensajes CAN recibidos por el microcontrolador. Los mensajes recibidos serán procesados para ser almacenados en formato CSV en una tarjeta SD. Las filas del archivo CSV serán almacenados en un *buffer*, para luego ser transferidas a la tarjeta SD. La velocidad máxima a la cual se deben almacenar datos sucede cuando se envían mensajes de tipo DATA de 8 bytes a 125 kbit/s con un periodo de 2 ms, es decir una tasa de mensajes de 500 msg/s.

Para los propósitos de obtener una estimación de la memoria requerida, se asume que la tarjeta SD es escrita cada 100 ms. Por otro lado, se debe determinar el formato con el que se almacenarán los datos. En la tabla 10.1 se pueden ver las columnas que tendrá cada fila del archivo CSV generado y la cantidad de bytes asociados a cada una. La cantidad de bytes de datos por fila es de 37 bytes. Adicionalmente, los campos de cada columna se separan con un carácter de coma, y al final de cada fila se inserta un carácter de nueva línea, lo cual incrementa la cantidad de bytes que conforman cada línea del CSV en 17 bytes. Esto resulta en un total de 54 bytes por fila.

En función de los resultados previos, el tamaño de *buffer* que debe ser almacenado en la memoria RAM se obtiene como se indica a continuación:

$$\text{Tamaño buffer} = 500 \text{ msg/s} \cdot 54 \text{ By/msg} \cdot 0,1 \text{ s} = 2700 \text{ By}$$

Se estima que el programa usa como máximo 10 KBy de RAM para un *heap* de memoria RAM, y otros 10 KBy distribuidos en buffers para los drivers de los distintos protocolos de comunicación.

por lo que es necesario que el microcontrolador tenga al menos 33,7 kBy de memoria RAM. Por otro lado, se estima que el programa va a ocupar como máximo 100 kBy de memoria FLASH.

Columna del CSV	Número de bytes
Año	4
Mes	2
Día	2
Hora	2
Minutos	2
Segundos	2
Milisegundos	3
ID	3
Data Length	1
B7	2
B6	2
B5	2
B4	2
B3	2
B2	2
B1	2
B0	2

Tabla 10.1: Descripción de las columnas de los archivos CSV generados.

**Selección del microcontrolador** En la tabla 10.2 se muestra una tabla comparativa con las alternativas de diseño analizadas para el microcontrolador.

#### 10.1.1.2 Transceptor CAN

Se evalúan las siguientes opciones para el transceptor CAN:

- NXP TJA1055
- Microchip ATA6560
- Microchip MCP2561

Las características a comparar son:

- Velocidad máxima
- Costo
- Alimentación
- Tensión digital de comunicación con el microcontrolador

La tabla 10.3 muestra la comparación entre las alternativas analizadas.

Fabricante	Familia	Modelo	Precio Int. (USD)	Flash	RAM	Frecuencia	CAN	USB	SD/MMC	Velocidad del bus SD
Espressif	ESP	ESP32-D0WDQ6	3,3	4 MB	520 kB	240 MHz	Sí	No	Sí	
Espressif	ESP	ESP8266EX	1,6	4 MB	96 kB	160 MHz	No	No	Sí	
Microchip	ATmega	ATMEGA328PB-AUR	1,72	32 kB	2 kB	20 MHz	No	No	No	
Microchip	ATmega	ATMEGA32U4RC-AU	4,92	32 kB	2,5 kB	16 MHz	No	Sí	No	
NXP	LPC	LPC4337JBD144E	21,88	1 MB	136 kB	204 MHz	Sí	Sí	Sí	
NXP	Kinetis K10	MK10DX32VLF5	4,36	32 kB	16 kB	50 MHz	No	No	No	
<b>NXP</b>		<b>MK20DX256VLK10R</b>	<b>9,41</b>	<b>256 kB</b>	<b>64 kB</b>	<b>100 MHz</b>	<b>Sí</b>	<b>Sí</b>	<b>Sí</b>	<b>25/50 MHz</b>
NXP	Kinetis K20	MK20DX256VLK7	9,38	256 kB	64 kB	72 MHz	Sí	Sí	Sí	No
NXP	Kinetis K20	MK20DX64VLH5	7,43	64 kB	16 kB	50 MHz	No	Sí	No	
NXP	Kinetis K60	MK60DN512VLQ10	19,66	512 kB	128 kB	100 MHz	Sí	Sí	Sí	No
NXP	Kinetis K60	MK60FN1M0VLQ15	24,2	1 MB	128 kB	150 MHz	Sí	Sí	Sí	No
<b>NXP</b>		<b>MK64FN1M0VLL12</b>	<b>20,9</b>	<b>1 MB</b>	<b>256 kB</b>	<b>120 MHz</b>	<b>Sí</b>	<b>Sí</b>	<b>Sí</b>	<b>25/50 MHz</b>
<b>NXP</b>		<b>MK64FN1M0VLLQ12</b>	<b>22,05</b>	<b>1 MB</b>	<b>256 kB</b>	<b>120 MHz</b>	<b>Sí</b>	<b>Sí</b>	<b>Sí</b>	<b>25/50 MHz</b>
ST	STM32	STM32F103C8T6	7,26	64 kB	20 kB	72 MHz	Sí	Sí	Sí	No
ST	STM32	STM32F302RDT6TR	4,53	384 kB	64 kB	72 MHz	Sí	Sí	Sí	No
ST	STM32	STM32F303CCT6	9,29	256 kB	40 kB	72 MHz	Sí	Sí	Sí	No
ST	STM32	STM32L431RBT6	7,66	128 kB	64 kB	80 MHz	Sí	No	Sí	
ST	STM32	STM32L475RCT3	11,12	256 kB	128 kB	80 MHz	Sí	Sí	Sí	48 MHz

Tabla 10.2: Tabla comparativa para las alternativas de diseño del microcontrolador.

Parámetro	TJA1055 (NXP)	ATA6561 (Microchip)	MCP2562 (Microchip)
Velocidad máxima	125KBits/s	5MBits/s	1MBits/s
Costo	2.62 USD	0.48 USD	1.22 USD
Alimentación	4.75V a 5.25V	4.5V a 5.5V	4.5V a 5.5V
Tensión digital	LOW: -0.2V a 0.8V HIGH: 2.0V a VCC+0.3V	VVIO: 2.8V a 5.5V LOW: -0.3V a 0.3 x VVIO HIGH: 0.7xVVIO a VVIO+0.3V	Acepta niveles de 1.8V a 5.5V

Tabla 10.3: Comparación entre las alternativas de diseño del transceptor CAN.

El modelo de microcontrolador seleccionado para el producto final es el MK20DX256VLK10R. Este dispositivo cuenta con todos los periféricos requeridos, la memoria FLASH y RAM se adecúan a los requerimientos y el precio se encuentra entre los más bajos de los modelos analizado.

Adicionalmente, existen otros dos modelos de microcontrolador pertenecientes a la misma familia de productos del fabricante NXP que se identificaron para el desarrollo del prototipo del producto. Estos modelos se encuentran disponibles en el mercado local, cuentan con placas de desarrollo y bibliotecas de firmware del fabricante, y además el firmware desarrollado para estos modelos es fácilmente adaptable al microcontrolador MK20DX256VLK10R mediante la configuración del entorno integrado de desarrollo (IDE por sus siglas en inglés, *Integrated Development Environment*).

#### 10.1.1.3 Transceptor CAN

El modelo de circuito integrado seleccionado para el desarrollo del producto final es el ATA6561 dado que se adecúa a los requerimientos de velocidad y niveles lógicos, y adicionalmente es la alternativa de menor precio.

#### 10.1.2. DFMEA

En esta sección se presenta un análisis de modos y efectos de fallas del diseño (DFMEA por sus siglas en inglés, *Design Failure Mode and Effects Analysis*). Mediante esta metodología de análisis se identificaron las potenciales fallas que puede experimentar el sistema para establecer las acciones a tomar para mejorar la capacidad de prevención y/o detección de las fallas.

En esta metodología, a cada modo de falla se asocia un efecto potencial de la falla. Este efecto potencial de falla se corresponde con un nivel de severidad de la falla mediante un valor numérico entre 1 y 5. Luego, a partir de la causa potencial de la falla se determina una estimación de la probabilidad de ocurrencia de esta causa mediante un valor numérico entre 1 y 5. A continuación, se enumeran los controles de prevención y detección actuales y se determina un valor numérico entre 1 y 5 que indique el grado de detección de esta falla. Por último, el número de prioridad de riesgo (RPN por sus siglas en inglés, *Risk Priority Number*) indica la prioridad con las que se deben implementar mejoras de diseño para mejorar la prevención o detección de cada tipo de falla. Esta prioridad se puede calcular como el producto de la severidad, ocurrencia y detección.

En la tabla 10.4 se pueden ver los resultados del análisis. A partir de los resultados del DFMEA se implementaron las acciones recomendadas en el diseño del prototipo.

Design Failure Mode and Effects Analysis							
Subsistema	Modo potencial de falla	Efecto potencial de falla	Severidad	Causa potencial de la falla	Ocurrencia	Controles preventivos actuales	Detección actual
CAN Transceiver	Des-sincronización del clock	Pérdida de datos	4	Jitter en el clock del BUS / Problema con el PLL interno del transceiver	1	Diseño robusto para minimizar el jitter y elección de un transceiver con buenas prestaciones	No hay un sistema de detección hasta el momento
	Entrada de alimentación	Daño permanente de diversos componentes del sistema	5	Tensión de entrada fuera del rango de entrada de la fuente de switching	1	Todavía no hay control	5
Conector DE-9	Conector desconectado o haciendo mal contacto	Pérdida de datos	4	Error del usuario / Elemento extenso / enganchado en el cable o dentro del conector	2	Fijación del conector mecánicamente (mediante tornillos)	Inspección visual y/o manual del conector
	Congelamiento de la ejecución del programa	Pérdida de datos	4	Error en el software / Deadlock	1	Todavía no hay control	2
Módulo Real-Time Clock	Reinicio indeseado del RTC / De-sincronización del RTC	Registro de datos con marca de tiempo incorrecta	3	Ruido eléctrico / Desconexión temporal de la batería del RTC debido a vibraciones	1	Diseño mecánica y eléctricamente robusto de la alimentación del RTC	Inspección a posteriori de los archivos de datos generados
	Daño permanente del diodo de protección / Daño permanente del MOSFET de paso	Daño del microcontrolador y otros circuitos lógicos	5	Corriente de <i>inrush</i> provocado por los capacitores de mantenimiento	1	Todavía no hay control	1
Entrada de alimentación						Inspección visual de un LED indicador	Realizar simulaciones para asegurar la robustez del circuito

Tabla 10.4: Análisis de modos y efectos de fallas del diseño.

## 10.2. Factibilidad de tiempos

### 10.2.1. Planificación

La tabla 10.5 lista todas las tareas a realizar para completar el proyecto y sus dependencias. Además, se incluye el tiempo optimista de realización, el más probable, y el pesimista. Se modela la duración de las tareas como una variable aleatoria de distribución beta, y se obtiene el valor esperado y el desvío estándar de la duración de cada una de las tareas.

Actividad	Descripción	Precedencia	To	Tm	Tp	Te	Tiempo acum	Desvio Estandar
1	Identificación de problemática o necesidad		32	38	50	39	39	3.00
2	Selección de proyecto	1	9	11	13	11	50	0.67
3	Definición de requerimientos del cliente	2	12	16	20	16	66	1.33
4	Investigación del mercado y tecnologías existentes	2	4	6	8	6		0.67
5	Análisis de factibilidad tecnológica	3,4	6	10	14	10	76	1.33
6	Análisis de factibilidad económica	3,4	6	10	14	10		1.33
7	Ánalisis de factibilidad legal y responsabilidad civil	3,4	6	10	14	10		1.33
8	Definición del alcance del proyecto	5	5	7	9	7	83	0.67
9	Especificaciones de diseño	8	7	10	13	10	93	1.00
10	Diseño de interfaces	9	2	3	5	3	96	0.50
11	Diseño de plan de validación	9	8	10	12	10		0.67
Hardware			81		54			0.00
13	Selección del microcontrolador y periféricos	9,10	12	16	20	16	112	1.33
14	Diseño de circuito y selección de componentes	9,10,13	8	10	12	10		0.67
15	Ingeniería de detalle	14	25	30	35	30		1.67
16	Diseño del PCB	13,14,15	8	10	14	10		1.00
17	Desarrollo u obtención de los controladores	13,21,15	10	11	16	12		1.00
18	Adquisición de componentes	13,14,15	10	15	20	15		1.67
19	Manufactura del PCB	16	12	15	20	15		1.33
Software			170		113			0.00
21	Diseño de la arquitectura del software	13	40	50	60	50	162	3.33
22	Desarrollo de la aplicación	21	100	120	150	122	284	8.33
23	Armado del circuito final	18,19	10	15	20	15		1.67
Mecánica			20		13			0.00
25	Diseño de la carcasa	16	4	7	9	7		0.83
26	Diseño del sistema de sujeción	25	4	7	9	7		0.83
27	Manufactura de la carcasa	25	8	10	12	10		0.67
28	Adquisición de componentes de sujeción	26	4	6	8	6		0.67
29	Construcción del prototipo	23,27,28	15	20	25	20		1.67
30	Realización de pruebas mecánicas	27,28,29	5	6	8	6		0.50
31	Realización de pruebas de adquisición de datos	22,23,29	25	30	40	31	315	2.50
32	Realización de pruebas finales y validación	29,30,31	8	10	14	10	325	1.00
33	Documentación técnica	32	4	5	10	6	331	1.00

Tiempo estimado total	Desvio estandar
331	10.26049598

Tabla 10.5: Definición de tareas, duraciones (en días), y precedencias. Las tareas marcadas en rojo forman parte del camino crítico hallado mediante el diagrama de PERT (figura 10.2).

Con los valores de valor esperado y varianza de la duración de las tareas, se realiza el diagrama de PERT (figura 10.2). Se marca en rojo el camino crítico. La duración esperada del proyecto es:

$$t_e = 331 \text{ días hábiles}$$

con un desvío estándar de:

$$\sigma_t = 10,26 \text{ días hábiles}$$

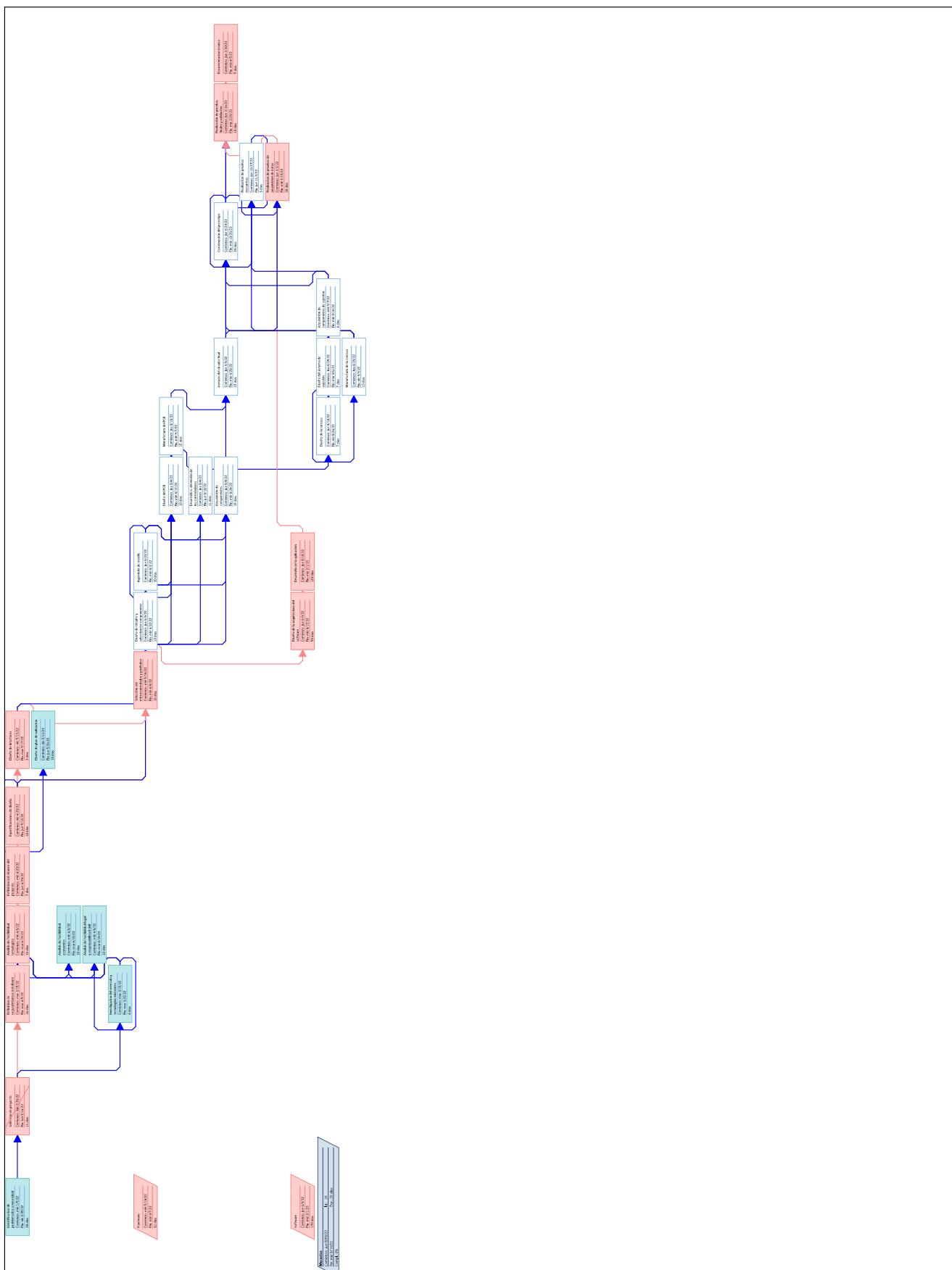


Figura 10.2: Diagrama de PERT. Se marca en rojo el camino crítico.

Se realiza una simulación de Montecarlo. La simulación concluye que hay un 81.8 % de probabilidad de terminar el proyecto en menos de 331 días hábiles (figura 10.3), y un 94.3 % de probabilidad de terminar el proyecto en menos de 338 días hábiles (figura 10.4).

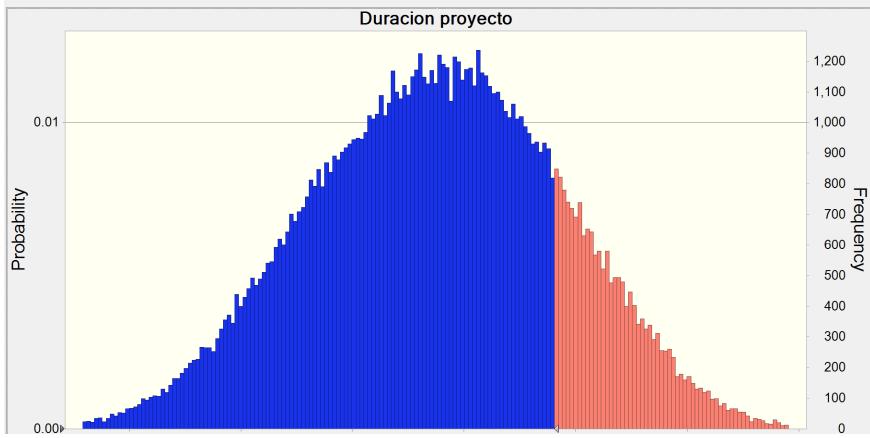


Figura 10.3: Simulación de Montecarlo: duración máxima con certeza del 80 %

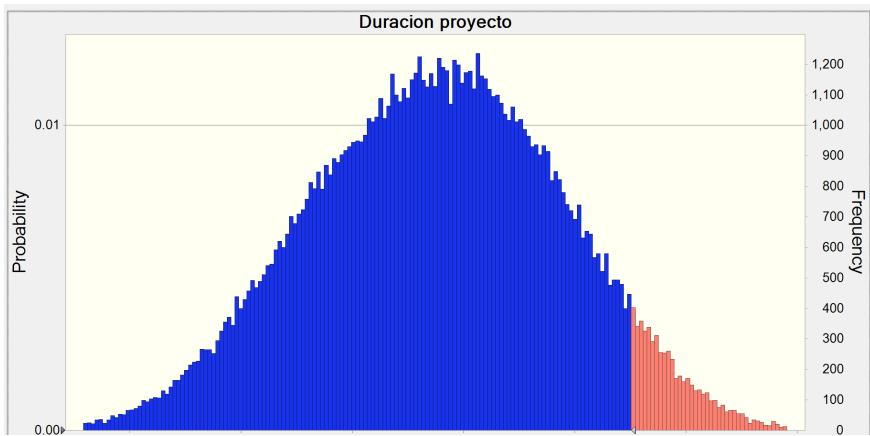


Figura 10.4: Simulación de Montecarlo: duración máxima con certeza del 95 %

### 10.2.2. Programación

Para programar las tareas, se utiliza un diagrama de Gantt (figura 10.5). Para su confección, se considera que el equipo de trabajo está formado por 4 personas, por lo que no puede haber más de 4 tareas en paralelo.

## 10.3. Factibilidad económica

### 10.3.1. Estudio de Mercado

En los últimos años hubo un crecimiento significativo en el desarrollo de vehículos eléctricos en comparación a los clásicos de combustión. Diversas universidades y centros de investigación pusieron su foco en este tipo de vehículos, ya sea para fines didácticos, de investigación, o para participar en competencias. Una de las competencias más reconocidas es la de fórmula SAE, cuya categoría de vehículos eléctricos ya supera el 50 % de la cantidad de vehículos de combustión. Las universidades más grandes que desarrollan este tipo de vehículos tienen en promedio 15-20 equipos de estudiantes y/o investigadores trabajando en estos proyectos.

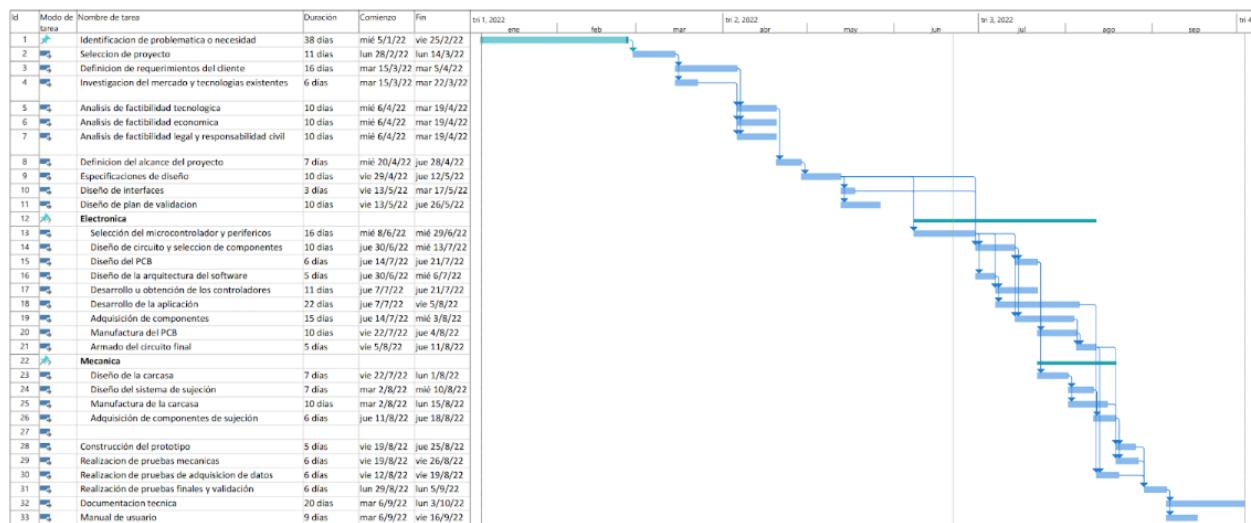


Figura 10.5: Diagrama de Gantt

Debido al creciente interés en el desarrollo de estos vehículos, también crece la necesidad de poder registrar su funcionamiento para mejoras o identificación de fallas.

Se hizo una investigación de los productos existentes en el mercado, sus funcionalidades, y el precio de venta. Se obtuvo como resultado que hay presentes en el mercado dispositivos de adquisición de datos del bus CAN con precios entre 700USD y 1500USD.

### 10.3.2. Modelo de negocios

El producto provee una solución más accesible a esta creciente demanda con un precio de venta de 350USD. Se estiman ventas de 200 unidades en el primer año de actividad. Los primeros años se enfocaran en la expansión de la ocupación del mercado, reflejando un incremento en las unidades estimadas de venta.

### 10.3.3. Costo de producción

En la tabla 10.6 se observan los costos de componentes. En la tabla 10.7 se pueden ver los costos de desarrollo, que contemplan el trabajo de 4 desarrolladores, compra de materiales para pruebas y ensayos, armado de prototipos, y gastos administrativos. Los costos de operación anual se detallan en la tabla 10.8.

### 10.3.4. Flujo de fondos anual

Para el flujo de fondos se considera un impuesto a las ganancias del 35 % y una tasa de descuento del 3 %. La tabla 10.9 muestra el flujo de fondos del proyecto. Con una inversión inicial de 21.150USD, se pronostican ganancias a partir del 4<sup>to</sup> año. Al cabo de 10 años, el VAN es 496.000USD.

Cant.	Componente	Importe unitario (USD)	Precio total (USD)
2	Capacitor 1000uF/50V	\$0.3466	\$0.6932
1	Capacitor 100uF/50V	\$0.0495	\$0.0495
1	Capacitor 4.7uF/50V	\$0.0546	\$0.0546
3	Capacitor 4.7uF/16V	\$0.0229	\$0.0686
1	Capacitor 1uF/50V	\$0.0047	\$0.0047
2	Capacitor 1uF/16V	\$0.0031	\$0.0062
1	Capacitor 100nF/50V	\$0.0110	\$0.0110
19	Capacitor 100nF/16V	\$0.0260	\$0.4942
2	Capacitor 10nF/16V	\$0.0130	\$0.0260
3	Resistencia 0R	\$0.0109	\$0.0328
2	Resistencia 1k	\$0.0060	\$0.0120
1	Resistencia 100k	\$0.0060	\$0.0060
4	Resistencia 1k8	\$0.0060	\$0.0240
6	Resistencia 3k3	\$0.0060	\$0.0361
2	Resistencia 22k	\$0.0060	\$0.0120
1	Resistencia 220k	\$0.0060	\$0.0060
6	Resistencia 220	\$0.0060	\$0.0361
2	Resistencia 330	\$0.0060	\$0.0120
2	Resistencia 330	\$0.0060	\$0.0120
2	Resistencia 120	\$0.0060	\$0.0120
7	Resistencia 10k	\$0.0060	\$0.0421
1	Inductor 330uH	\$1.2223	\$1.2223
6	Diodo LED rojo	\$0.1390	\$0.8340
1	Diodo LED verde	\$0.1390	\$0.1390
6	Diodo Schottky B5819W	\$0.0464	\$0.2782
1	Diodo Zener 1N4749ATR	\$0.0546	\$0.0546
1	Diodo TVS SMAJ64CA	\$0.0895	\$0.0895
1	Diodo TVS SMAJ6.0CA	\$0.0749	\$0.0749
1	Diodo Zener 1N5225B	\$0.0210	\$0.0210
1	Portapilas BH-25C-1	\$0.0547	\$0.0547
2	Conecotor hembra DE-9	\$1.2302	\$2.4604
1	Conecotor USB-C	\$0.5616	\$0.5616

2	Pin Header 1x4 (2.54mm)	\$0.2000	\$0.4000
1	Conektor SD	\$0.8181	\$0.8181
1	Pin Header 2x5 (1.27mm)	\$0.1800	\$0.1800
1	MOSFET Si4948BEY	\$0.5565	\$0.5565
1	Regulador de Tensión LM2596S-ADJ	\$2.4253	\$2.4253
1	Regulador de Tensión AMS1117	\$0.2294	\$0.2294
1	Tranceiver CAN MCP2562T	\$0.9100	\$0.9100
1	Microcontrolador MK64FN1M0VLL12	\$14.1677	\$14.1677
1	Cristal 32.768 kHz	\$0.1131	\$0.1131
1	Cristal 16MHz	\$0.0680	\$0.0680
1	Placa PCB	\$3.0000	\$3.0000
1	Carcasa	\$5.0000	\$5.0000
4	Tornillo 1/8"	\$0.1500	\$0.6000
4	Tuerca 1/8"	\$0.0500	\$0.2000
		<b>TOTAL</b>	<b>\$36.1091</b>

Tabla 10.6: Detalle de costos de componentes de un CANDLE

Detalle	Importe (USD)
Trabajo de 4 desarrolladores	\$20,000.00
Materiales de desarrollo electrónico	\$100.00
Material de prototipado mecánico	\$50.00
Gastos administrativos	\$1,000.00
<b>TOTAL</b>	<b>\$21,150.00</b>

Tabla 10.7: Detalle de costos de desarrollo del CANDLE

Detalle	Importe (USD)
Sueldo 4 ingenieros	\$72,000.00
Gastos logísticos	\$2,400.00
Gastos administrativos	\$3,600.00
Gastos legales	\$1,200.00

Tabla 10.8: Detalle de costos fijos anuales

Año	0	1	2	3	4	5	6	7	8	9	10
Unidades	200	250	315	395	500	625	750	800	850	900	
Ventas	\$70,000.00	\$87,500.00	\$110,250.00	\$138,250.00	\$175,000.00	\$218,750.00	\$262,500.00	\$280,000.00	\$297,500.00	\$315,000.00	
Costos	\$82,000.00	\$83,900.00	\$86,370.00	\$89,410.00	\$93,400.00	\$98,150.00	\$102,900.00	\$104,800.00	\$106,700.00	\$108,600.00	
UAIG	-\$12,000.00	\$3,600.00	\$23,880.00	\$48,840.00	\$81,600.00	\$120,600.00	\$159,600.00	\$175,200.00	\$190,800.00	\$206,400.00	
IG	-\$4,200.00	\$1,260.00	\$8,358.00	\$17,094.00	\$28,560.00	\$42,210.00	\$55,860.00	\$61,320.00	\$66,780.00	\$72,240.00	
Uneta	-\$7,800.00	\$2,340.00	\$15,522.00	\$31,746.00	\$53,040.00	\$78,390.00	\$103,740.00	\$113,880.00	\$124,020.00	\$134,160.00	
FEO	-\$7,800.00	\$2,340.00	\$15,522.00	\$31,746.00	\$53,040.00	\$78,390.00	\$103,740.00	\$113,880.00	\$124,020.00	\$134,160.00	
Inversiones	-\$21,150.00	-	-	-	-	-	-	-	-	-	
FFN	-\$21,150.00	-\$7,800.00	\$2,340.00	\$15,522.00	\$31,746.00	\$53,040.00	\$78,390.00	\$103,740.00	\$113,880.00	\$124,020.00	\$134,160.00
VA	-\$21,150.00	-\$7,572.82	\$2,205.67	\$14,204.83	\$28,205.91	\$45,752.77	\$65,650.39	\$84,350.11	\$89,897.92	\$95,051.00	\$99,827.64
VAN	-\$21,150.00	-\$28,722.82	-\$26,517.14	-\$12,312.31	\$15,893.60	\$61,646.37	\$127,296.76	\$211,646.87	\$301,544.80	\$396,595.80	\$496,423.44

Tabla 10.9: Diagrama de flujo de fondos

## **10.4. Factibilidad legal y de responsabilidad civil**

Aunque no es imprescindible constituir una sociedad para llevar a cabo actividades como responsable inscripto, optar por hacerlo puede acarrear diversas ventajas para los individuos involucrados. Entre estas, se destaca la mitigación de riesgos asociados a la pérdida del patrimonio personal al establecer una clara separación entre el patrimonio comercial y el personal, así como la distribución de obligaciones y responsabilidades entre los socios.

Para este proyecto en particular, la constitución de una Sociedad por Acciones Simplificada (S.A.S.) se presenta como la opción más adecuada. Este tipo de sociedad fue introducido en Argentina en 2017 como parte de la Ley 27.349 de Apoyo al Capital Emprendedor. Entre sus características más destacadas, se encuentra el hecho de que requiere un capital mínimo sustancialmente menor en comparación con otros tipos de sociedades, como la Sociedad de Responsabilidad Limitada (S.R.L.) o la Sociedad Anónima (S.A.). En el caso de una S.A.S., el capital inicial necesario equivale a dos salarios mínimos vitales y móviles, con la opción de invertir solo el 25% en la constitución inicial y el resto en los dos años siguientes, lo que proporciona flexibilidad adicional para los socios.

Al igual que en una S.R.L., una S.A.S. garantiza la responsabilidad limitada de los socios y puede ser unipersonal o contar con varios socios. Además, el proceso de designación de administradores cumple con los requisitos establecidos por la AFIP para asegurar la conformidad con las regulaciones fiscales vigentes. Es importante destacar que la constitución de una sociedad de este tipo es un proceso ágil y se realiza de forma online, lo que la hace especialmente adecuada para pequeñas empresas y emprendimientos. En resumen, la elección de constituir una Sociedad por Acciones Simplificada puede proporcionar una estructura legal sólida y flexible para el desarrollo de actividades comerciales en Argentina.

### **10.4.1. Leyes, Normas y Regulaciones**

#### **Ley de Defensa del Consumidor (Ley 24.240):**

Artículo 1: Objeto. La presente ley tiene por objeto la defensa del consumidor o usuario, entendiéndose por tal a toda persona física o jurídica que adquiere o utiliza bienes o servicios en forma gratuita u onerosa como destinatario final, en beneficio propio o de su grupo familiar o social. Artículo 11: Obligaciones del proveedor. El proveedor está obligado a suministrar al consumidor productos o servicios que sean conformes con los contratos que celebre y responder, en consecuencia, por los vicios o defectos de toda índole que presenten los productos o servicios.

#### **Regulación de Ensayos de Laboratorio en Electrónica y Comunicaciones (IRAM 62408):**

Esta norma establece los requisitos técnicos y metodológicos para la realización de ensayos de laboratorio en equipos electrónicos y de comunicaciones. Los ensayos deben llevarse a cabo de acuerdo con las disposiciones detalladas en esta norma para verificar el cumplimiento de las regulaciones de seguridad y calidad aplicables.

#### **Normas de Seguridad Eléctrica (IRAM):**

Las normas de seguridad eléctrica del IRAM establecen los requisitos para la fabricación y comercialización de productos electrónicos, incluyendo medidas de protección contra descargas eléctricas, cortocircuitos y otros riesgos eléctricos. Estas normas son de aplicación obligatoria para garantizar la seguridad de los usuarios y la calidad de los productos eléctricos.

**Normativas de Importación y Aduanas:**

Las regulaciones aduaneras de Argentina establecen los procedimientos y requisitos para la importación de productos electrónicos, incluyendo la presentación de documentación aduanera adecuada y el cumplimiento de normativas de homologación y seguridad establecidas por las autoridades competentes.

## 11. Ingeniería de detalle

### 11.1. Hardware

En esta sección se especifican los detalles de implementación del hardware del prototipo. El diseño y elección de los componentes electrónicos debe atenerse a las especificaciones del producto. Se debe procurar respetar las especificaciones que no conllevan un test asociado, sino que deben cumplirse por diseño. En particular, el rango de temperatura de todos los componentes seleccionados debe respetar la especificación de temperatura IMP-OPE-01.

#### 11.1.1. Diagrama de bloques

En la figura 11.1 se puede observar un diagrama en bloques del diseño de hardware del prototipo.

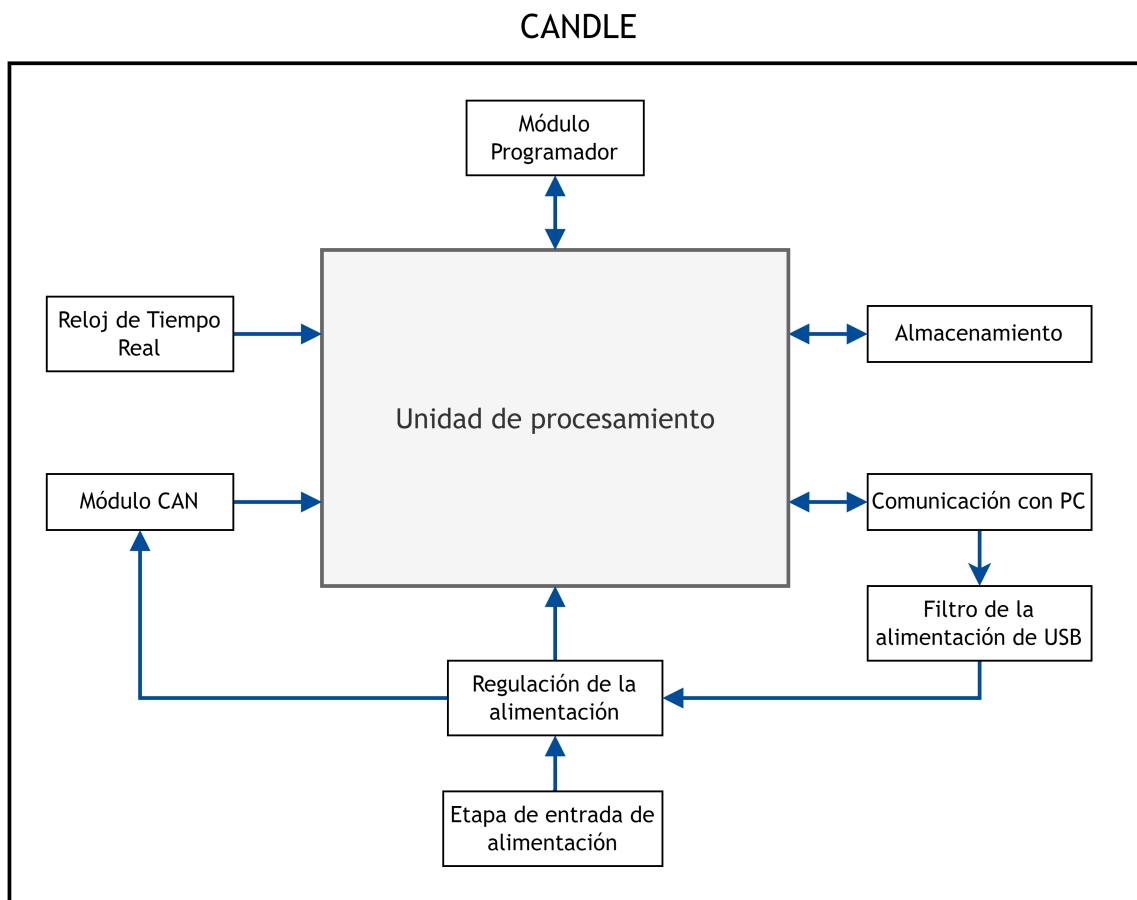


Figura 11.1: Diagrama de bloques de hardware

#### 11.1.2. Descripción de cada módulo

##### 11.1.2.1 Módulo de unidad de procesamiento

El módulo de unidad de procesamiento se compone de un microcontrolador, un oscilador basado en un cristal de cuarzo, un circuito de detección de cortes de energía, y un conjunto de indicadores LED.

En la figura 11.2 se muestra el conexionado del microcontrolador y el circuito oscilador. El cristal elegido tiene una frecuencia de 16 MHz y se emplean los cristales internos del MCU para la simplificación del circuito. El PLL (del inglés *Phase-Locked Loop*) interno del MCU fue configurado con un factor de división de 4, que obtiene una frecuencia de 4 MHz a partir de la frecuencia 16 MHz del cristal, y un factor de multiplicación de 30, que obtiene una frecuencia de 120 MHz a partir de la frecuencia de 4 MHz. Así, la frecuencia de sistema para el MCU es la máxima posible dentro de las especificaciones.

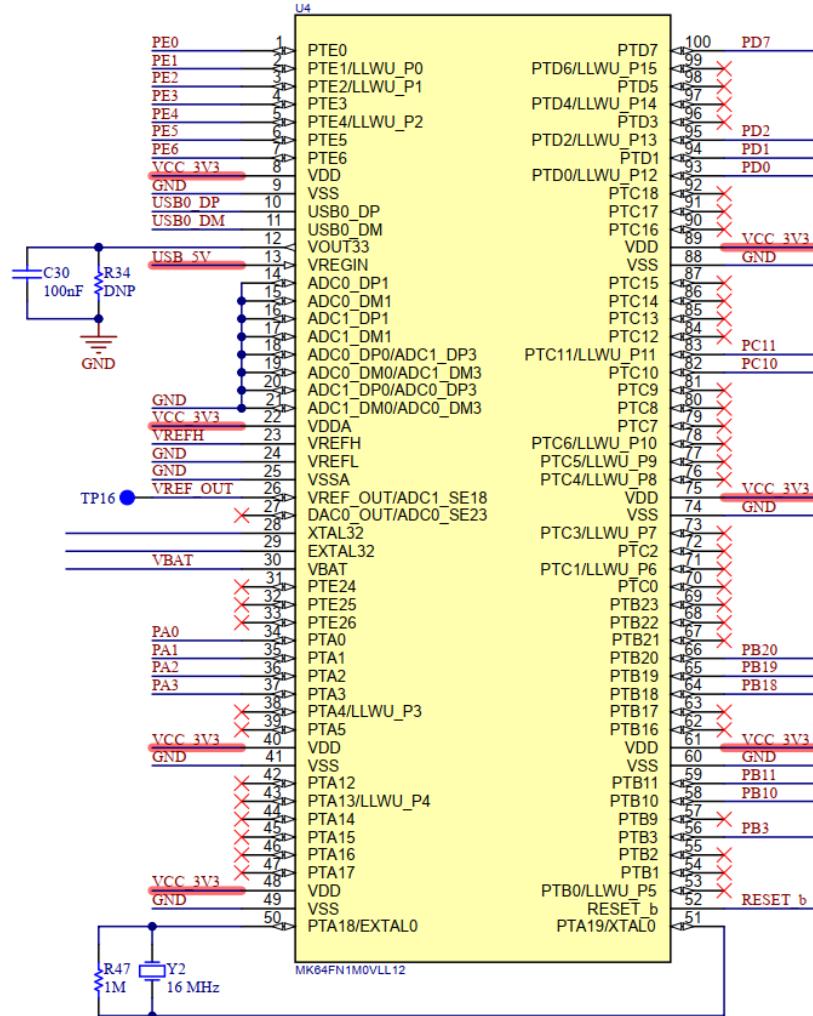


Figura 11.2: Circuito del microcontrolador.

En la figura 11.3 se puede ver el circuito empleado para la detección de corte de alimentación. Para esta funcionalidad se emplea el periférico interno de ADC (conversor analógico-digital) del MCU. Dado que la precisión requerida para la detección de tensión no es alta, se emplea una simple referencia de tensión analógica compuesta por un diodo zener. Únicamente la tensión de entrada del bus es requerida para detectar cortes de alimentación, sin embargo, dado que se cuentan con múltiples canales en el periférico ADC, todos los rieles de alimentación relevantes del circuito se conectan al mismo.

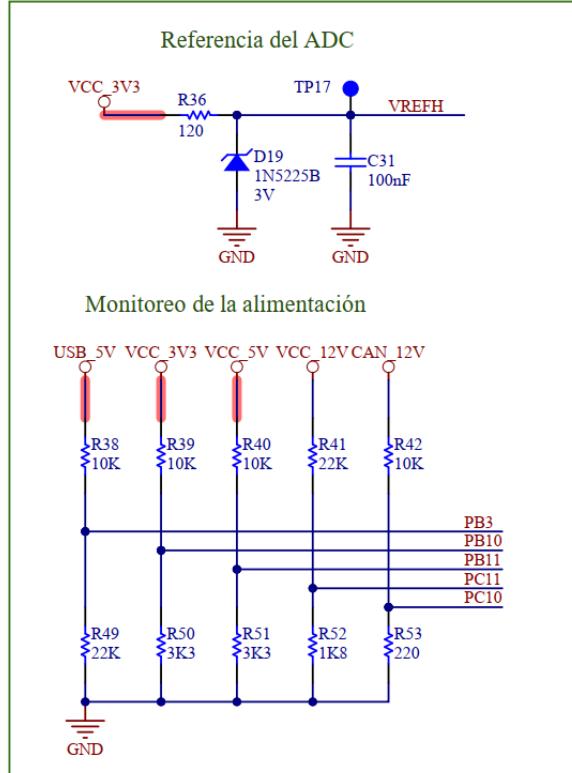


Figura 11.3: Circuito de detección de corte de energía.

#### 11.1.2.2 Módulo de reloj de tiempo real

La especificación PER-POW-3 indica que el dispositivo deberá mantener su fecha y hora interna por 1 año y que debe cumplirlo incluso sin alimentación externa. Para el primer punto (mantener la hora) se utiliza el periférico de RTC perteneciente al microcontrolador, que cuenta el paso del tiempo. Para el segundo punto (no perder la hora incluso sin alimentación externa) el microcontrolador cuenta con la opción de conectar una fuente de alimentación en un pin especial (VBAT) que solamente energiza al módulo RTC. De esta forma, no se pierde la cuenta del tiempo incluso cuando el microcontrolador está apagado. Para hacer uso de esto, el dispositivo incorpora una pequeña batería de tipo CR2032 conectada al pin VBAT.

La corriente promedio que consume el periférico de RTC cuando el MCU se encuentra apagado es de  $0.71 \mu A$ . Para una batería de tipo CR2032 de 3V y capacidad 210 mAh, la vida útil esperada de la batería es de 33.8 años. Por lo tanto, la especificación PER-POW-03 se cumple por diseño.

El periférico RTC utiliza una señal de clock de 32.768 kHz que es generada por un circuito de oscilador. Todo el circuito del oscilador es interno al microcontrolador con la excepción de un cristal que debe ser agregado externamente entre sus pines XTAL32 y EXTAL32.

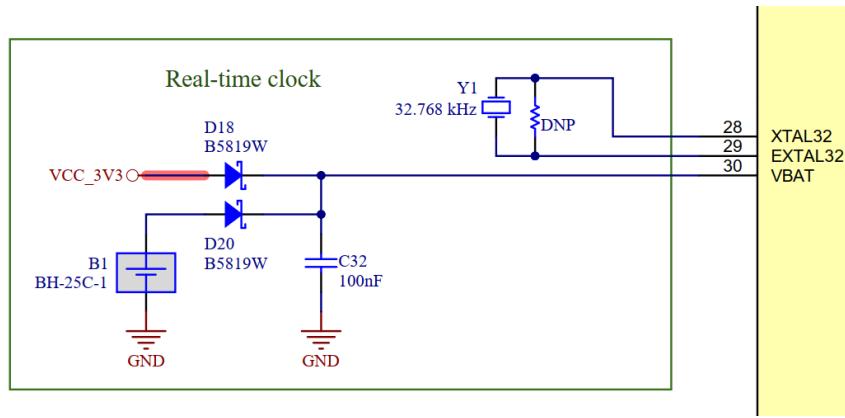


Figura 11.4: Circuito del módulo RTC.

#### 11.1.2.3 Módulo de comunicación con PC

El módulo USB tiene dos funcionalidades: primero, permite el acceso a la información de los mensajes CAN recopilada por el dispositivo, y segundo, funciona como fuente de alimentación del sistema (ver módulo de alimentación en esta sección).

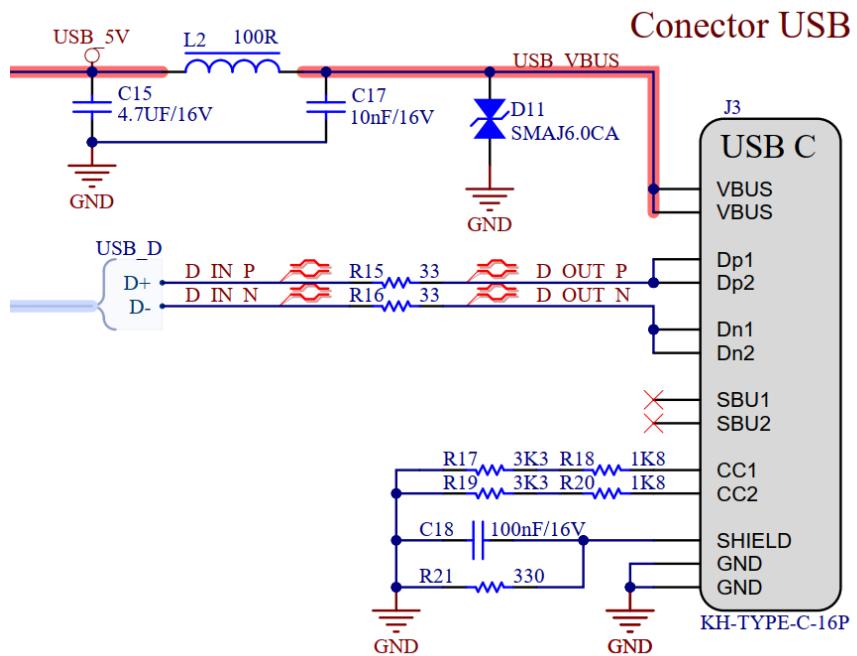


Figura 11.5: Circuito del módulo USB

Se emplea un conector USB Tipo-C y el protocolo de comunicación usado es USB 2.0, utilizando el periférico de USB 2.0 del MCU. En la figura 11.5 se puede ver el circuito de conexión USB. Las señales de datos D+ y D- del conector USB Tipo-C se conectan al microcontrolador a través de resistencias de protección, y el terminal VBUS del conector se conecta al módulo de alimentación a través de un filtro de ruido de alta frecuencia. El capacitor C18 y la resistencia R21 forman un filtro que conecta el chasis del conector con la tierra de referencia del circuito. Los resistores R17, R18, R19 y R20 se emplean para indicar al dispositivo “huésped” que se ha establecido una conexión con otro dispositivo y también puede indicar la orientación

de conexión del cable en caso de ser necesario. En la figura 11.6 se muestra un diagrama de la funcionalidad de detección mediante los pines CC1 y CC2 que implementan los conectores USB Tipo-C.

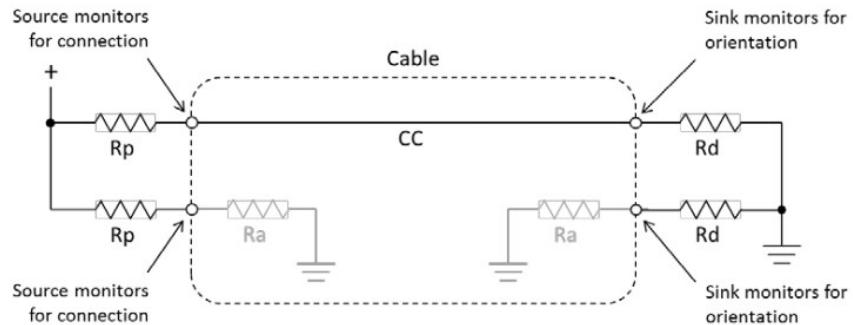


Figura 11.6: Circuito de detección de conexión en el conector USB Tipo-C.

#### 11.1.2.4 Módulo de almacenamiento

El módulo de almacenamiento se encarga de guardar los datos de los mensajes CAN recopilados por el dispositivo. Cuenta con un conector en donde se inserta una tarjeta SD.

El microcontrolador cuenta con un periférico para la lectura y escritura de tarjetas SD usando el protocolo SDHC. Se conectan directamente las señales DAT0, DAT1, DAT2, DAT3, DAT4, DAT5, CLK y COM del conector de la tarjeta SD a los puertos correspondientes del microcontrolador, siguiendo las recomendaciones de la nota de aplicación AN13031 de NXP Semiconductors [14]. Además, se alimenta a la tarjeta con 3.3 V a través del pin VDD del conector. En la figura 11.7 se puede ver el circuito diseñado.

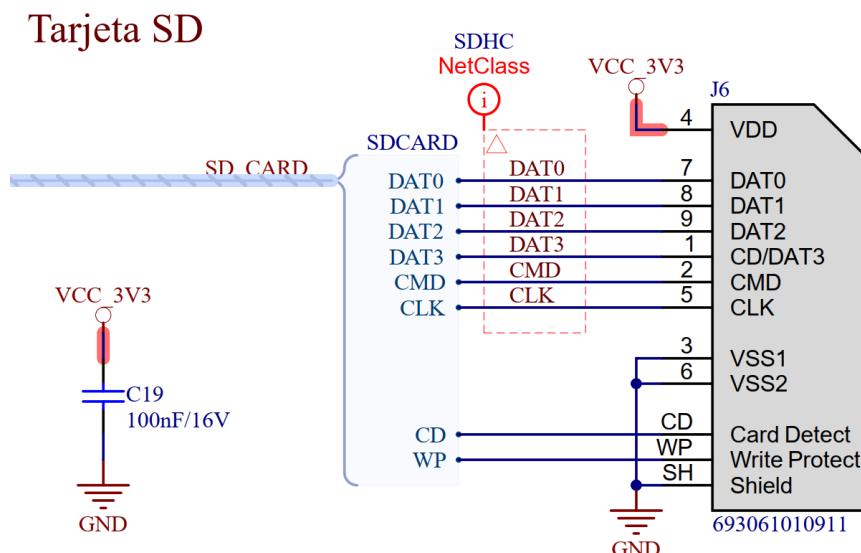


Figura 11.7: Circuito del módulo de almacenamiento.

### 11.1.2.5 Módulo CAN

El bus CAN se compone de dos conductores, CANH y CANL (por *high* y *low*, respectivamente), que utiliza un sistema de señalización diferencial.

El bus utiliza lógica AND cableada (*wired-AND*) con dos estados posibles: estado dominante, en donde se establece activamente una diferencia de tensión entre las señales CANH y CANL de 2 V y una tensión de modo común igual a 2.5 V, o estado recesivo, donde las señales no son establecidas activamente sino que su tensión se establece en forma pasiva por resistores. Un bit de datos 0 codifica un estado dominante, mientras que un bit de datos 1 codifica un estado recesivo, lo que admite una convención AND cableada, que otorga prioridad a los nodos con números de identificación más bajos en el bus.

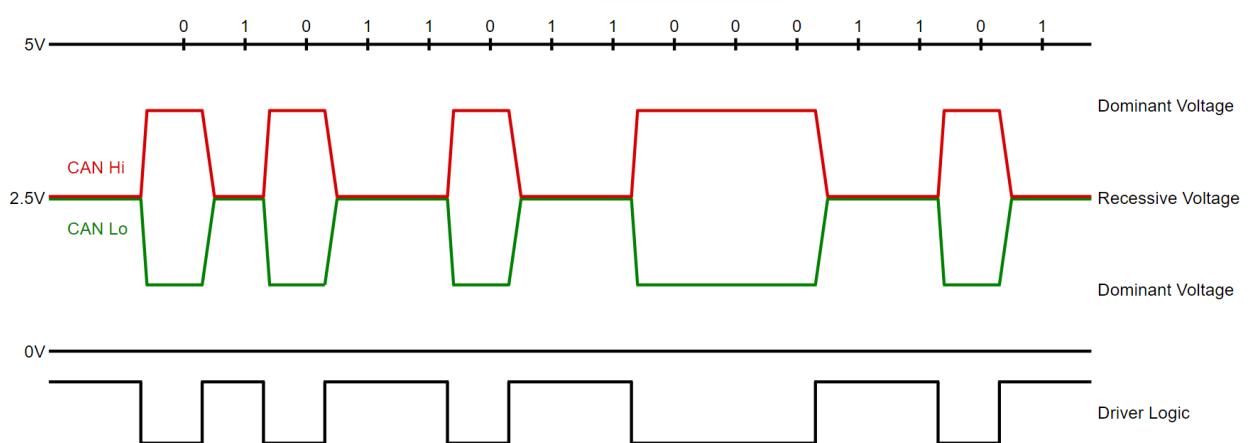


Figura 11.8: Niveles de tensión en el bus CAN.

En la figura 11.9 se puede ver el circuito esquemático con los dos conectores DE-9 a través de los cuales se establece una interfaz con el bus CAN del vehículo. Es necesario conectar en paralelo ambos conectores para permitir una conexión en cadena (*daisy-chain*) del bus CAN. El capacitor C14 y el resistor R14 conforman un circuito de filtrado que une el mallado del cable con la tierra de referencia del PCB. Adicionalmente, se incluye un diodo TVS (*transient voltage suppressor*) para proteger al resto de los componentes de picos de tensión que puedan ingresar a través del conector. Las señales CANH y CANL están protegidas ante picos de tensión debido a los diodos TVS internos al circuito integrado transceptor CAN. Para facilitar la depuración del prototipo fabricado, se incluyeron los conectores J4 y J5 de pines estándar con separación de una décima de pulgada.

El microcontrolador cuenta con un periférico integrado para la transmisión de datos a través de un bus CAN. El periférico implementa la capa de enlace de datos del protocolo CAN según el modelo ISO-OSI, mas no implementa la capa física. El periférico del microcontrolador implementa lógica LVTTL (del inglés *Low Voltage Transistor-Transistor Logic*) para transmitir la información del bus mediante dos señales *single-ended*, una de transmisión y otra de recepción. Por lo tanto, es imprescindible emplear un circuito transceptor de capa física para implementar la interfaz eléctrica con el bus CAN.

En la figura 11.10 se puede observar el circuito esquemático correspondiente al transceptor CAN. Los capacitores C21 y C22 cumplen la función de desacoplamiento. La impedancia característica del bus CAN está estandarizada por la norma ISO 11898-2:2016, y su valor es de  $120 \Omega$ . Los dispositivos CAN que se encuentren en las puntas de una cadena que conforma al bus, deben contar con una resistencia de terminación de  $120 \Omega$  que previene reflexiones indeseadas de las señales que viajan a través del bus. La resistencia R22 corresponde a la terminación de las señales diferenciales del bus CAN. Esta terminación puede ser deshabilitada cortando la conexión en el PCB que une ambos terminales del resistor R23, el cual por defecto no se popula en el PCB. En el caso de cortar dicha conexión, es posible recuperar la terminación soldando una resistencia de  $0 \Omega$  en la posición R23.

## Conecotor bus CAN

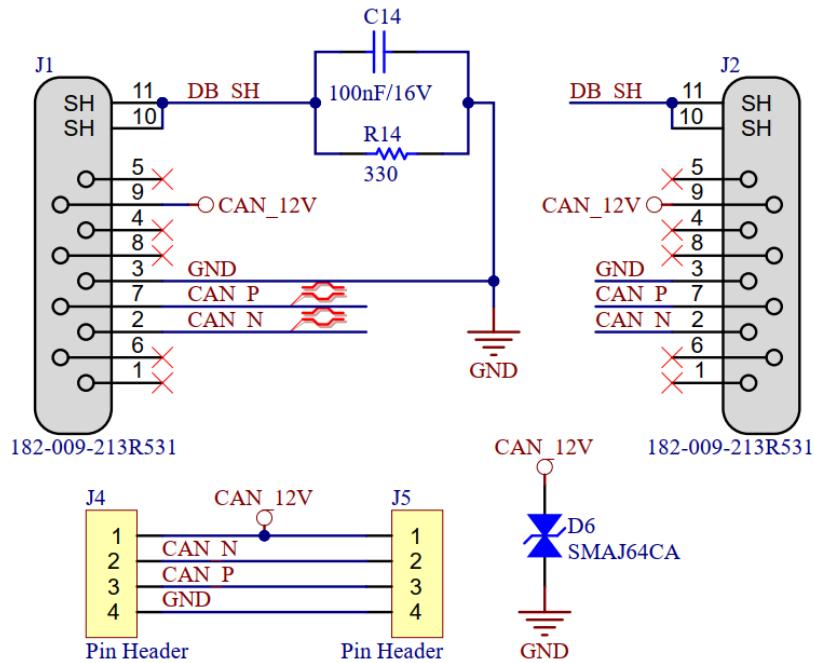


Figura 11.9: Circuito del conector CAN

## Transceiver CAN

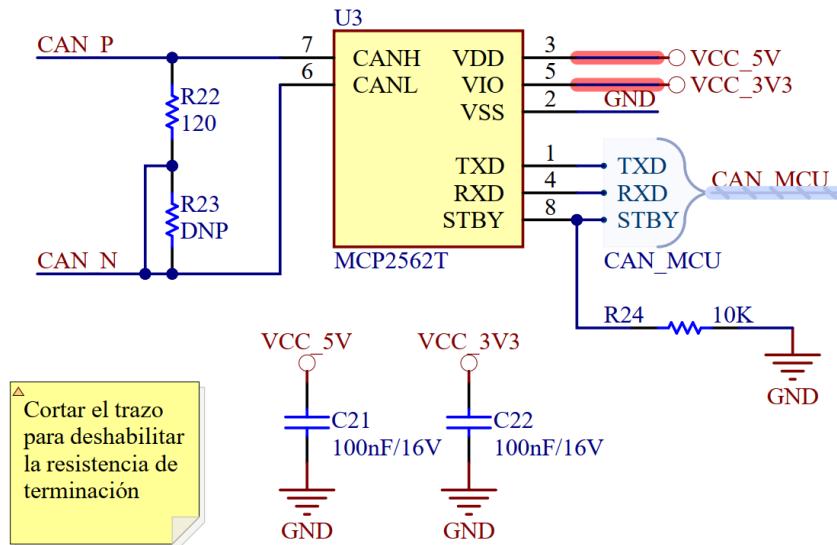
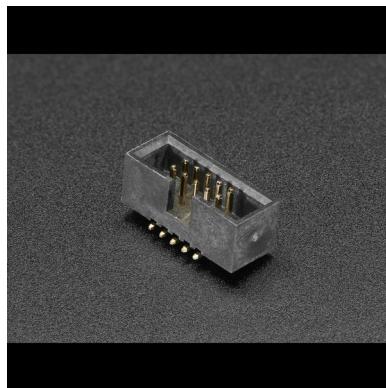


Figura 11.10: Circuito del modulo Tranceiver CAN

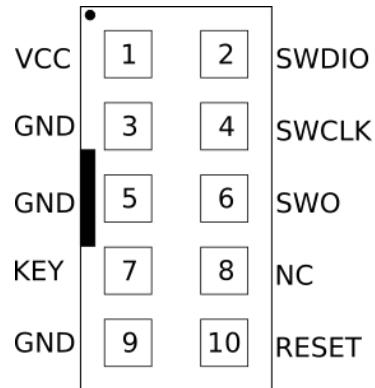
### 11.1.2.6 Módulo programador

Para programar el firmware del microcontrolador, se utiliza un dispositivo programador que utiliza el estándar de comunicación SWD<sup>1</sup>. Este estándar define los posibles conectores, las señales transmitidas por cada pin, y el protocolo de comunicación.

<sup>1</sup>Variación del estándar IEEE 1441.1, también conocido como JTAG, especifica para microprocesadores ARM.



(a)



(b)

Figura 11.11: Conector para programación según estándar SWD.

#### 11.1.2.7 Módulo de alimentación

El módulo de alimentación suministra energía al microcontrolador y al módulo de la tarjeta SD con 3.3 V y al módulo del transceiver CAN con 5V. Existen 3 posibles fuentes de alimentación que se utilizan para diferentes motivos y que cubren todos los casos de uso. Estas fuentes son<sup>2</sup>:

1. Batería del karting
2. Pin del conector USB
3. Pin del conector del programador

La fuente de la batería requiere un circuito de protección para que no se dañe ninguna parte del dispositivo en caso de que haya picos de tensión o tensión inversa como lo indicado en el requerimiento REQ-ALI-AUT-2. A su vez, en funcionamiento normal, la tensión puede ser de entre 10V a 30V (requerimiento REQ-ALI-AUT-1), por lo que se agrega un circuito que regule la tensión hasta 5V. El circuito de protección y el de regulación de tensión se ilustran en las figuras 11.12 y 11.13.

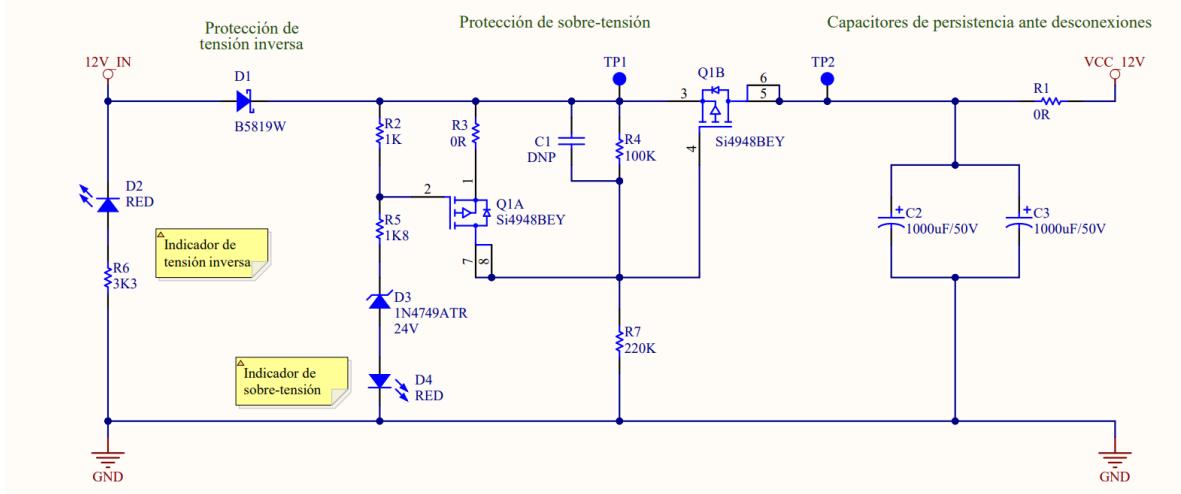


Figura 11.12: Circuito de protección de tensión para la batería.

<sup>2</sup>No se incluye la fuente del RTC ya que esta independientemente a las otras fuentes y es explicada con el módulo RTC.

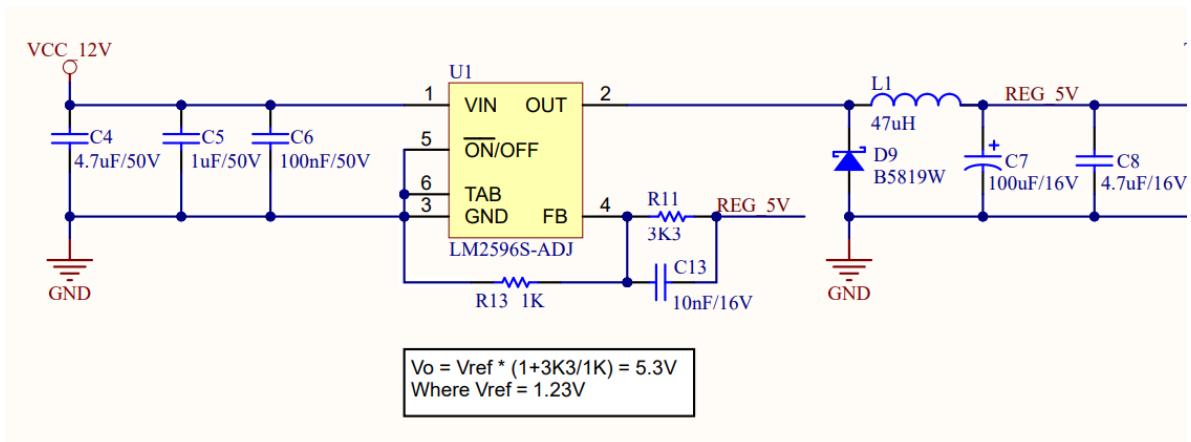


Figura 11.13: Circuito de regulación de tensión para la batería.

A las fuentes de provenientes del conector USB y del conector del programador se les agrega un filtro pasabajos de segundo orden para evitar picos de corriente.

Para que evitar cortocircuitos si hay múltiples fuentes conectadas en simultáneo, se usan diodos Schottky en los puntos en donde se unen dos fuentes.

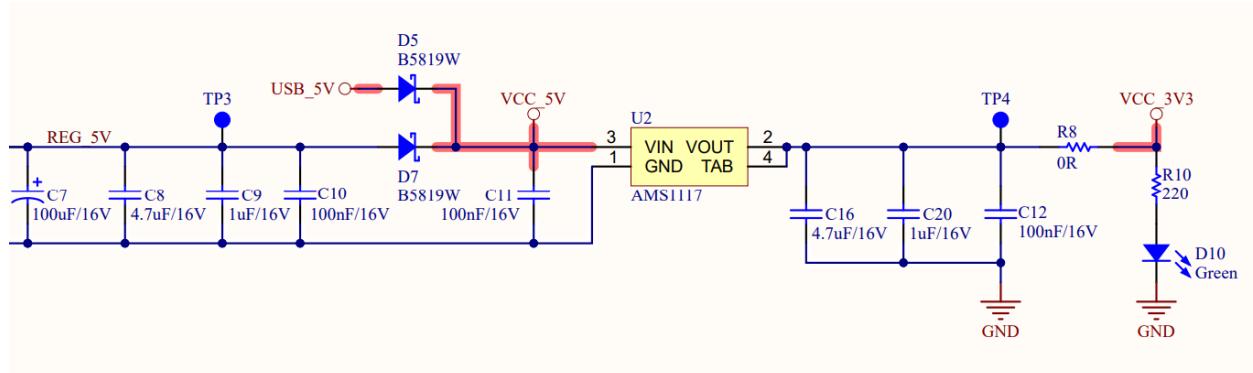


Figura 11.14: Circuito del módulo de alimentación

### 11.1.3. Selección y cálculo de elementos circuitales de módulos

#### 11.1.3.1 Circuito de protección de la batería

El circuito de protección presentado cuenta con 4 posibles casos dependiendo de la tensión de alimentación.

- Baja tensión
- Modo de funcionamiento normal según requerimientos
- Protección por sobretensión
- Protección por tensión negativa

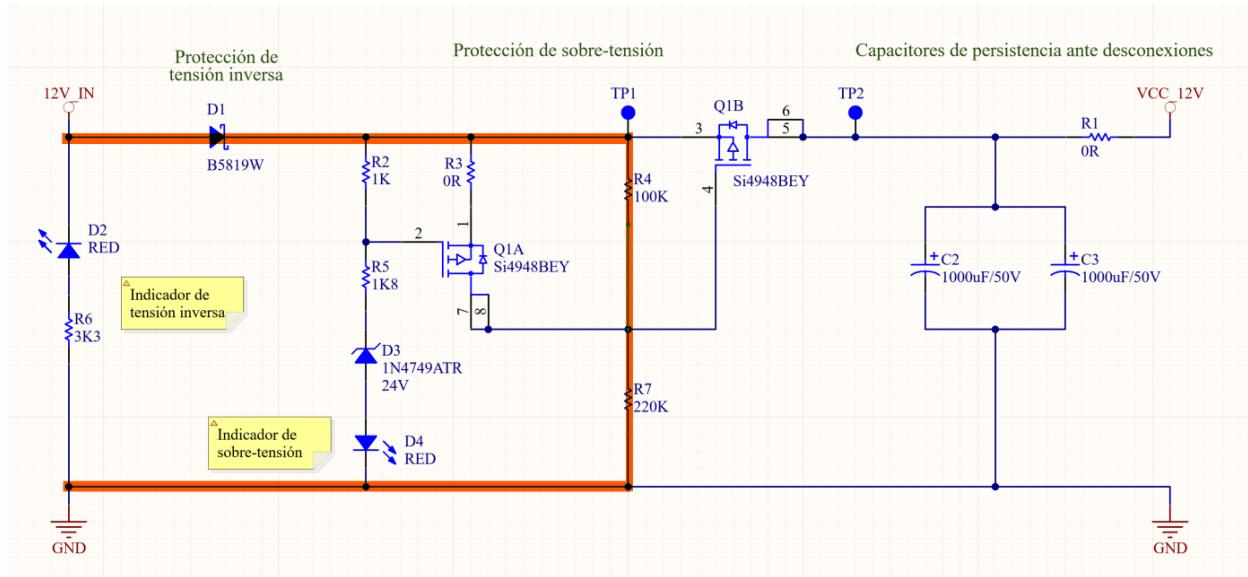


Figura 11.15: Circuito de protección en modo baja tensión

Para el encendido del circuito de protección se calcularon las resistencias R4 y R7 considerando que el transistor Si4948BEY tiene una V<sub>SGth</sub> de entre 1V y 3V. Con la siguiente ecuación se obtiene la tensión de encendido del circuito de protección.

$$V_{ONth} = V_{SGth} \cdot \frac{R_4 + R_7}{R_4} \quad (11.1)$$

Utilizando los valores  $R_4 = 100\text{k}\Omega$  y  $R_7 = 220\text{k}\Omega$ , se obtuvo que la tensión de encendido del circuito es entre 3.2V y 9.6V.

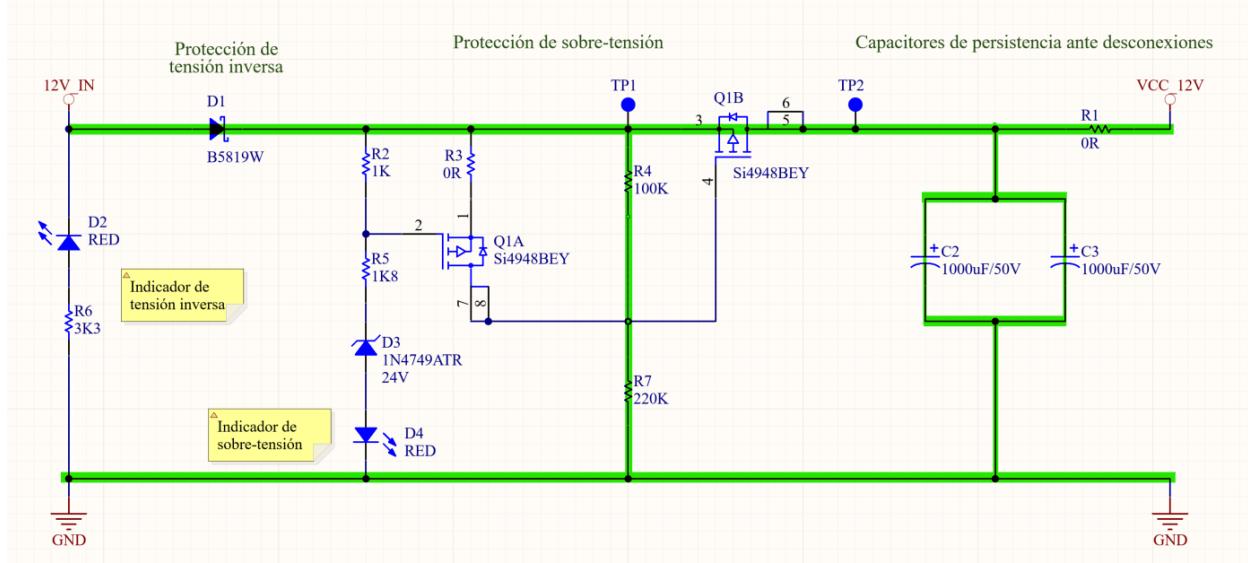


Figura 11.16: Circuito de protección en funcionamiento normal

Para evaluar el apagado del circuito por sobretension, se evalúa el transistor Q1A con las resistencias R2 y R5 y el diodo zener D3 de 24V

$$V_{OFFth} = V_{SGth} \cdot \frac{R_2 + R_5}{R_2} + V_z + V_D \quad (11.2)$$

Con valores  $R_2 = 1\text{k}\Omega$ ,  $R_5 = 1.8\text{k}\Omega$ ,  $V_z = 24\text{V}$  y  $V_D = 1.7\text{V}$ , se obtuvo que las tensiones de apagado del circuito es entre 28.5V y 34.1V.

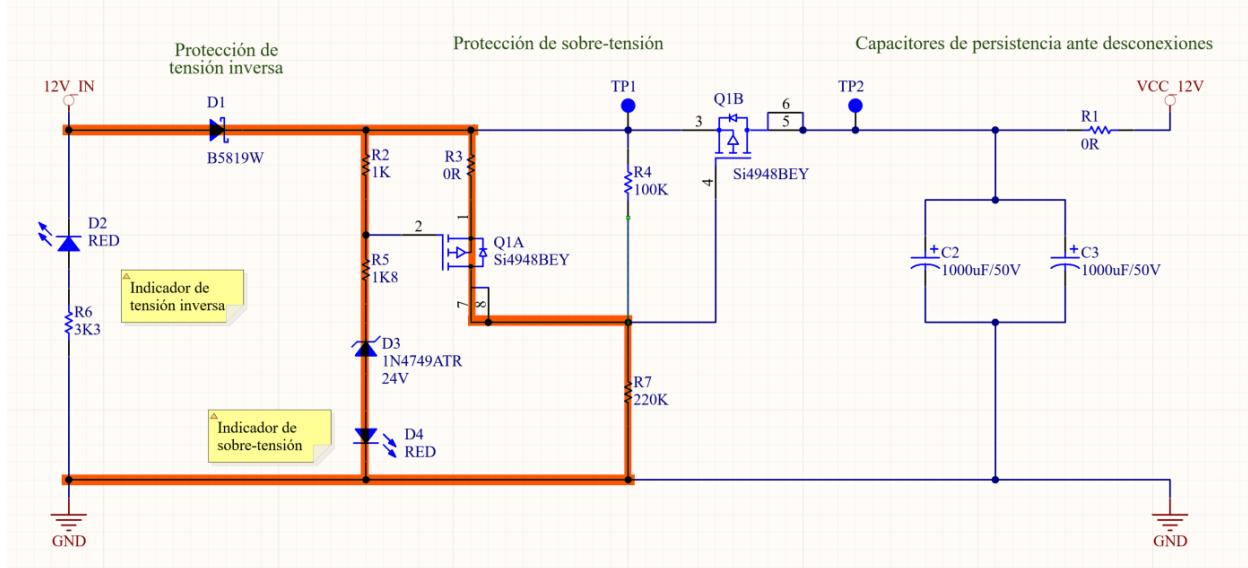


Figura 11.17: Circuito de protección en modo sobretensión

Se valido que las tensiones de gate-source de los transistores no superen el máximo permitido por el fabricante, que es de 20V.

$$V_{gsQ1A} \leq (V_{MAX} - V_z - V_D) \cdot \frac{R_2}{(R_2 + R_5)} \quad (11.3)$$

$$V_{gsQ1B} = V_{MAX} \cdot \frac{R4}{R4 + R7} \quad (11.4)$$

Tomando que la tensión máxima de entrada es de 50V, se obtuvo que las tensiones máximas de gate-source de los transistores son 8.7V y 15.6V para los transistores Q1A y Q1B respectivamente, siendo ambos valores inferiores a los límites determinados por el fabricante.

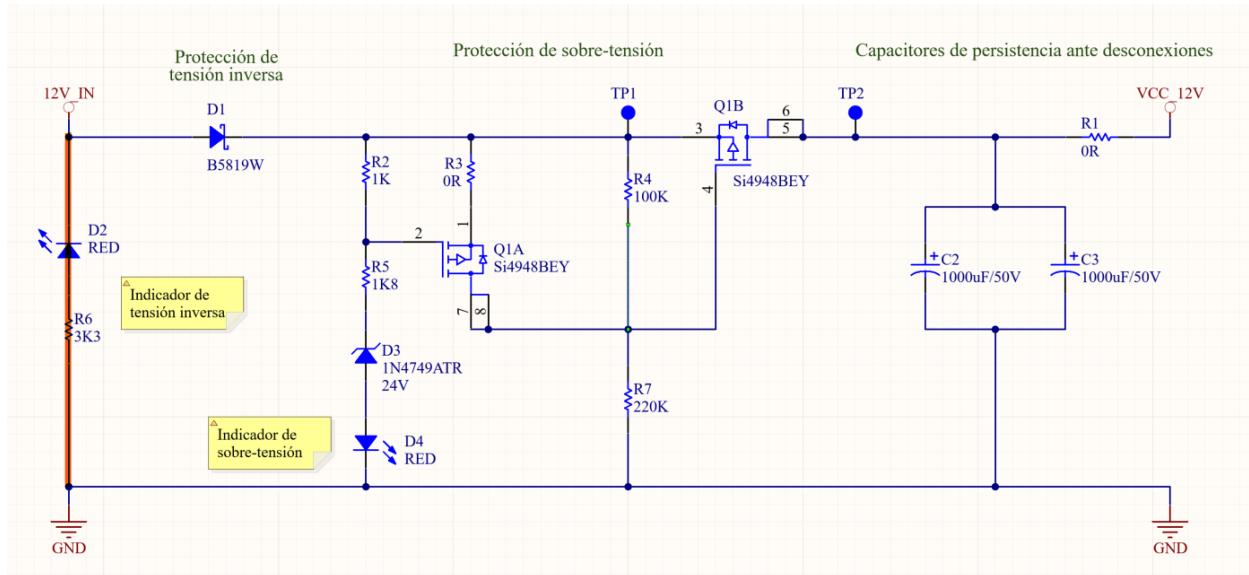


Figura 11.18: Circuito de protección en modo tensión inversa

### 11.1.3.2 Circuito de regulación de la batería

Para el circuito de regulación de tensión 12V a 5V, se utiliza un integrado Step-Down LM2596S-ADJ.

Para seleccionar las resistencias de realimentación para un regulador de tensión step down ajustable como el LM2596, se utiliza la siguiente ecuación, logrando una tensión de salida ligeramente superior a 5V.

$$V_o = V_{ref} \cdot \left(1 + \frac{R_{11}}{R_{13}}\right) = 1,23V \cdot \left(1 + \frac{3,3k\Omega}{1k\Omega}\right) = 5,3V \quad (11.5)$$

Para el dimensionamiento del inductor, se considera que la frecuencia de conmutación del LM2596S-ADJ es de 150kHz.

$$L \geq \frac{Vd - V_o}{0,25 * I_o} * \frac{V_o}{Vd} * \frac{1}{f_s} = 39\mu H \quad (11.6)$$

Los capacitores en la salida se seleccionaron para que la variación de tensión no sea pronunciada debido a la corriente de salida.

$$\Delta V_o = \frac{\Delta i_L}{8f_s C} = 1,6mV \quad (11.7)$$

### 11.1.3.3 Filtros de la fuente de tensión de USB

Para reducir interferencias electromagnéticas (EMI) y mejorar la integridad de la señal una práctica estándar es la adición de un filtro con topología “pi” conformado por un inductor (ferrite) y dos capacitores [8]. La línea de VBUS del conector USB, que suministra energía a los dispositivos conectados, puede ser propensa a ruido electromagnético, lo que podría afectar negativamente a la transmisión de datos y la estabilidad del sistema. El ferrite actúa como un filtro de alta frecuencia al absorber el ruido y evitar que se propague a lo largo de la línea, mejorando así la calidad de la señal y reduciendo la posibilidad de interferencias que podrían afectar el rendimiento del dispositivo USB y otros componentes electrónicos circundantes.

El diseño del filtro implementado en el PCB se realizó de acuerdo a la nota de aplicación AN-146 de FTDI [9]. En la figura 11.5 se puede ver el circuito empleado.

#### 11.1.4. Plan de pruebas de módulos

##### 11.1.4.1 Módulo de la unidad de procesamiento y módulo programador

1. Configurar la placa de desarrollo FRDM-K64F para programación de un microcontrolador externo, removiendo el *jumper* J11.
2. Conectar el conector SWD la placa de desarrollo FRDM-K64F al conector SWD de la placa CANDLE.
3. Cargar el programa de prueba que configura los pines GPIO de los indicadores LED para encenderse repetidamente.
4. Verificar que los indicadores LED se comportan de la forma esperada, y que el intervalo de tiempo en que se mantienen encendidos es el apropiado.

##### 11.1.4.2 Módulo de reloj de tiempo real

1. Configurar la placa de desarrollo FRDM-K64F para programación de un microcontrolador externo, removiendo el *jumper* J11.
2. Conectar el conector SWD la placa de desarrollo FRDM-K64F al conector SWD de la placa CANDLE.
3. Cargar el programa de prueba que configura el módulo RTC con la hora actual.
4. Des-energizar la placa CANDLE y esperar un mínimo de 5 minutos.
5. Conectar el conector SWD la placa de desarrollo FRDM-K64F al conector SWD de la placa CANDLE.
6. Cargar el programa de prueba que lee e imprime en pantalla la hora actual reportada por el periférico RTC.
7. Verificar que el tiempo reportado coincide con el intervalo transcurrido desde la configuración inicial de la hora actual.

##### 11.1.4.3 Módulo de comunicación con PC

1. Configurar la placa de desarrollo FRDM-K64F para programación de un microcontrolador externo, removiendo el *jumper* J11.
2. Conectar el conector SWD la placa de desarrollo FRDM-K64F al conector SWD de la placa CANDLE.
3. Cargar el programa ejemplo del kit de desarrollo de software que emplea el periférico USB como un dispositivo de almacenamiento masivo.
4. Desconectar el cable de programación de la placa CANDLE.
5. Conectar un cable USB entre la placa CANDLE y una computadora.
6. Verificar que la computadora monta el dispositivo conectado como un disco de almacenamiento.

#### **11.1.4.4 Módulo de almacenamiento**

1. Configurar la placa de desarrollo FRDM-K64F para programación de un microcontrolador externo, removiendo el *jumper J11*.
2. Conectar el conector SWD la placa de desarrollo FRDM-K64F al conector SWD de la placa CANDLE.
3. Cargar el programa ejemplo del kit de desarrollo de software que emplea el periférico SD y le escribe un archivo de texto vacío a la tarjeta.
4. Desconectar el cable de programación de la placa CANDLE.
5. Remover la tarjeta SD del CANDLE e insertarla en una computadora.
6. Verificar que la tarjeta tiene un archivo de texto vacío.

#### **11.1.4.5 Módulo CAN**

1. Alimentar VDD con 5 V, VIO con 3.3 V y aplicar 0 V a las señales de STDBY y GND.
2. Excitar la entrada RXD con una señal cuadrada de 3.3 V y 125 kHz.
3. Verificar que cuando RXD sea baja, la tensión de salida de CANH y CANL 2.5 V.
4. Verificar que cuando RXD sea alta, las tensiones de salida de CANH y CANL sean  $4 \pm 0.5$  V y  $1 \pm 0.5$  V respectivamente.

#### **11.1.4.6 Módulo de alimentación**

##### **■ Sistema de protección**

1. Alimentar la entrada con -20 V.
2. Verificar que la tensión a la salida de la etapa de protección sea 0 V.
3. Alimentar la entrada con una rampa de 0V a 50V.
4. Verificar que la salida de la etapa de protección tiene menos de 1 V de diferencia con la entrada cuando la entrada esté entre 10 V y 30 V.
5. Verificar que la tensión a la salida de la etapa de protección sea de 0 V cuando la tensión de entrada supere 30 V.

##### **■ Sistema de regulación**

1. Alimentar la entrada con 12V.
2. Verificar que en el TP7 la tensión es 5V.
3. Verificar que en el TP9 la tensión es 3,3V.
4. Repetir alimentando la entrada con 10V, 20V y 30V.

##### **■ Capacitores de mantenimiento**

1. Preparar un osciloscopio para medir las siguientes señales cuando se apague la fuente:
  - Entrada de alimentación de bus CAN
  - Tensión en los capacitores de mantenimiento (TP2)

- Alimentación del microcontrolador (TP9)
  - Tensión en el LED que se enciende durante la duración de la tarea shutdownTask y se apaga al finalizar.
2. Alimentar la entrada con 12V.
  3. Enviar mensajes CAN.
  4. Apagar la fuente, y durante el apagado capturar las señales con el osciloscopio.
  5. Verificar que la shutdownTask finalice antes de que la alimentación del microcontrolador comience a apagarse.

## 11.2. Software

Para el desarrollo del software del producto, se utilizó el SDK (*Software Development Kit*) de la empresa NXP específico para el microcontrolador del diseño. Este conjunto de herramientas suministra los controladores necesarios para los periféricos integrados, permitiendo que el enfoque del desarrollo se centre en la implementación de la aplicación, que está basada en FreeRTOS para un eficiente manejo multitarea.

El código cumple dos funciones principales: almacenar los datos capturados del bus CAN en una tarjeta SD (Modo *Logger*), y permitir el acceso a estos datos desde cualquier PC a través de un conector USB (Modo *MSD*, o *Mass Storage Device*).

### 11.2.1. Diagramas de bloques y estados

La Figura 11.19 muestra la arquitectura del código. La capa de controladores de bajo nivel es la tomada del SDK, con modificaciones como aumentar la precisión del reloj de tiempo real a milisegundos. Las tareas principales se ejecutan de forma concurrente haciendo uso de los controladores. La aplicación principal se encarga de instanciar y conectar a las tareas principales.

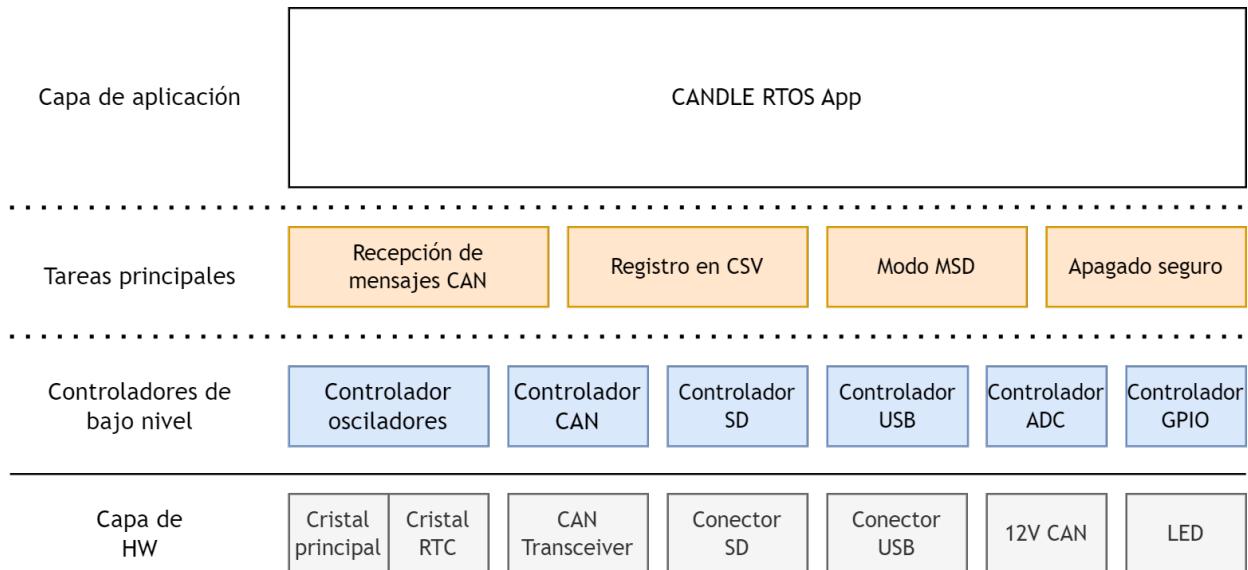


Figura 11.19: Arquitectura del software

La Figura 11.20 ilustra un diagrama de estados del software. Al energizarse el microcontrolador, se verifica si su alimentación proviene del bus CAN o del conector USB. En caso de ser desde el USB, entra en modo *Mass Storage Device*. Luego de concluir su inicialización, el dispositivo es detectado por la PC a la cual se encuentre conectado y permite acceder a los datos recolectados en la tarjeta SD.

En caso de ser alimentado desde el bus CAN, pasa al modo *Logger*. En este estado, se mantiene recibiendo continuamente mensajes CAN y guardándolos en un buffer. Al llenarse, se dispara una escritura a la tarjeta SD. Esto permite disminuir la cantidad de llamados a la rutina de escritura, la cual resulta costosa en recursos de ejecución. Si un corte de alimentación es detectado, se procede con un apagado seguro que deshabilita las tareas que se encuentran activas. Luego, se realiza una escritura a la tarjeta SD con los datos que estén pendientes y un mensaje indicando que se terminó la sesión de *logging* con un apagado seguro.

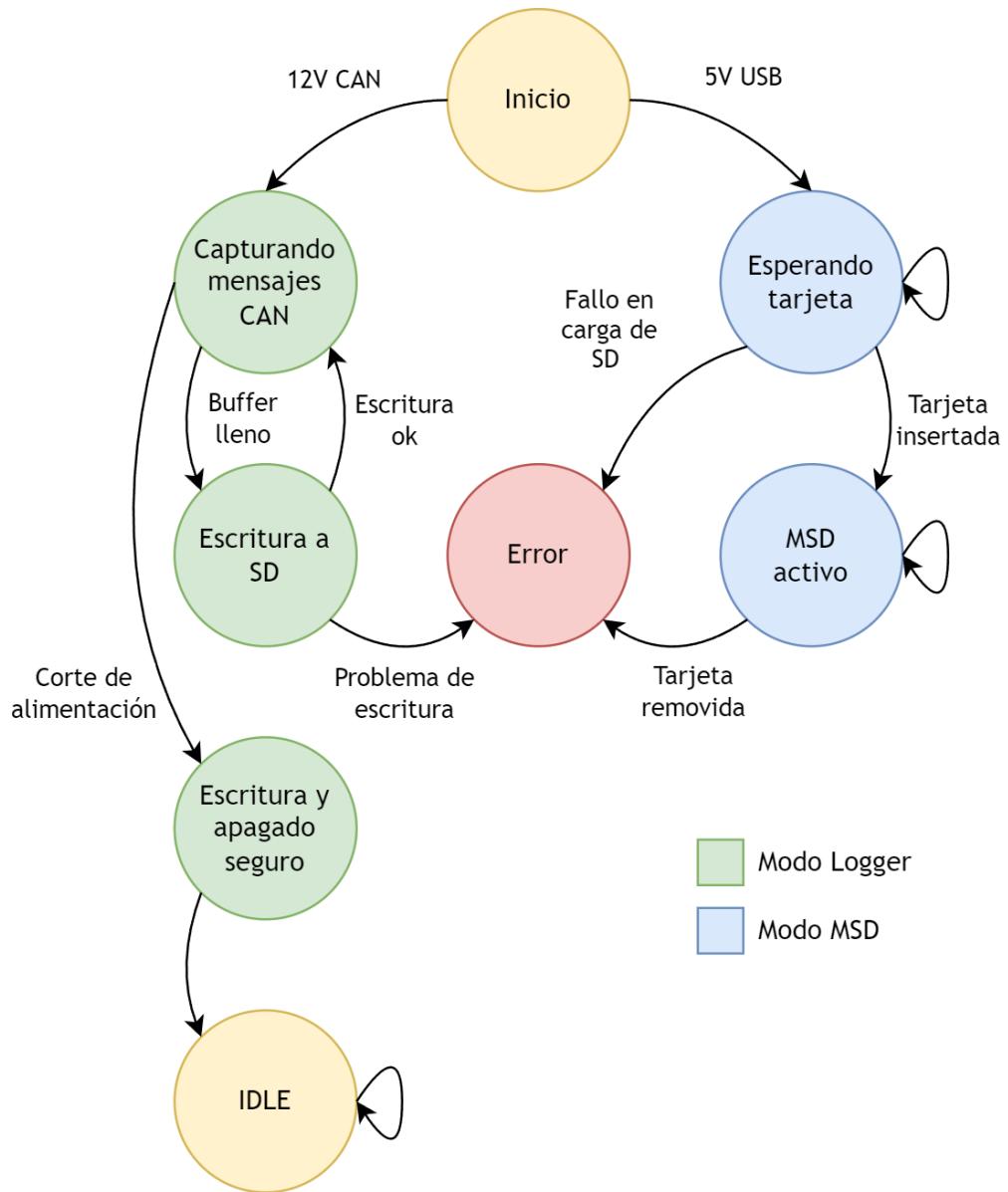


Figura 11.20: Diagrama de estados

### 11.2.2. Descripción de subrutinas

Las principales subrutinas del programa son:

- main
- DriveTask
- FlexCanTask
- CardDetectTask
- FileAccessTask
- ADC16\_IRQ\_HANDLER\_FUNC
- ShutdownTask

#### **11.2.2.1 main**

La subrutina principal del programa. Está diseñada para inicializar y configurar el hardware y el software del producto. Inicialmente, configura los pines y los relojes de la placa. Luego, determina el modo de operación (MSD o Logging) sensando el origen de la alimentación a través de un pin GPIO.

Si la placa está en modo de registro (“LOGGER\_MODE”), inicializa el Reloj de Tiempo Real (RTC), el Convertidor Analógico a Digital (ADC), el controlador FlexCAN y el sistema de registro de datos, y después crea tareas específicas para manejar la detección de tarjetas SD, la comunicación FlexCAN, el acceso a archivos y el proceso de apagado. Si está en modo de MSD (‘DRIVE\_MODE’), inicializa el RTC y la configuración de MSD y crea una tarea dedicada a esta modalidad.

Después de configurar el entorno según el modo seleccionado, inicia el planificador de tareas de FreeRTOS, permitiendo que el sistema operativo de tiempo real (RTOS) maneje la ejecución de las tareas definidas. Finalmente, la función entra en un bucle infinito, manteniendo activo el programa. El bucle incluye una instrucción NOP (no operación) para permitir el avance paso a paso durante la depuración.

#### **11.2.2.2 DriveTask**

La función DriveTask se encarga de gestionar el funcionamiento de la placa cuando se comporta como un dispositivo *Mass Storage Device*. Inicialmente, la tarea activa dos LED para indicar que la tarjeta SD aún no ha sido inicializada. Posteriormente, se lleva a cabo la inicialización de la aplicación del dispositivo USB para preparar el manejo de las comunicaciones.

Después de la inicialización, la función verifica si el controlador del dispositivo USB está correctamente inicializado. Si lo está, intenta crear una nueva tarea dedicada exclusivamente al dispositivo USB. Esta nueva tarea se encargará de gestionar todas las operaciones relacionadas con el dispositivo, como pueden ser la comunicación de datos y la detección de eventos. La creación de la tarea se realiza con una prioridad específica y un tamaño de pila definido. Si la creación de esta tarea falla, la función informa del error y termina su ejecución. Si la tarea ha sido creada con éxito, se enciende un LED para indicar que la unidad USB está operativa.

Finalmente, independientemente del resultado de las operaciones anteriores, la tarea se suspende a sí misma indefinidamente. Esto significa que la tarea ha finalizado su trabajo principal y se detiene, lo cual ayuda a reducir el consumo de recursos del sistema, como el tiempo de CPU, permitiendo que otras tareas puedan ejecutarse.

#### **11.2.2.3 FlexCanTask**

La función FlexCanTask es una tarea diseñada para la recepción de mensajes a través del protocolo CAN en sistemas embebidos. La tarea opera en un bucle infinito, asegurando una continua espera y procesamiento de mensajes CAN.

Inicialmente, la tarea intenta iniciar la recepción de datos a través de una cola FIFO (First In, First Out) dedicada a mensajes CAN. Esta función se encarga de configurar la recepción de mensajes de forma no bloqueante, permitiendo que el sistema operativo RTOS regule las esperas.

La tarea se detiene hasta que se complete la recepción de un mensaje CAN, utilizando un semáforo para sincronizar este evento. La espera se realiza de manera indefinida hasta que el semáforo se libere, indicando que un mensaje ha sido recibido exitosamente.

Una vez recibido el mensaje, la tarea procede a construir una estructura de datos con la información recibida. Esta estructura incluye la marca de tiempo actual (obtenida mediante el RTC), el identificador del mensaje, su longitud, y los datos distribuidos en un arreglo de bytes.

Después de construir la estructura, la tarea la envía a una cola. Esto permite que otros componentes o tareas del sistema que estén esperando mensajes CAN puedan procesar la información recibida.

#### 11.2.2.4 CardDetectTask

La función CardDetectTask está diseñada para gestionar la detección y la inicialización de una tarjeta SD en la placa CANDLE. La tarea comienza creando un semáforo binario que se utiliza para controlar la detección de la tarjeta SD. Luego configura la tarjeta SD y registra una función de callback específica para manejar la detección de la tarjeta.

La inicialización del host de la tarjeta SD se verifica y, si es exitosa, la tarea entra en un bucle infinito donde espera la señalización del semáforo para proceder. Esto se hace para manejar de forma asíncrona la inserción o extracción de la tarjeta SD. Cuando se detecta una tarjeta, la tarea verifica si el estado de inserción ha cambiado y, de ser así, se reinicializa la tarjeta apagándola y encendiéndola de nuevo.

Si la tarjeta está insertada correctamente, se intenta crear un sistema de archivos en ella y prepara la carpeta que contendrá los registros de la sesión asignando como nombre la fecha en formato YY\_MM\_DD. Si este proceso es exitoso, se actualizan los LED para reflejar el estado operativo del sistema de archivos. Finalmente, si el sistema de archivos está listo, se notifica a la tarea de acceso a archivos para que pueda comenzar a operar con la tarjeta SD. La tarea se suspende indefinidamente al final de la ejecución para no consumir más recursos del sistema operativo en tiempo real.

#### 11.2.2.5 FileAccessTask

La función FileAccessTask es una tarea diseñada para gestionar el acceso a archivos, específicamente para la creación y escritura de archivos de log de mensajes CAN. Esta tarea comienza esperando una notificación para iniciar su ejecución, lo que permite sincronizarla con la tarea CardDetectTask.

Una vez recibida la notificación, la tarea inicializa un archivo de log. Su nombre es el horario de creación en formato HH\_MM\_SS, facilitando la organización y búsqueda de los registros. Si el archivo especificado no existe, intenta crearlo; si hay algún error en este proceso, se indicará mediante el encendido de LEDs y se terminará la ejecución de la tarea.

Tras abrir o crear el archivo correctamente, la tarea escribe una cabecera en el archivo, la cual incluye etiquetas para los datos que se registrarán, como fecha, hora, ID del mensaje CAN, longitud del mensaje y los datos del mensaje. Esta cabecera permite entender el formato de los datos registrados en el archivo.

La tarea continúa en un bucle infinito, esperando recibir mensajes CAN a través de una cola. Al recibir un mensaje, este se añade a un buffer. Cuando el buffer alcanza un umbral definido, su contenido se escribe en el archivo de log. Este mecanismo de buffering mejora la eficiencia de la escritura en el archivo, reduciendo el número de operaciones de escritura y, por ende, el desgaste del medio de almacenamiento y el tiempo de ejecución de la tarea.

Si se produce un error durante la escritura en el archivo o si el buffer alcanza su capacidad máxima, se indicará un error mediante el encendido de LEDs, se cerrará el archivo de log para preservar los datos ya escritos, y se terminará la ejecución de la tarea. Finalmente, si la tarea completa su ciclo sin errores o si

debe terminar su ejecución por cualquier motivo, se suspenderá a sí misma indefinidamente, liberando el control al sistema operativo para que ejecute otras tareas.

Este enfoque asegura que los datos importantes se registren de manera organizada y eficiente, facilitando su posterior recuperación y análisis. La suspensión de la tarea al final de su ejecución ayuda a gestionar los recursos del sistema de manera efectiva, asegurando que solo se consuman cuando sea estrictamente necesario.

#### **11.2.2.6 ADC16\_IRQ\_HANDLER\_FUNC**

La función ADC16\_IRQ\_HANDLER\_FUNC actúa como un manejador de interrupciones para eventos específicos relacionados con la medición de señales analógicas en un sistema embebido. Su propósito principal es responder a señales que cruzan ciertos umbrales de interés, como pueden ser niveles de voltaje críticos. Al detectar una caída en la alimentación de la placa proveniente del bus CAN, la función inicia un proceso de cierre o transición segura para el sistema, asegurándose de que las operaciones en curso se detengan de manera controlada.

El proceso comienza desactivando las interrupciones adicionales para prevenir que un ciclo de llamados a la función interfieran con el proceso de cierre. Luego, procede a deshabilitar el sistema de medición de señales analógicas para evitar lecturas adicionales y mayor consumo de recursos.

Para gestionar el cierre del sistema, la función señala a la tarea ShutdownTask que debe prepararse para el cierre. Si al enviar esta señal se identifica que tareas de mayor prioridad necesitan ser atendidas inmediatamente, la función facilita un cambio de contexto para asegurar que estas tareas se ejecuten sin demora. Esto es crítico en sistemas de tiempo real donde la respuesta oportuna a eventos es fundamental.

#### **11.2.2.7 ShutdownTask**

La función ShutdownTask está diseñada para gestionar el proceso de apagado de la placa CANDLE. Esta tarea opera continuamente a la espera, lo que le permite estar siempre lista para ejecutar la secuencia de apagado cuando se le indique.

La tarea entra en un bucle infinito donde su principal acción es esperar a ser disparada por la detección de pérdida de alimentación. Una vez que esto ocurre, la tarea ejecuta las acciones necesarias para el apagado. Esto incluye encender un LED para indicar visualmente que el proceso de apagado ha comenzado, como también llamadas a funciones que detienen los servicios y escriben los datos pendientes en la SD antes de que el sistema pierda energía por completo.

### **11.2.3. Listados comentados del código**

En los listados que se muestran a continuación aparecen declaraciones condicionales de compilación atadas a la placa en uso. Esto permite utilizar distintos bloques de código para la placa de desarrollo de NXP (FRDM) y la placa de prototipo (CANDLE), facilitando la depuración.

```

1 int main(void) {
2     /* Inicializar el hardware de la placa. */
3     BOARD_InitBootPins();
4     BOARD_InitBootClocks();
5
6     op_mode_t op_mode; // Definir variable para el modo de operación.
7
8     // Leer el estado del pin y establecer el modo de operación correspondiente.
9     if (GPIO_PinRead(BOARD_MODE_GPIO, BOARD_MODE_PIN) == BOARD_MODE_DRIVE)
10    {
11        op_mode = DRIVE_MODE; // Modo de Mass Storage Device.
12    }
13    else
14    {
15        op_mode = LOGGER_MODE; // Modo de registro.
16    }
17
18    SYSMPU_Enable(SYSMPU, false);
19
20    // Configurar las tareas y el hardware según el modo de operación.
21    if (op_mode == LOGGER_MODE)
22    {
23        Init_RTC(false); // Inicializar el RTC.
24        Init_ADC(&shutdownTaskHandle); // Inicializar el ADC.
25        Init_FlexCAN(&flexCanTaskHandle); // Inicializar FlexCAN.
26        Init_Logging(&fileAccessTaskHandle); // Inicializar el sistema de registro.
27
28        // Crear tareas para el modo de registro.
29        xTaskCreate(FileAccessTask, (char const *)"FileAccessTask", ACCESSFILE_TASK_STACK_SIZE,\n
30                    NULL, ACCESSFILE_TASK_PRIORITY, &fileAccessTaskHandle);
31        xTaskCreate(CardDetectTask, (char const *)"CardDetectTask", CARDDETECT_TASK_STACK_SIZE,\n
32                    NULL, CARDDETECT_TASK_PRIORITY, NULL);
33        xTaskCreate(FlexCanTask, (char const *)"FlexCanTask", FLEXCAN_TASK_STACK_SIZE,\n
34                    NULL, FLEXCAN_TASK_PRIORITY, &flexCanTaskHandle);
35        xTaskCreate(ShutdownTask, (char const *)"ShutdownTask", SHUTDOWN_TASK_STACK_SIZE,\n
36                    NULL, SHUTDOWN_TASK_PRIORITY, &shutdownTaskHandle);
37    }
38    else if (op_mode == DRIVE_MODE) // Si el modo de operación es DRIVE_MODE.
39    {
40        Init_RTC(false); // Inicializar el RTC.
41        Init_Drive(); // Inicializar la configuración de MSD.
42        // Crear tarea para el modo de MSD.
43        xTaskCreate(DriveTask, (char const *)"DriveTask", DRIVE_TASK_STACK_SIZE, &g_msc,\n
44                    DRIVE_TASK_PRIORITY, &g_msc.application_task_handle);
45    }
46
47    // Iniciar el planificador de tareas de FreeRTOS.
48    vTaskStartScheduler();
49
50    /* Forzar que un contador se coloque en memoria. */
51    volatile static int i = 0;
52    /* Entrar en un bucle infinito, incrementando simplemente un contador. */
53    while(1) {
54        i++; // Incrementar el contador.
55        /* NOP 'Dummy' para permitir el avance paso a paso a nivel de fuente
56         * del bucle while() ajustado */
57        __asm volatile ("nop");
58    }
59    return 0 ; // Este punto no se alcanza debido al bucle infinito.
60 }

```

Figura 11.21: Subrutina main

```

1 void DriveTask(void *pvParameters)
2 {
3 #if BOARD == CANDLE
4     // Escribir en los LEDs para indicar fallo. Desaparece cuando se detecta la SD.
5     BOARD_WriteLEDs(BOARD_CODE_6_SDFAIL);
6 #endif
7
8     // Inicializar la aplicación del dispositivo USB.
9     USB_DeviceApplicationInit();
10
11    // Comprobar si el manejador del dispositivo USB se ha inicializado correctamente.
12    if (g_msc.deviceHandle)
13    {
14        // Intentar crear una tarea para el dispositivo USB.
15        if (xTaskCreate(USB_DeviceTask,           /* puntero a la tarea */
16                        (char const *)"usb device task", /* nombre de la tarea para depuración */
17                        5000L / sizeof(portSTACK_TYPE), /* tamaño del stack de la tarea */
18                        g_msc.deviceHandle,           /* argumento de inicio opcional de la tarea */
19                        5,                           /* prioridad inicial */
20                        &g_msc.device_task_handle    /* manejador de la tarea opcional para crear */
21                        ) != pdPASS)
22        {
23            // Si la creación de la tarea falla, mostrar un mensaje de error.
24            usb_echo("usb device task create failed!\r\n");
25            return; // Finalizar la ejecución de la tarea.
26        }
27 #if BOARD == CANDLE
28     // Escribir en los LEDs para indicar que la unidad está operativa.
29     BOARD_WriteLEDs(BOARD_CODE_3_DRIVEUP);
30 #elif BOARD == FRDM
31     // Encender el LED azul si la placa es FRDM.
32     GPIO_PinWrite(BOARD_LED_BLUE_GPIO, BOARD_LED_BLUE_PIN, LOGIC_LED_ON);
33 #endif
34    }
35    // Suspender la tarea actual indefinidamente.
36    vTaskSuspend(NULL);
37 }

```

Figura 11.22: Subrutina DriveTask

```

1 void FlexCanTask(void *pvParameters)
2 {
3     can_msg_t can_msg; // Definir estructura para el mensaje CAN.
4     while (1) // Bucle infinito para la tarea.
5     {
6         /* Iniciar la recepción de datos a través del Rx Fifo */
7         rxFifoXfer.frame = &rxFrame;
8         (void)FLEXCAN_TransferReceiveFifoNonBlocking(CAN_BASE, &flexcanHandle, &rxFifoXfer);
9
10        /* Esperar hasta que la recepción del mensaje Rx se complete. */
11        if (xSemaphoreTake(s_FlexCanSemaphore, portMAX_DELAY) == pdTRUE)
12        {
13            // Crear una estructura de mensaje CAN con los datos recibidos.
14            RTC_GetDatetime(RTC, &can_msg.timestamp); // Obtener la marca de tiempo actual.
15            can_msg.id = FLEXCAN_ID_INVERSE(rxFrame.id); // Asignar el ID del mensaje.
16            can_msg.length = rxFrame.length; // Asignar la longitud del mensaje.
17            // Asignar los datos recibidos al mensaje CAN.
18            can_msg.data[0] = rxFrame.dataByte7;
19            can_msg.data[1] = rxFrame.dataByte6;
20            can_msg.data[2] = rxFrame.dataByte5;
21            can_msg.data[3] = rxFrame.dataByte4;
22            can_msg.data[4] = rxFrame.dataByte3;
23            can_msg.data[5] = rxFrame.dataByte2;
24            can_msg.data[6] = rxFrame.dataByte1;
25            can_msg.data[7] = rxFrame.dataByte0;
26
27            // Enviar la estructura del mensaje CAN a la cola de mensajes CAN.
28            xQueueSend(s_CanQueueHandle, &can_msg, portMAX_DELAY);
29        }
30    }
31    // Suspender la tarea actual indefinidamente.
32    vTaskSuspend(NULL);
33 }

```

Figura 11.23: Subrutina FlexCanTask

```

1 void CardDetectTask(void *pvParameters)
2 {
3     // Crear un semáforo binario para la detección de la tarjeta SD.
4     s_CardDetectSemaphore = xSemaphoreCreateBinary();
5
6     // Configurar la tarjeta SD y registrar la función de callback para la detección de la tarjeta.
7     BOARD_SD_Config(&g_sd, SDCARD_DetectCallBack, BOARD_SDMMC_SD_HOST_IRQ_PRIORITY, NULL);
8
9     /* Inicializar el host SD */
10    if (SD_HostInit(&g_sd) == kStatus_Success)
11    {
12        sdHostInit = true; // Indicar que la inicialización del host de SD ha sido exitosa.
13        while (true)
14        {
15            /* Tomar el semáforo de detección de tarjeta */
16            if (xSemaphoreTake(s_CardDetectSemaphore, portMAX_DELAY) == pdTRUE)
17            {
18                // Verificar si el estado de inserción de la tarjeta ha cambiado.
19                if (s_cardInserted != s_cardInsertStatus)
20                {
21                    s_cardInserted = s_cardInsertStatus; // Actualizar el estado de inserción de la tarjeta.
22
23                    // Si la tarjeta está insertada, inicializarla y preparar el sistema de archivos.
24                    if (s_cardInserted)
25                    {
26                        // Apagar y luego encender la tarjeta para resetearla.
27                        SD_SetCardPower(&g_sd, false); // Apagar la tarjeta.
28                        SD_SetCardPower(&g_sd, true); // Encender la tarjeta.
29
30                        // Intentar crear el sistema de archivos en la tarjeta.
31                        if (DEMO_MakeFileSystem() != kStatus_Success)
32                        {
33                            continue; // Si falla, continuar intentando.
34                        }
35                        // Notificar a la tarea de acceso a archivos que el sistema de archivos está listo.
36                        xTaskNotifyGive(*fileAccessTaskHandle);
37                    }
38                }
39                // Si no hay tarjeta insertada, imprimir mensaje solicitando su inserción.
40                if (!s_cardInserted)
41                {
42                    PRINTF("\r\nPlease insert a card into board.\r\n");
43                }
44            }
45        }
46    }
47    else
48    {
49        // Si la inicialización del host SD falla, imprimir mensaje de error.
50        PRINTF("\r\nSD host init fail\r\n");
51    }
52    // Suspender la tarea actual indefinidamente.
53    vTaskSuspend(NULL);
54 }

```

Figura 11.24: Subrutina CardDetectTask

```

1 void FileAccessTask(void *pvParameters)
2 {
3     rtc_datetime_t date; // Estructura para almacenar la fecha y hora actuales.
4     FRESULT error; // Variable para almacenar el resultado de las operaciones de archivo.
5
6     // Esperar una notificación para iniciar la tarea.
7     xTaskNotifyWait(ULONG_MAX, ULONG_MAX, NULL, portMAX_DELAY);
8
9     // Inicializar el archivo
10    char fileName[30]; // Array para almacenar el nombre del archivo.
11    RTC_GetDatetime(RTC, &date); // Obtener la fecha y hora actuales.
12    // Formatear el nombre del archivo con la fecha y hora actuales.
13    sprintf(fileName, "logs/02d_02d/02d_02d.csv", (date.year)%100, date.month,\n
14           date.day, date.hour, date.minute, date.second);
15    // Intentar abrir el archivo para escribir y en modo de añadido.
16    error = f_open(&g_fileObject, _T(fileName), FA_WRITE | FA_OPEN_APPEND);
17    if (error)
18    {
19        // Si el archivo no existe, intentar crear uno nuevo.
20        if (error == FR_NO_FILE)
21        {
22            error = f_open(&g_fileObject, _T(fileName), FA_WRITE | FA_CREATE_NEW);
23            if (error)
24            {
25                // Encender LEDs para indicar error.
26                BOARD_WriteLED(BOARD_LED_2_PIN, LOGIC_LED_ON);
27                BOARD_WriteLED(BOARD_LED_1_PIN, LOGIC_LED_ON);
28                return; // Salir de la tarea si hay un error.
29            }
30        }
31        else
32        {
33            // Encender LEDs para indicar error.
34            BOARD_WriteLED(BOARD_LED_2_PIN, LOGIC_LED_ON);
35            BOARD_WriteLED(BOARD_LED_1_PIN, LOGIC_LED_ON);
36            return; // Salir de la tarea si hay un error.
37        }
38    }
39    fileOpen = true; // Indicar que el archivo ha sido abierto.
40
41    // Escribir la cabecera en el archivo.
42    char s_buffer0[] = "Year,Month,Day,Hour,Minute,Second,Milisecond,ID,Data Length,\n
43                      B7,B6,B5,B4,B3,B2,B1,B0\r\n";
44    error = f_write(&g_fileObject, s_buffer0, sizeof(s_buffer0) - 1, &bytesWritten);
45    if (error || (bytesWritten != sizeof(s_buffer0) - 1))
46    {
47        // Encender LEDs para indicar error y cerrar archivo.
48        BOARD_WriteLED(BOARD_LED_2_PIN, LOGIC_LED_ON);
49        BOARD_WriteLED(BOARD_LED_1_PIN, LOGIC_LED_ON);
50        CloseLoggingFile();
51        return; // Salir de la tarea si la cabecera no puede ser escrita.
52    }
53    f_sync(&g_fileObject); // Sincronizar el archivo.
54
55    // Inicialización del mensaje CAN.
56    QueueHandle_t canMsgQueue = getCanMsgQueue(); // Obtener la cola de mensajes CAN.
57    can_msg_t canMsg; // Estructura para almacenar mensajes CAN.
58    bufferSize = sizeof(buffer); // Tamaño total del buffer.
59    bufferWriteThreshold = bufferSize - 50; // Umbral para escribir en el archivo.
60
61    while (1)
62    {
63        // Esperar por un mensaje CAN en la cola.
64        if (xQueueReceive(canMsgQueue, &canMsg, portMAX_DELAY) == pdTRUE)
65        {
66            bufferIndex = writeToBuffer(buffer, bufferIndex, &canMsg); // Escribir el mensaje CAN en el buffer.
67
68            // Verificar si se alcanzó el umbral para escribir en el archivo.
69            if (bufferIndex >= bufferWriteThreshold)
70            {
71                // Escribir el contenido del buffer en el archivo.
72                error = f_write(&g_fileObject, buffer, bufferIndex, &bytesWritten);
73                if (error || (bytesWritten != bufferIndex))
74                {
75                    // Encender LEDs para indicar error y cerrar archivo.
76                    BOARD_WriteLED(BOARD_LED_2_PIN, LOGIC_LED_ON);
77                    BOARD_WriteLED(BOARD_LED_1_PIN, LOGIC_LED_ON);
78                    CloseLoggingFile();
79                    break; // Salir del bucle si hay un error.
80                }
81                bufferIndex = 0; // Reiniciar el índice del buffer después de escribir.
82            }
83        }
84    }
85    // Suspender la tarea actual indefinidamente.
86    vTaskSuspend(NULL);
87 }

```

Figura 11.25: Subrutina FileAccessTask

```

1 void ADC16_IRQ_HANDLER_FUNC(void)
2 {
3     // Comprobar si la tarea está en ejecución y si el valor de ADC está por debajo del umbral.
4     if (taskRunning)
5     {
6         taskRunning = false; // Establecer la bandera indicando que la tarea ya no está en ejecución.
7         DisableIRQ(ADC16_IRQn); // Desactivar las interrupciones del ADC para prevenir más interrupciones.
8         ADC16_Deinit(ADC16_BASE); // Desinicializar el ADC para detener las operaciones de conversión.
9
10    // Despertar la tarea de apagado dando un semáforo desde una ISR (Interrupción de Servicio).
11    xSemaphoreGiveFromISR(s_ShutdownSemaphore, &xHigherPriorityTaskWokenByPost);
12
13    // Si al dar el semáforo se despertó una tarea de mayor prioridad, realizar un cambio de contexto.
14    if (xHigherPriorityTaskWokenByPost == pdTRUE)
15    {
16        portYIELD_FROM_ISR(xHigherPriorityTaskWokenByPost); // Cambiar el contexto al retornar de la ISR.
17    }
18 }
19 }
```

Figura 11.26: Interrupción de ADC

```

1 void ShutdownTask(void *pvParameters)
2 {
3     taskRunning = true; // Establecer la bandera indicando que la tarea está en ejecución.
4
5     while(1) // Bucle infinito para la tarea.
6     {
7         // Esperar indefinidamente hasta que se reciba el semáforo de apagado.
8         if (xSemaphoreTake(s_ShutdownSemaphore, portMAX_DELAY) == pdTRUE)
9         {
10            // Encender LED para indicar el inicio del proceso de apagado.
11            BOARD_WriteLED(BOARD_LED_0_PIN, LOGIC_LED_ON);
12
13            // Detener las tareas de la aplicación, como la tarea de registro.
14            StopLogging();
15
16            // Suspender esta tarea indefinidamente, deteniendo así la ejecución de la tarea de apagado.
17            vTaskSuspend(NULL);
18        }
19    }
20 }
```

Figura 11.27: Subrutina ShutDownTask

#### 11.2.4. Plan de prueba de módulos y de depuración del software

El plan de pruebas de software fue dividido según los distintos niveles de la arquitectura ilustrados en la figura 11.19. En primer lugar, probar el correcto funcionamiento de los controladores de bajo y medio nivel pertenecientes al SDK de NXP. En segundo lugar, probar el funcionamiento de las tareas principales de forma aislada, simulando artificialmente sus entradas y/o salidas. Finalmente, probar la integración de las tareas a través de la aplicación principal.

La figura 11.28 muestra el flujo de pruebas según sus etapas y respectivas dependencias.

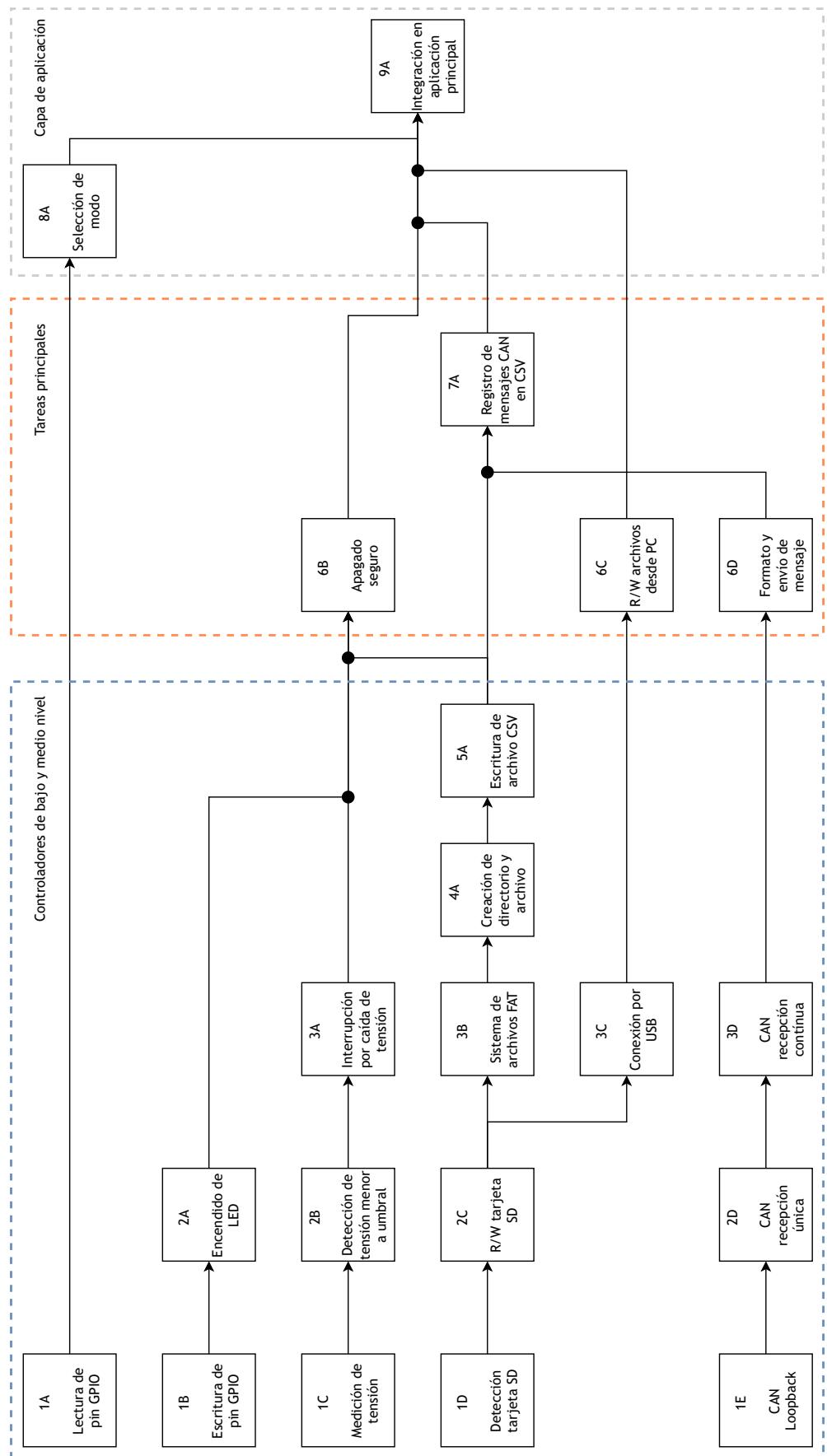


Figura 11.28: Plan de pruebas de SW

#### **11.2.4.1 Descripción de pruebas para controladores de bajo y medio nivel**

- 1A. Lectura de pin GPIO:** Verificar la capacidad del firmware para leer el estado (alto o bajo) de un pin GPIO específico.
- 1B. Escritura de pin GPIO:** Probar la funcionalidad de escribir (establecer alto o bajo) en un pin GPIO.
- 1C. Medición de tensión:** Evaluar la precisión y la capacidad del sistema para medir voltajes en entradas analógicas.
- 1D. Detección tarjeta SD:** Comprobar la detección correcta de una tarjeta SD insertada.
- 1E. CAN Loopback:** Verificar la comunicación interna CAN mediante el modo loopback, asegurando que los mensajes enviados sean correctamente recibidos por el mismo módulo.
- 2A. Encendido de LED:** Validar la función de controlar un LED (encendido/apagado) utilizando un pin GPIO específico en la placa de desarrollo.
- 2B. Detección de tensión menor a umbral:** Confirmar que el sistema puede detectar y reaccionar ante tensiones que caen por debajo de un umbral específico utilizando la función de comparación automática.
- 2C. R/W tarjeta SD:** Probar las operaciones básicas de lectura y escritura en una tarjeta SD.
- 2D. CAN recepción única:** Evaluar la capacidad del sistema para recibir correctamente un mensaje CAN único desde un nodo externo.
- 3A. Interrupción por caída de tensión:** Comprobar que el sistema puede ejecutar adecuadamente una interrupción generada por una caída de tensión según el umbral definido.
- 3B. Sistema de archivos FAT:** Verificar la posibilidad de montar un sistema de archivos FAT sobre la tarjeta SD.
- 3C. Conexión por USB:** Probar que una PC reconozca al dispositivo como MSD a través de USB.
- 3D. CAN recepción continua:** Evaluar la capacidad del sistema para recibir mensajes CAN de forma continua sin pérdida de datos desde un nodo externo.
- 4A. Creación de directorio y archivo:** Verificar que el sistema puede crear directorios y archivos dentro del sistema de archivos FAT en la tarjeta SD.
- 5A. Escritura de archivo CSV:** Probar la capacidad de generar y escribir en archivos CSV.

#### **11.2.4.2 Descripción de pruebas para tareas principales**

- 6B. Apagado seguro:** Evaluar la implementación de un procedimiento de apagado seguro que garantice la integridad de los datos y el estado del sistema.
- 6C. R/W archivos desde PC:** Comprobar la funcionalidad de acceso y gestión de archivos en la tarjeta SD desde un PC a través de USB.
- 6D. Formato y envío de mensaje CAN:** Probar la capacidad del sistema para formatear y enviar los mensajes CAN recibidos hacia una cola utilizada por otra tarea.
- 7A. Registro de mensajes CAN en CSV:** Evaluar la capacidad de capturar mensajes CAN y registrarlos en un archivo CSV con el formato adecuado.

#### **11.2.4.3 Descripción de pruebas para la capa de aplicación**

- 8A. Selección de modo:** Validar la funcionalidad que permite al programa seleccionar diferentes modos de operación del dispositivo según el origen de la alimentación.
- 9A. Integración en aplicación principal:** Verificar la integración completa de todas las funcionalidades anteriores en la aplicación principal, asegurando que trabajan conjuntamente sin conflictos.

## 12. Construcción del prototipo

### 12.1. Definición de los módulos

En esta sección se describen los detalles de diseño, manufactura y ensamblaje del prototipo. Este se compone de los siguientes módulos:

- Placa de circuito impreso.
- Carcasa plástica.
- Elementos de sujeción (tornillos y tuercas).

En la tabla 17.3 se puede observar el listado de partes y componentes (BOM) que corresponden a la manufactura del prototipo.

### 12.2. Diseño de los circuitos impresos

La placa de circuito impreso (PCB por sus siglas en inglés, *Printed Circuit Board*) fue diseñada en Altium Designer. El PCB fue diseñado para implementar todas las funcionalidades deseadas respetando la especificación de dimensiones IMP-DIM-02.

Se utilizó un *layer stack-up* de 4 capas para el PCB a partir de las recomendaciones del fabricante del microcontrolador (NXP Semiconductors) en su manual de referencia. En la figura 12.1 se puede ver una captura del mismo.

#### 4-Layer PCB B:

- Layer 1 (top – MCU location)—signals and poured power
- Layer 2 (inner)—ground plane
- Thick core
- Layer 3 (inner)—ground plane
- Layer 4 (bottom)—signals and poured power

Figura 12.1: Recomendaciones de NXP para diseñar el *layer stack-up* del PCB.

El *routing* (enrutamiento) del PCB se realizó siguiendo ciertas consideraciones especiales. Las señales del bus CAN se diseñaron como un par diferencial con impedancia característica de  $120\ \Omega$ , las señales de datos del puerto USB se diseñaron como un par diferencial de impedancia característica de  $90\ \Omega$ , y las señales del bus de datos de la tarjeta SD se diseñaron con longitudes equiparadas (*matched lengths*) para garantizar la sincronía entre las señales de *clock*, *command* y datos que son transmitidas a la tarjeta. Algunos puntos destacados del PCB se pueden observar en la figura 12.2.

Otro punto a destacar es el enrutamiento del circuito de regulación de potencia. El regulador DC-DC se diseñó de forma tal que las longitudes de los trazos sean minimizados, ya que de esta forma se reduce la inductancia parásita que presentan los trazos del PCB. Adicionalmente, tanto el controlador del regulador como el inductor que forma parte del circuito fueron posicionados de la forma más alejada posible del resto del circuito. Así, se busca minimizar la introducción de ruido electromagnético en el resto de las señales que conforman la placa.

En la figura 12.3 se puede ver una representación tridimensional del PCB que fue diseñado, mientras que en la figura 12.4 se indican los elementos más importantes del circuito. La dimensión del PCB diseñado es de 80 mm x 85 mm.

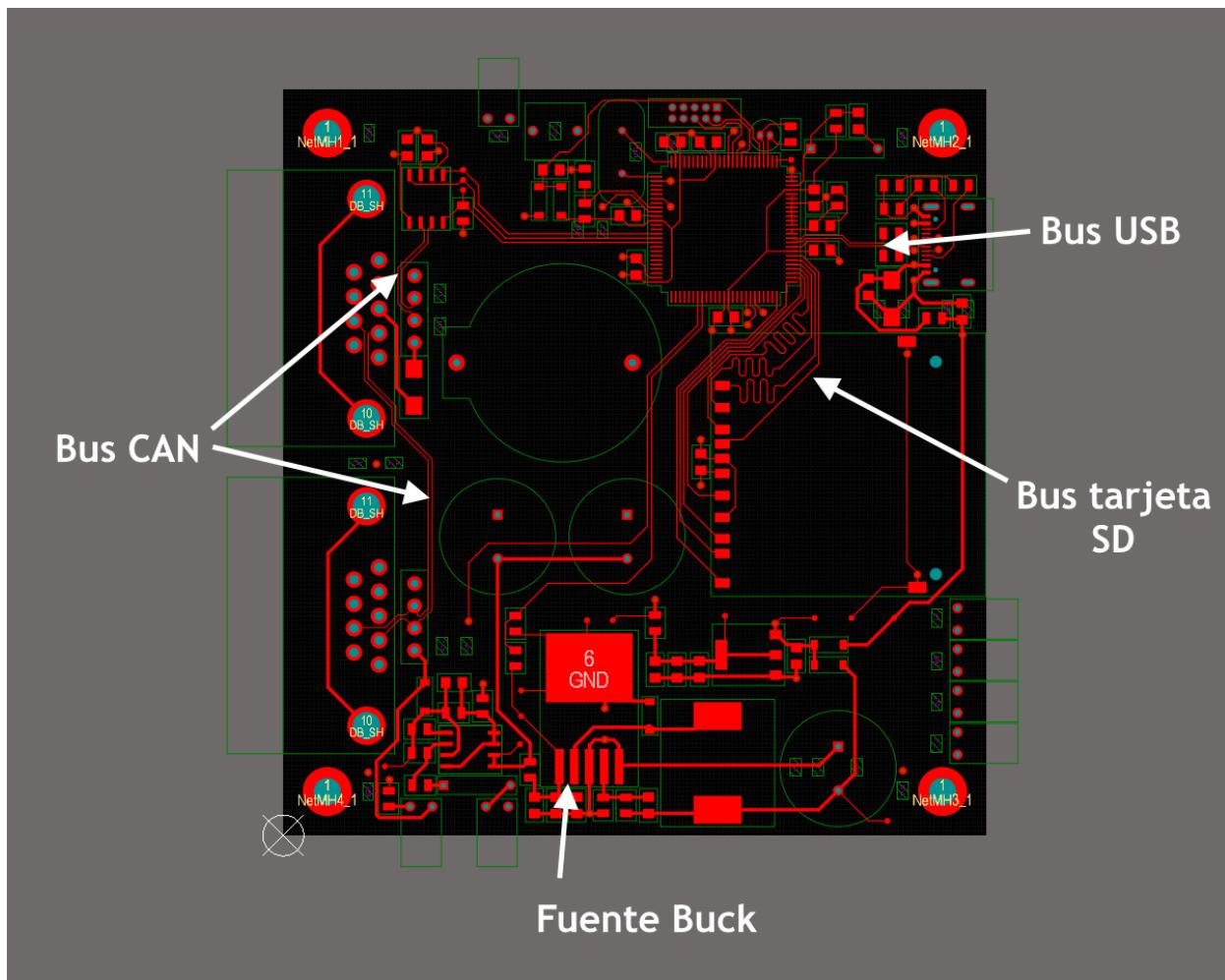


Figura 12.2: Elementos destacados del *routing* del PCB.

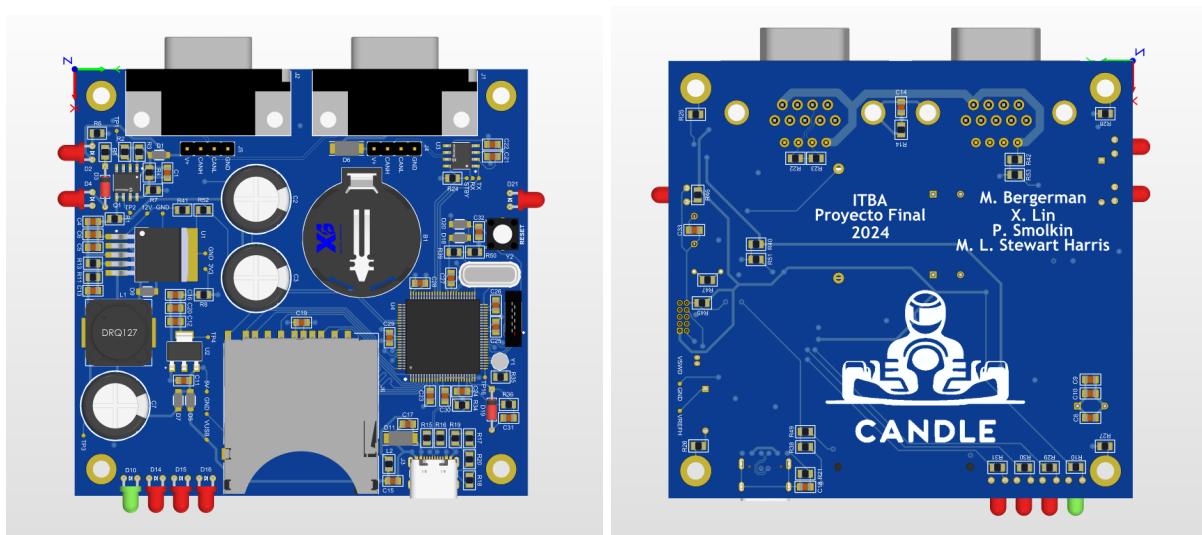


Figura 12.3: Vista superior e inferior de la placa electrónica diseñada.

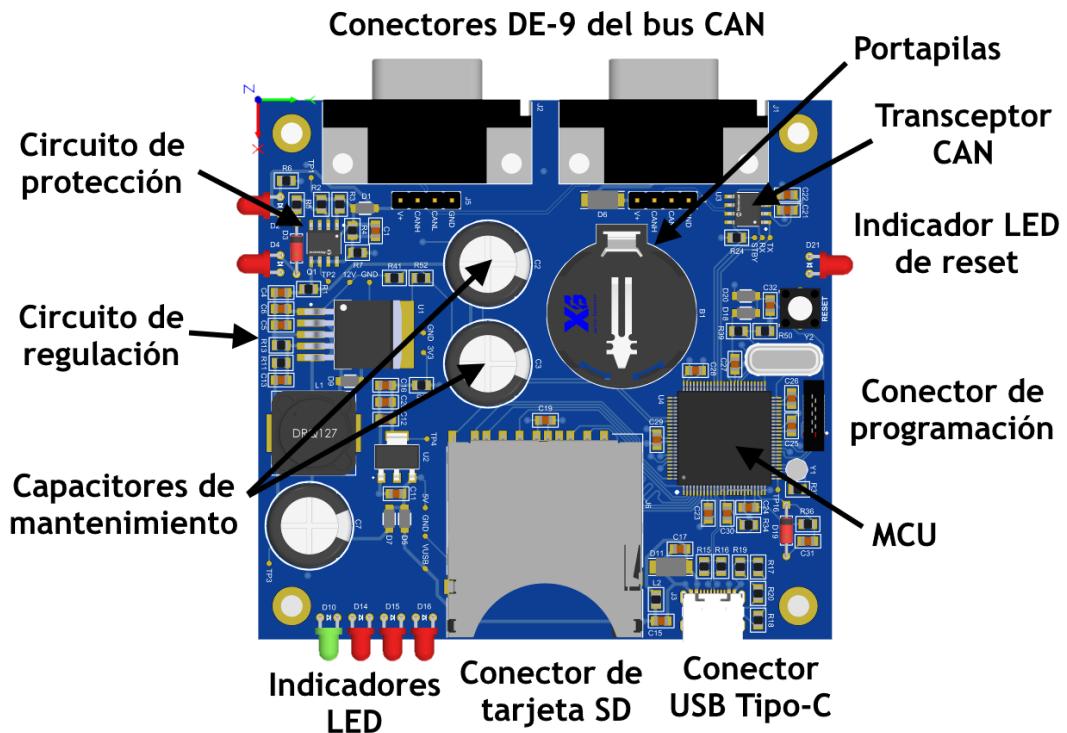


Figura 12.4: Vista superior de la placa electrónica diseñada con indicaciones.

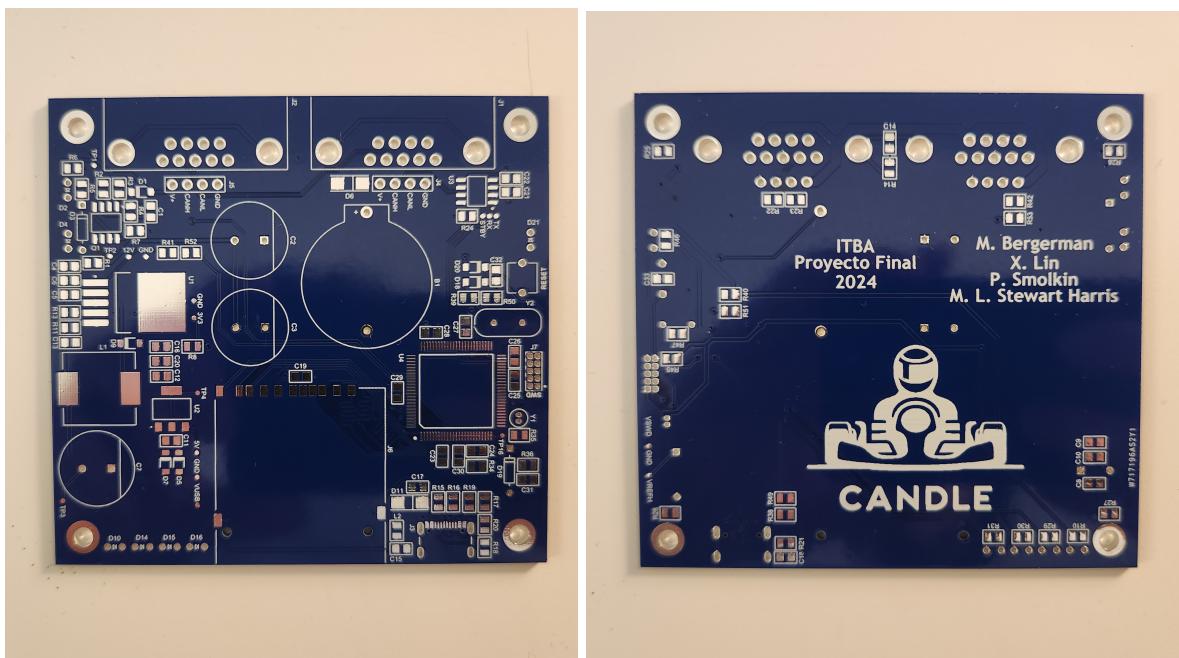


Figura 12.5: Vista superior e inferior de la placa electrónica fabricada.

Para la manufactura del prototipo, se encargó la fabricación del PCB diseñado a la empresa PCBWay<sup>3</sup> y se realizó el ensamblaje de los componentes en el laboratorio de electrónica del ITBA. En la figura 12.6 se muestran imágenes del PCB fabricado, en la figura 12.6 se muestran imágenes del proceso de colocación de componentes y soldado de los mismos, y en la figura 12.7 se muestra el PCB ensamblado por completo.

<sup>3</sup><https://www.pcbway.com/>

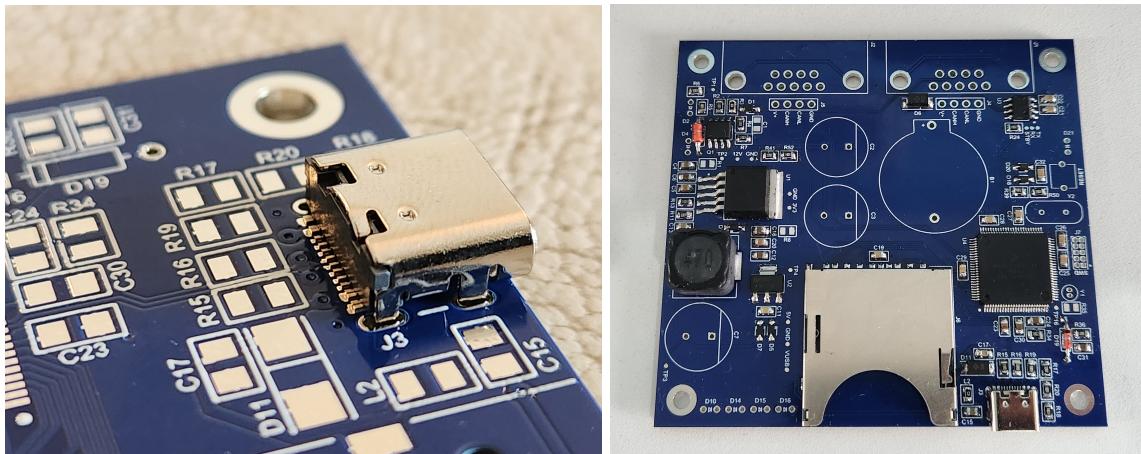


Figura 12.6: Imagenes del proceso de ensamblaje y soldado de los componentes del PCB.

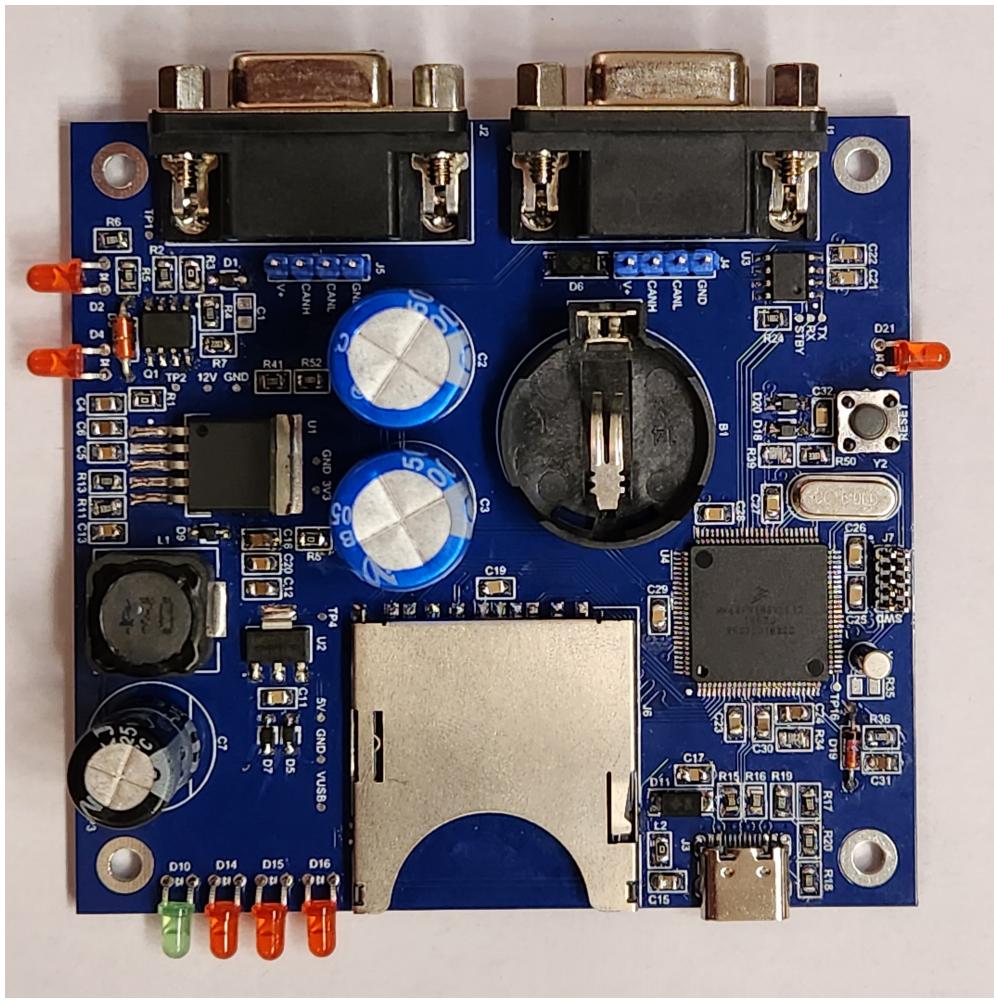


Figura 12.7: Placa electrónica ensamblada.

### 12.3. Diseño mecánico

Para cumplir con las especificaciones de protección ante polvo (INT-MEC-01) y de sujeción al vehículo (INT-MEC-02), el diseño del prototipo incorpora una carcasa plástica dentro de la cual se monta la placa

de circuito impreso fabricada. Esta carcasa debe cumplir adicionalmente con las especificaciones de peso (INT-DIM-01) y dimensiones (IMP-DIM-02).

La carcasa plástica fue diseñada en el software de diseño asistido por computadora (CAD por sus siglas en inglés, *Computer Aided Design*) OnShape<sup>4</sup>. La carcasa se compone de dos partes, una superior y una inferior. Para ensamblar la carcasa se debe colocar el PCB sobre la parte inferior, luego se apoya la parte superior. Finalmente, se utilizan 4 tornillos de diámetro 1/8 de pulgada y longitud de 2 pulgadas para asegurar las dos partes de la carcasa y el PCB; los tornillos se colocan por la parte superior y se ajustan mediante el uso de tuercas que se ubican en la parte inferior.

Las dos partes de la carcasa cuentan con ranuras de diferentes factores de forma que exponen a los conectores e indicadores LED que se ubican en el PCB. Adicionalmente, la carcasa cuenta con “aletas” a ambos lados de la parte inferior donde hay agujeros de sujeción a una separación de 35 mm entre sí.

En la figura 12.8 se puede observar un diagrama de despiece de la carcasa y el PCB (no se muestran los tornillos y tuercas que sujetan ambas partes de la carcasa).

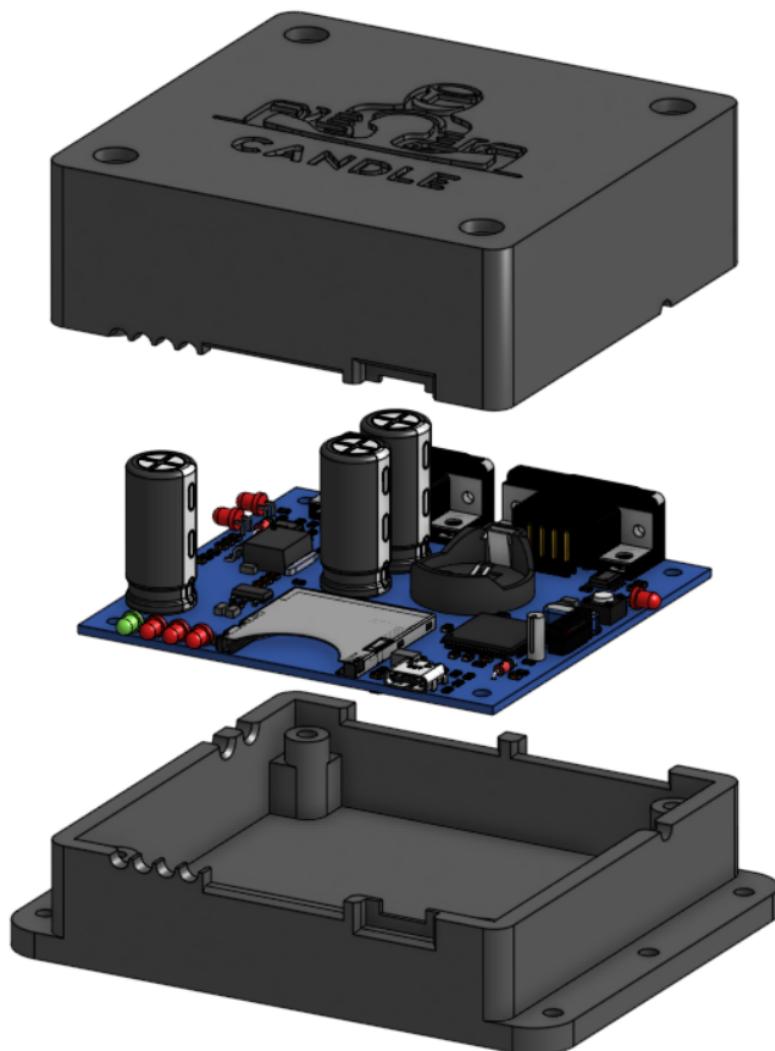


Figura 12.8: Diagrama de despiece de la carcasa y el PCB.

---

<sup>4</sup><https://www.onshape.com/>

La carcasa se fabricó utilizando tecnología con método de deposición de material plástico de tipo PLA. En las figuras 12.9 y 12.10 se pueden observar imágenes de la carcasa fabricada con el PCB montado. Este material no cumple con la especificación de temperatura IMP-OPE-01, por lo cual a pesar de haber sido empleado para la construcción del prototipo, un plástico con un rango apropiado de temperatura debe ser seleccionado para el producto final.



Figura 12.9: Carcasa de plástico con el PCB montado.



Figura 12.10: Carcasa de plástico con el PCB montado.

## 13. Validación del prototipo

### 13.1. Plan y protocolos especiales de medición

Se realizaron pruebas de módulos de hardware y software. Luego se realizaron pruebas de validación del producto (ver tabla 13.1). Por último, se realizaron pruebas de integración entre el prototipo y el karting.

Las primeras pruebas en realizarse fueron las pruebas de la protección del módulo de alimentación. Se realizaron con la resistencia R1 des-populada para evitar dañar el resto del circuito del prototipo en caso de fallar la prueba. Una vez aprobada, R1 fue soldada.

### 13.2. Conexiones para pruebas

Las figuras 13.1 y 13.2 muestran el estado de los bancos de pruebas N°1 y N°2 mientras se ejecutaban pruebas de validación. La figura 13.3 muestra el detalle de la construcción de los cables que forman al bus CAN. La figura 13.4 muestra el nodo CAN transmisor. La figura 13.5 muestra el banco de pruebas N°1 utilizado para probar el requerimiento de protección de tensión inversa.

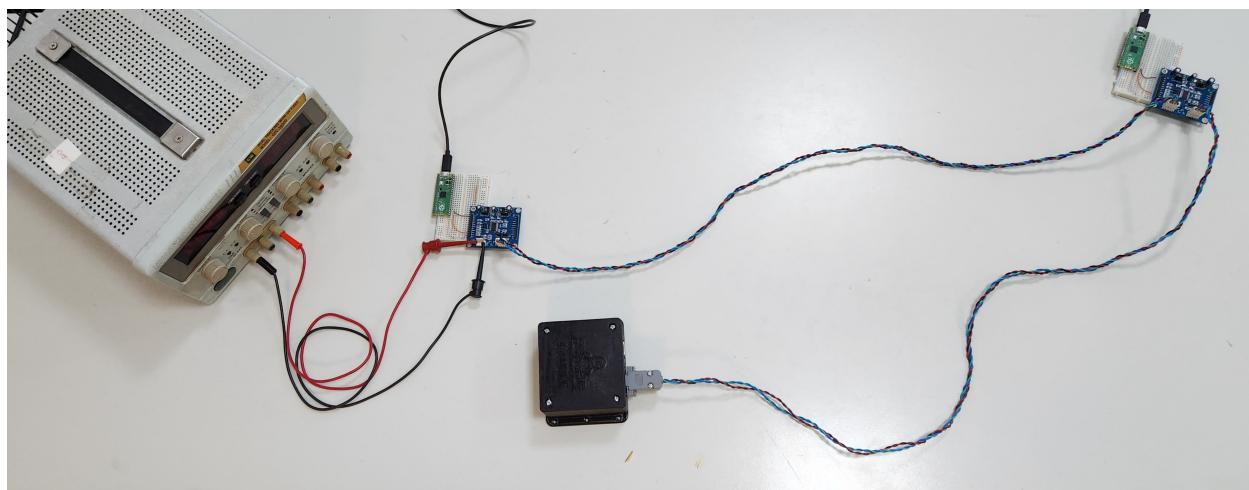


Figura 13.1: Banco de pruebas N°1 durante la ejecución de una prueba.

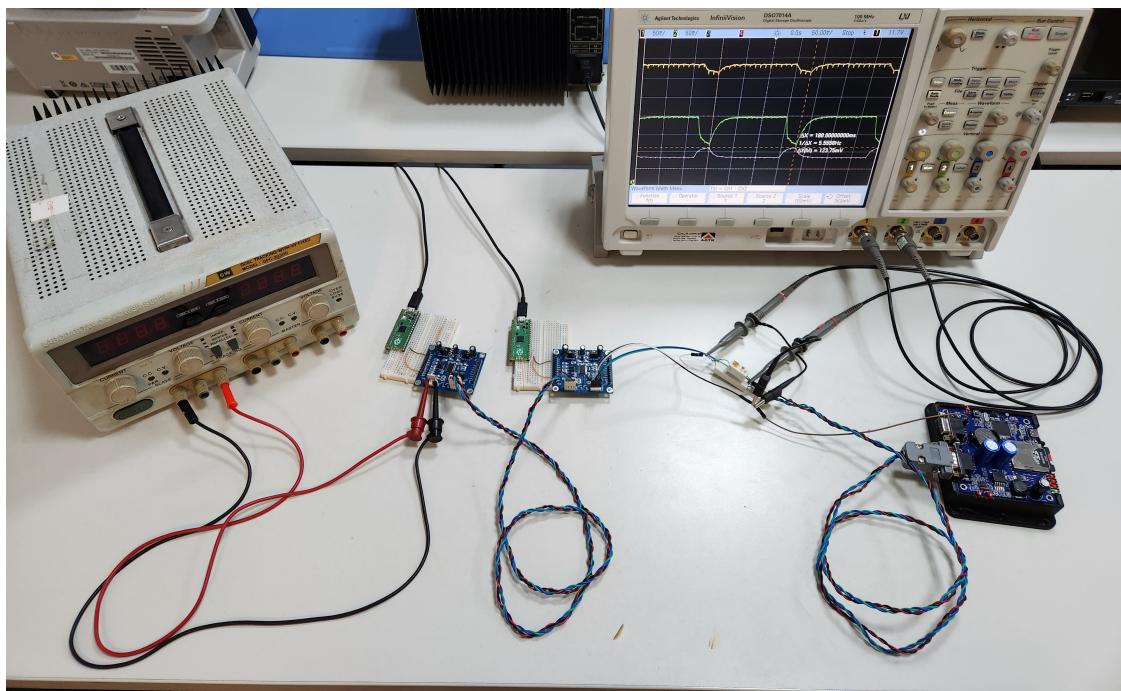


Figura 13.2: Banco de pruebas N°2 durante la ejecución de una prueba.

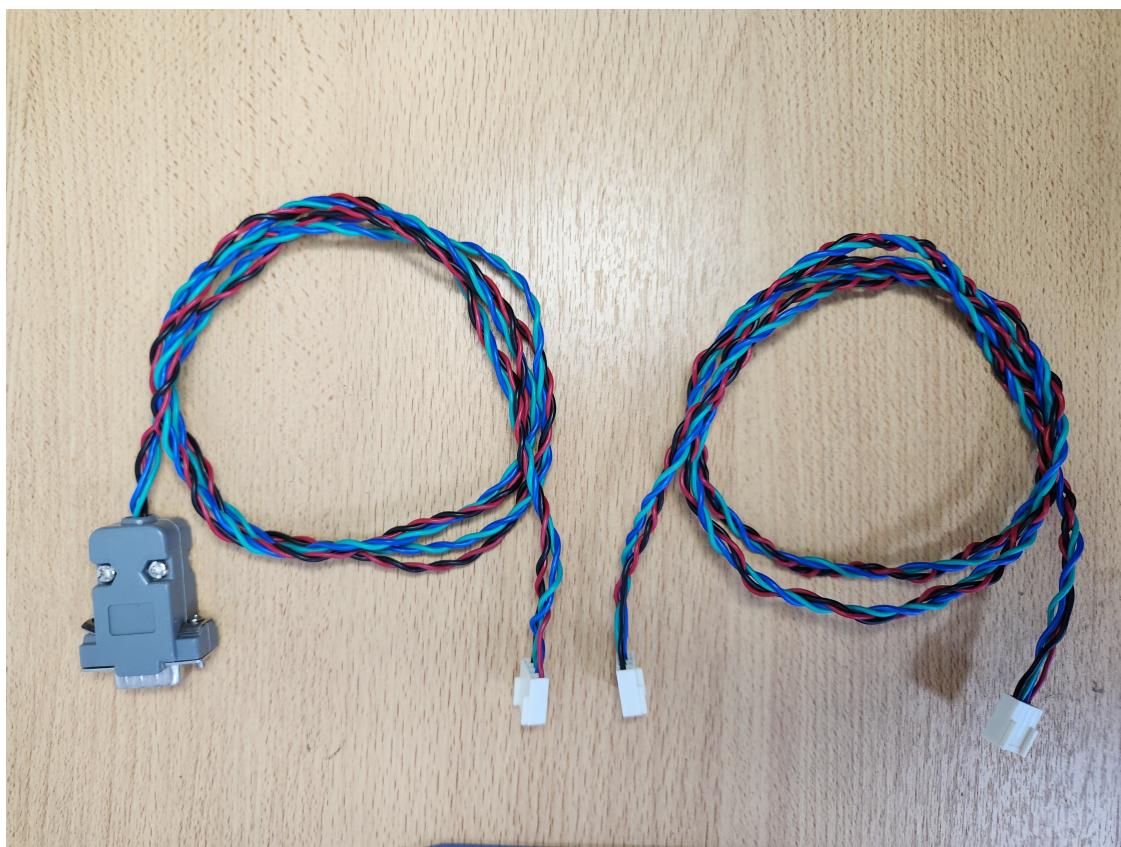


Figura 13.3: Cables de bus CAN para pruebas del prototipo.

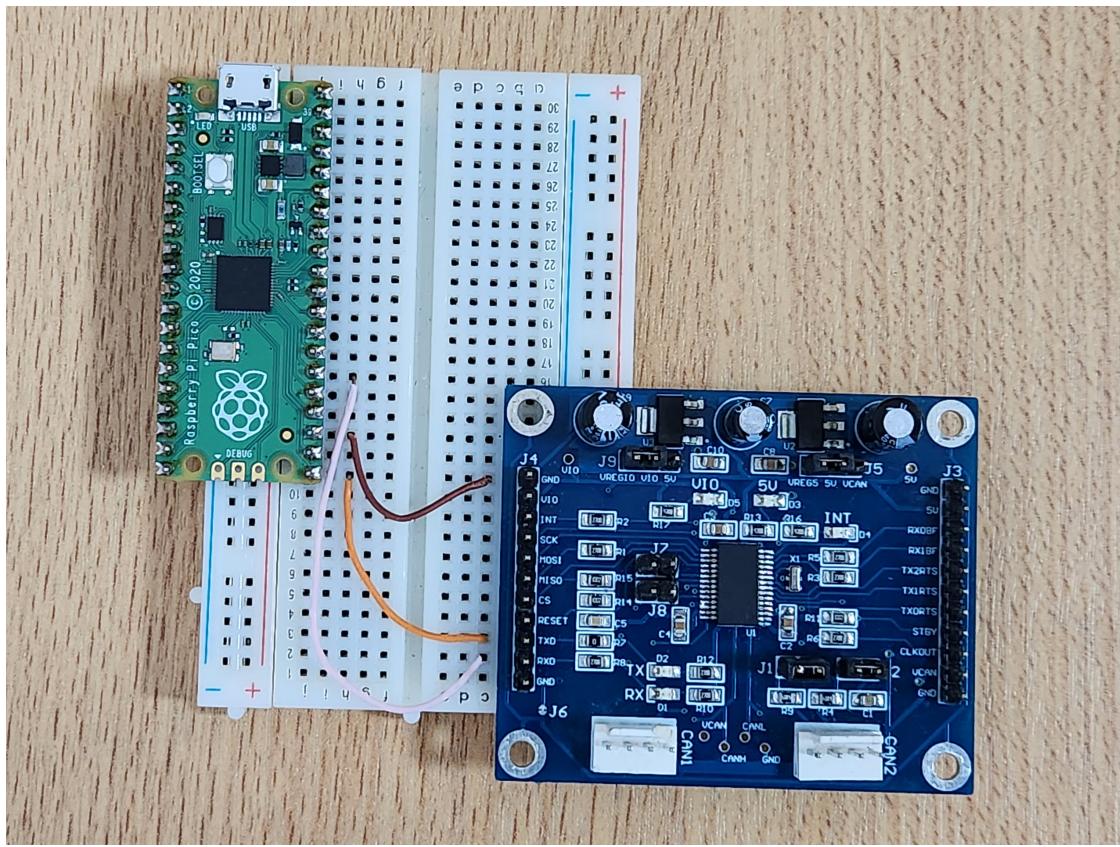


Figura 13.4: Nodo CAN transmisor utilizado en los bancos de prueba N°1 y N°2 para pruebas en el prototipo. El nodo CAN receptor es idéntico en hardware, y con diferente firmware.

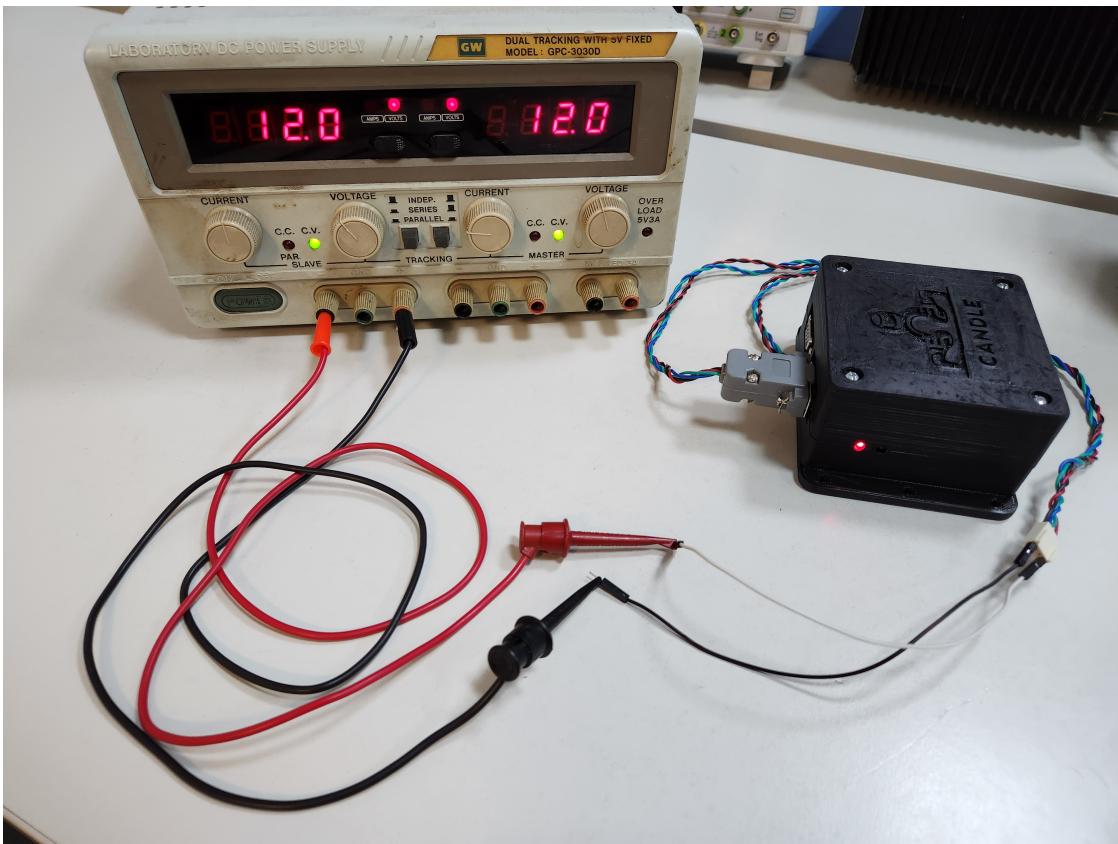


Figura 13.5: LED indicador de tensión inversa encendido durante la ejecución de una prueba.

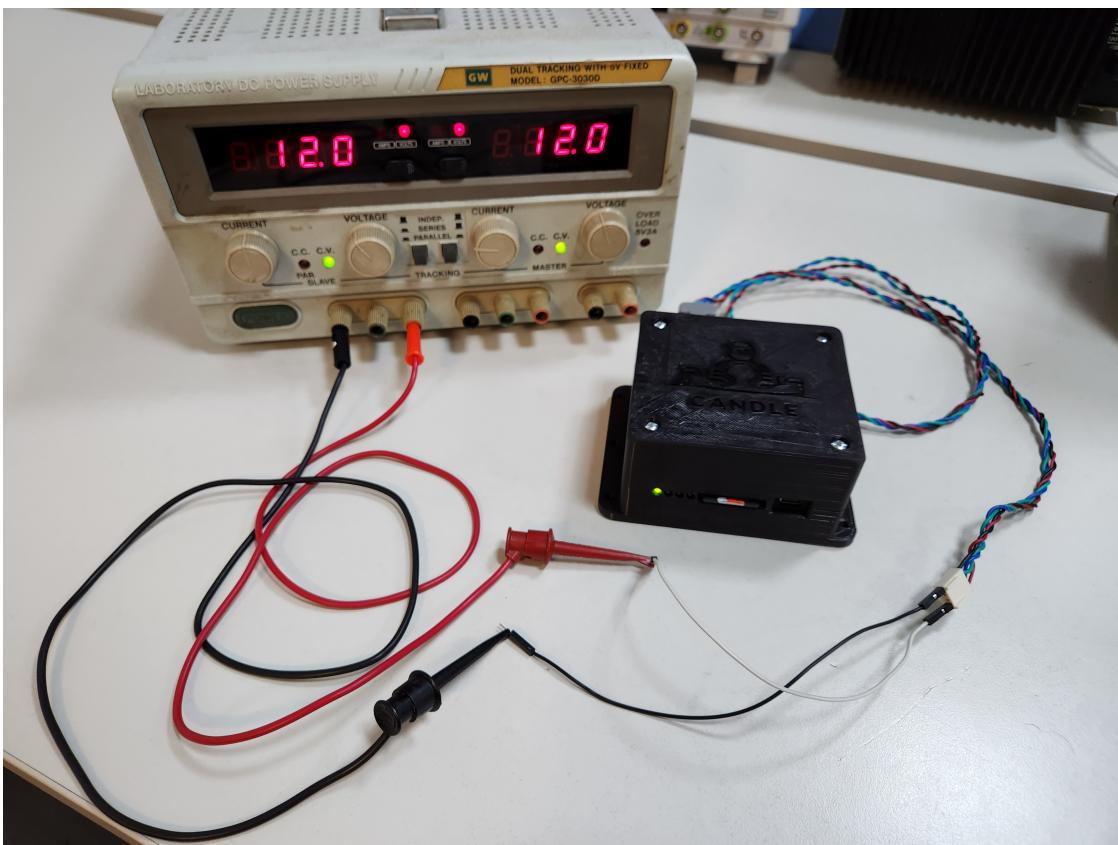


Figura 13.6: LED indicador de alimentación encendido.

La figura 13.7 muestra el banco de pruebas N3 mientras se pesa el prototipo.



Figura 13.7: Medición del peso del prototipo.

Las figuras 13.8 y 13.9 muestran el conexionado utilizado para las pruebas de integración entre el prototipo y el karting.



Figura 13.8: Conexionado entre el prototipo y el Karting.



Figura 13.9: Setup de prueba de integración entre el prototipo y el Karting.

### 13.3. Resultados

#### 13.3.1. Resultados de pruebas de módulos

Las pruebas de módulos fueron aprobadas exitosamente.

La figura 13.10 muestra el resultado de la prueba de la protección dentro del módulo de alimentación. Se observa que si se va aumentando la entrada desde 0V, la protección se desactiva a los 7V en vez de a los 10V. Luego, se vuelve a activar a los 30V, como era esperado.

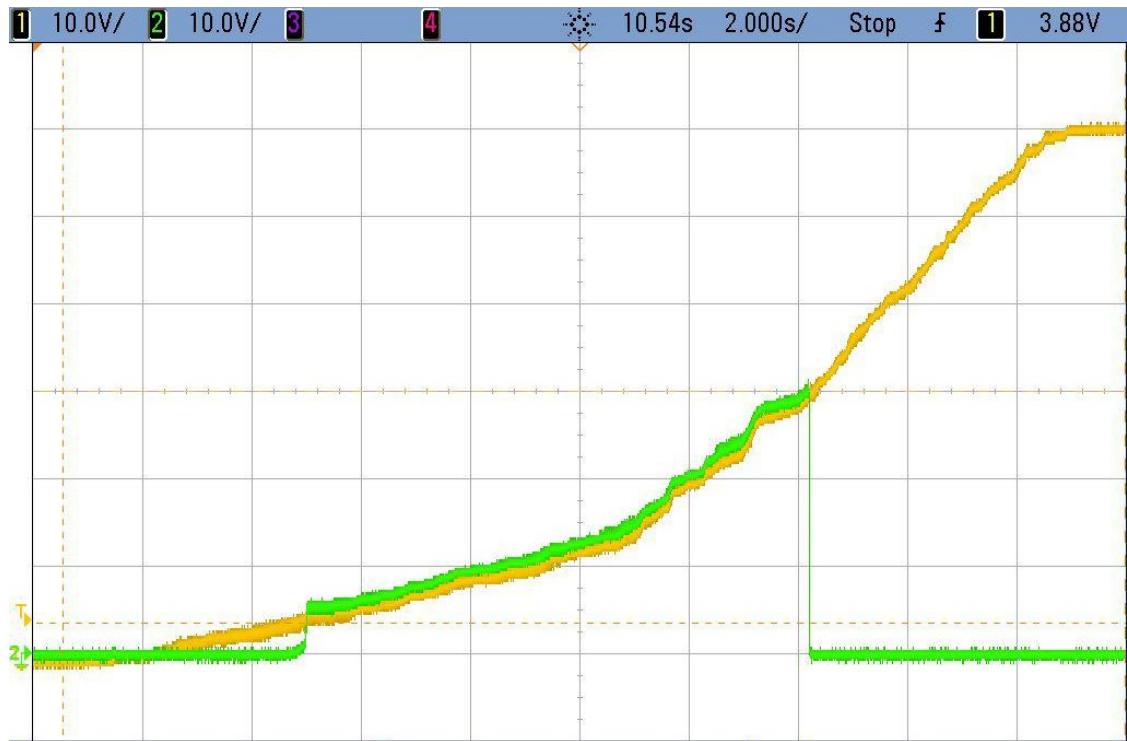


Figura 13.10: Medición de salida de circuito de protección para tensiones de entre 0V y 60V.

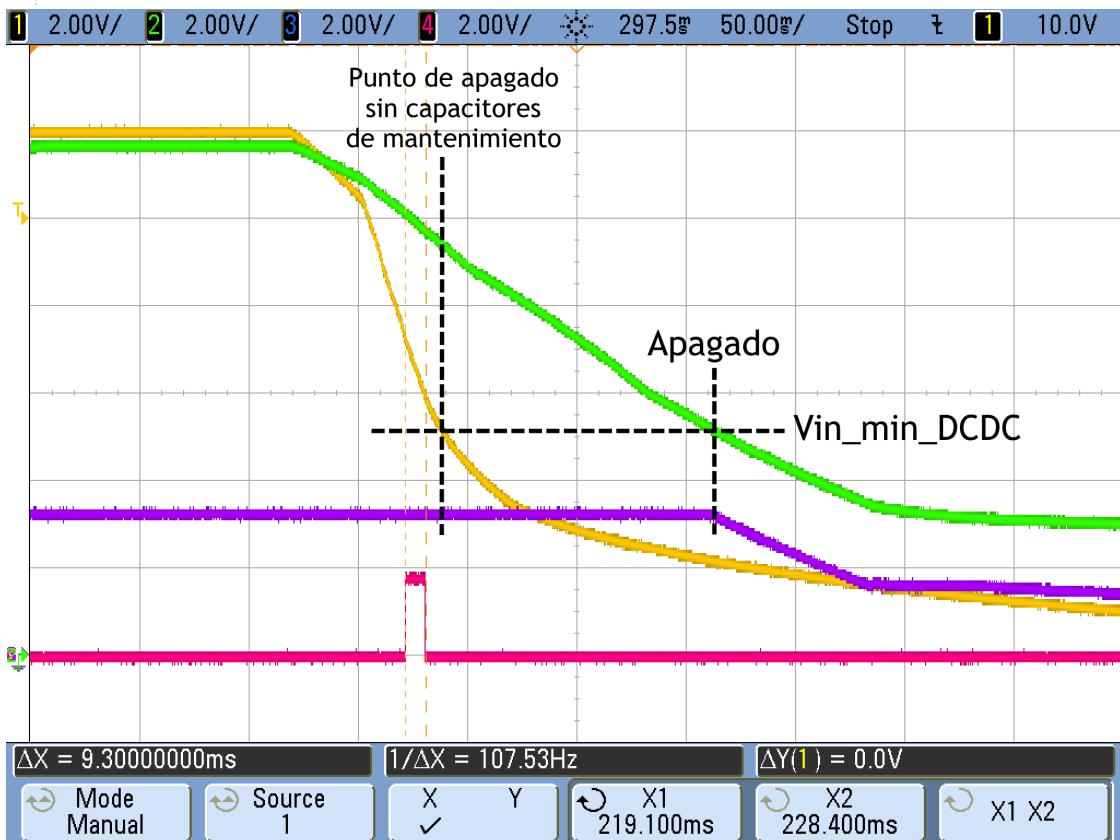


Figura 13.11: Relación entre la tarea *shutdownTask* y el apagado de las señales de alimentación internas. Medición de entrada del módulo de alimentación (amarillo), tensión en los capacitores de mantenimiento (verde), tiempo de ejecución de tarea *shutdownTask* (rosa), y alimentación del microcontrolador (violeta).

La figura 13.11 ilustra cómo la tarea *shutdownTask* depende del proceso de apagado de las señales de alimentación internas. En verde, se muestra la salida de la protección que alimenta al conversor DC-DC, mientras que en violeta se representa la salida del regulador lineal que suministra energía al microcontrolador. Este proceso se activa una vez que se interrumpe la alimentación externa proveniente del bus CAN.

Es importante mencionar que la tensión mínima de entrada del conversor DC-DC es de 5.3V. En la figura se puede apreciar cómo, al descender esta tensión por debajo del umbral mínimo, el sistema de regulación de tensión ya no es capaz de mantener la salida en 3.3V, lo que resulta en una disminución de la tensión de salida. En este punto, se considera que el microcontrolador ha sido apagado.

Es relevante señalar que la presencia de capacitores de regulación juega un papel fundamental en este proceso. Sin ellos, el apagado del microcontrolador ocurriría de manera más rápida, reduciendo el tiempo disponible para que *shutdownTask* complete sus tareas.

### 13.3.2. Resultados tests de validación

Test	Resultado
TST-MSD-1	OK
TST-SD-1	OK
TST-ALIM-1	OK
TST-ALIM-2	OK
TST-ALIM-3	OK
TST-DE9-1	OK
TST-FUN-1	OK
TST-FUN-2	OK
TST-PER-1	OK
TST-IMP-1	OK

Tabla 13.1: Resultados de los tests de validación corridos en el prototipo.

La figura 13.13 muestra la tensión medida en la resistencia de shunt al encender la fuente CC durante la prueba de validación TST-PER-1. Con  $R = 2,8\Omega$ , la corriente de inrush es

$$i = \frac{0,46V}{2,8\Omega} = 160mA$$

y la potencia pico consumida es

$$P = 12V \cdot 160mA = 2,0W < 3W$$

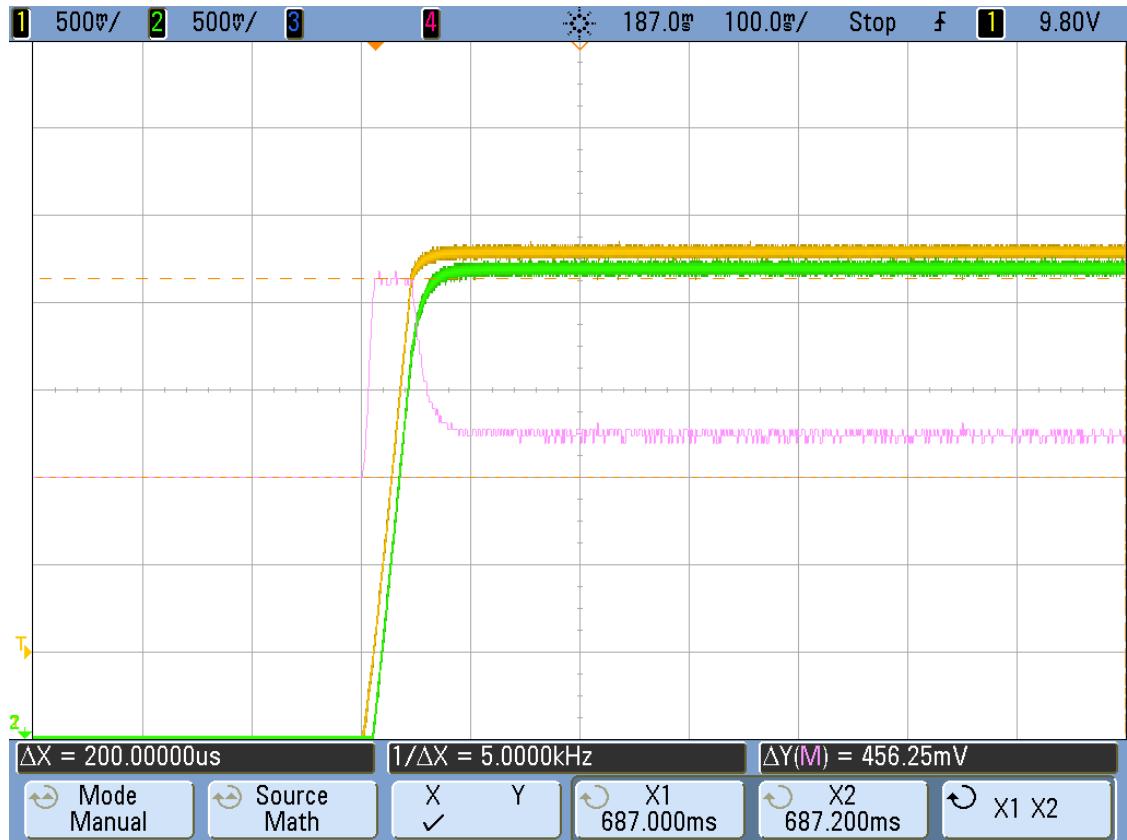


Figura 13.12: Medición de tensión en resistencia de shunt en el encendido para calcular la corriente de inrush.

La figura 13.13 muestra la tensión medida en la resistencia de shunt mientras se están recibiendo mensajes CAN durante la ejecución de TST-PER-1. Con  $R = 2,8\Omega$ , la corriente pico en esta sección es

$$i = \frac{0,15V}{2,8\Omega} = 53mA$$

y la potencia pico consumida es

$$P = 12V \cdot 53mA = 0,64W < 1W$$

Mirando en detalle la curva rosa, la cual es directamente proporcional a la corriente consumida por el prototipo, se ven picos significativos cada 180ms y pequeños valles cada 10ms. Los pequeños valles son causados por una baja en la tensión de alimentación del bus CAN, debido a la corriente consumida por el nodo transmisor. Los picos de corriente se deben a que cada 18 mensajes CAN, se llena un buffer interno que almacena los mensajes CAN recibidos, y los escribe a la tarjeta.

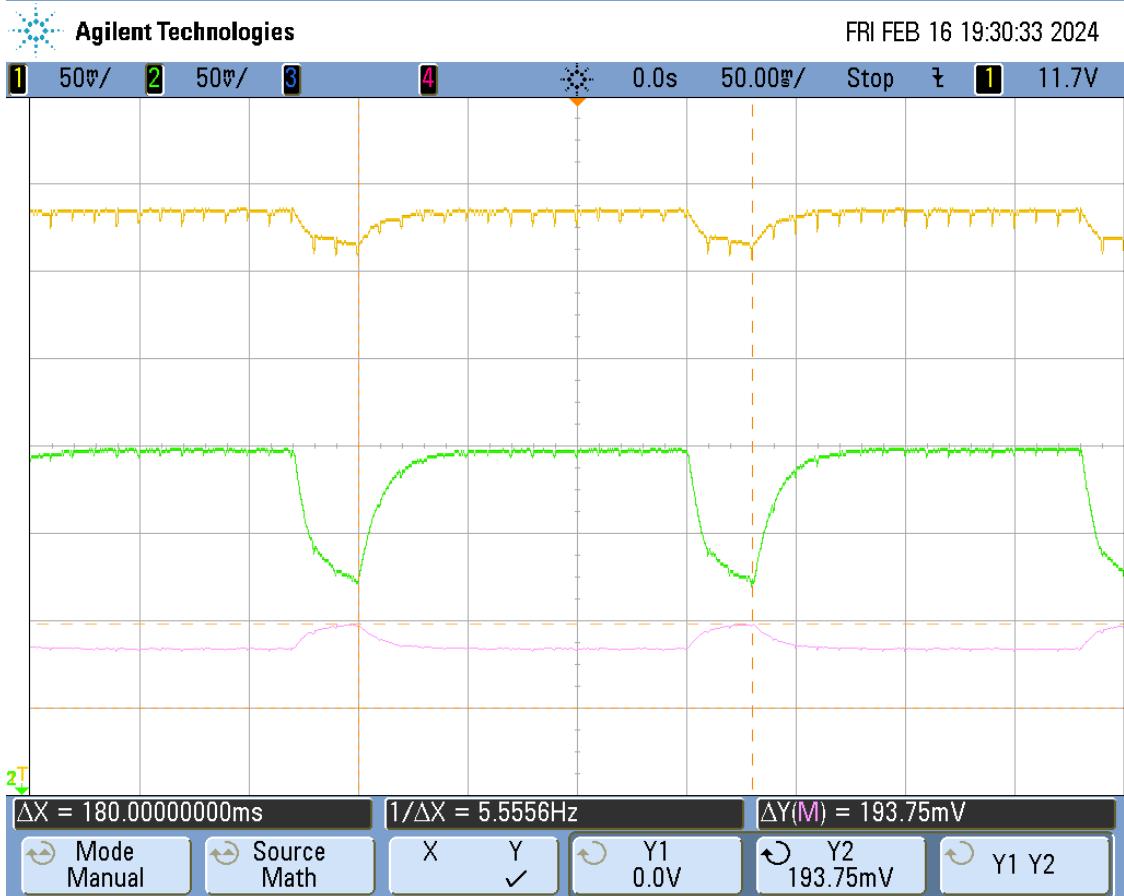
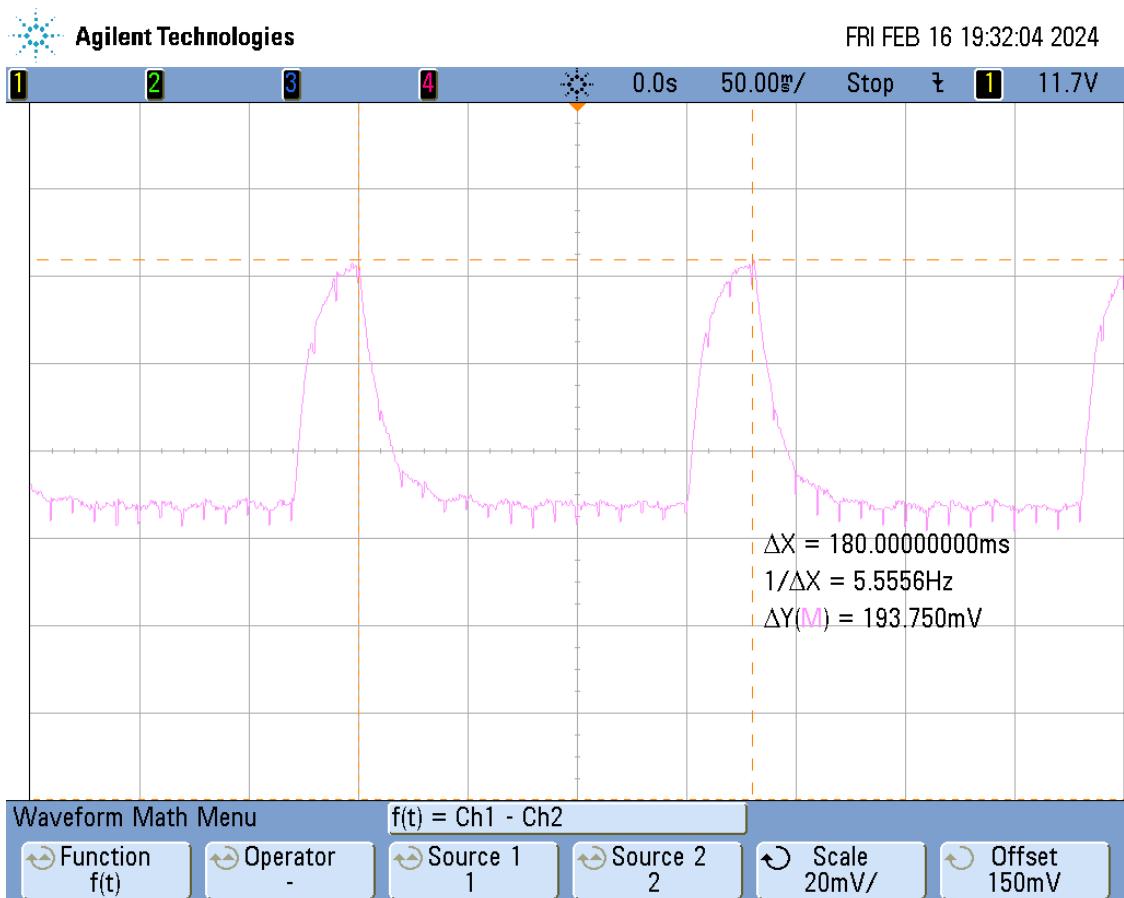


Figura 13.13: Medición de tensión en resistencia de shunt durante la recepción de mensajes CAN para calcular la corriente en uso normal.

### 13.3.3. Resultados de pruebas de integración prototipo - karting

El prototipo capturó mensajes del karting exitosamente. La figura 13.14 muestra una serie de mensajes CAN con distintos IDs, y concluye con un mensaje de apagado seguro al deshabilitar la alimentación.

Year	Month	Day	Hour	Minute	Second	Milisecond	ID	Data Length	B7	B6	B5	B4	B3	B2	B1	B0
2024	02	16	17	43	17	184	107	6	00	00	00	00	00	00	00	00
2024	02	16	17	43	17	193	100	3	00	00	00	00	00	00	00	A3
2024	02	16	17	43	17	286	107	6	00	00	00	00	00	00	00	00
2024	02	16	17	43	17	296	100	3	00	00	00	00	00	00	00	A3
2024	02	16	17	43	17	389	107	6	00	00	00	00	00	00	01	00
2024	02	16	17	43	17	398	100	3	00	00	00	00	00	0A	00	A3
2024	02	16	17	43	17	491	107	6	00	00	00	00	00	00	03	00
2024	02	16	17	43	17	501	100	3	00	00	00	00	00	1E	00	A3
2024	02	16	17	43	17	593	107	6	00	00	00	00	00	00	00	00
2024	02	16	17	43	17	603	100	3	00	00	00	00	00	00	00	A3
2024	02	16	17	43	17	696	107	6	00	00	00	00	00	00	00	00
2024	02	16	17	43	17	705	100	3	00	00	00	00	00	00	00	A3
2024	02	16	17	43	17	799	107	6	00	00	00	00	00	00	00	00
2024	02	16	17	43	17	808	100	3	00	00	00	00	00	00	00	A3
2024	02	16	17	43	17	900	107	6	00	00	00	00	00	00	02	00
2024	02	16	17	43	17	910	100	3	00	00	00	00	00	14	00	A3
2024	02	16	17	43	17	999	107	6	00	00	00	00	00	00	09	00
2024	02	16	17	43	17	999	100	3	00	00	00	00	00	5C	00	A3
2024	02	16	17	43	18	082	107	6	00	00	00	00	00	00	06	00
2024	02	16	17	43	18	091	100	3	00	00	00	00	00	3D	00	A3
2024	02	16	17	43	18	183	107	6	00	00	00	00	00	00	08	00
2024	02	16	17	43	18	193	100	3	00	00	00	00	00	51	00	A3
▪																
▪																
▪																
2024	02	16	17	43	31	083	107	6	00	00	00	00	00	00	00	00
2024	02	16	17	43	31	092	100	3	00	00	00	00	00	00	00	A3
2024	02	16	17	43	31	186	107	6	00	00	00	00	00	00	00	00
2024	02	16	17	43	31	195	100	3	00	00	00	00	00	00	00	A3
2024	02	16	17	43	31	287	107	6	00	00	00	00	00	00	00	00
2024	02	16	17	43	31	297	100	3	00	00	00	00	00	00	00	A3
2024	02	16	17	43	31	390	107	6	00	00	00	00	00	00	00	00
2024	02	16	17	43	31	400	100	3	00	00	00	00	00	00	00	A3
2024	02	16	17	43	31	492	107	6	00	00	00	00	00	00	00	00
2024	02	16	17	43	31	502	100	3	00	00	00	00	00	00	00	A3
2024	02	16	17	43	31	594	107	6	00	00	00	00	00	00	00	00
2024	02	16	17	43	31	604	100	3	00	00	00	00	00	00	00	A3
2024	02	16	17	43	31	698	107	6	00	00	00	00	00	00	00	00
2024	02	16	17	43	31	707	100	3	00	00	00	00	00	00	00	A3
2024	02	16	17	43	31	799	107	6	00	00	00	00	00	00	00	00
2024	02	16	17	43	31	809	100	3	00	00	00	00	00	00	00	A3
2024	02	16	17	43	31	901	107	6	00	00	00	00	00	00	00	00
2024	02	16	17	43	31	912	100	3	00	00	00	00	00	00	00	A3
2024	02	16	17	43	31	999	107	6	00	00	00	00	00	00	00	00
2024	02	16	17	43	31	999	100	3	00	00	00	00	00	00	00	A3
2024	02	16	17	43	32	073 Safe shutdown triggered										

Figura 13.14: CSV con los datos de la prueba en el karting.

## 14. Estudios de confiabilidad de hardware y de software

### 14.1. Estudio de confiabilidad de hardware

La confiabilidad del hardware del CAN Data Logger se puede determinar empleando la norma MIL-HDBK-217F [3], este es un manual militar que proporciona pautas y métodos para analizar la confiabilidad de componentes y sistemas electrónicos. El manual ofrece modelos matemáticos y datos empíricos para estimar tasas de falla y tiempo medio entre fallas (MTBF por sus siglas en inglés, *Mean Time Between Failures*) para varios tipos de componentes electrónicos, como resistencias, capacitores, transistores, circuitos integrados y sistemas completos. Utilizando estas predicciones, se pueden tomar decisiones informadas sobre elecciones de diseño, identificar posibles puntos débiles en un sistema y evaluar la confiabilidad esperada a lo largo de la vida operativa de un equipo electrónico.

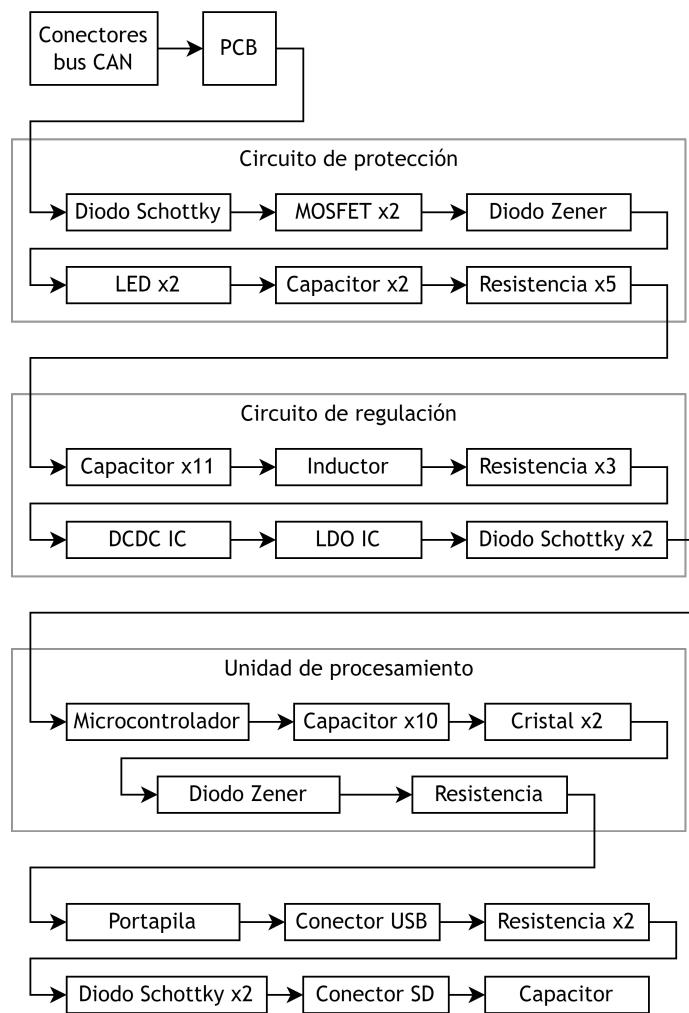


Figura 14.1: Diagrama en bloques de confiabilidad.

El primer paso del análisis de confiabilidad que se llevó a cabo consiste en determinar los elementos del sistema que pueden fallar y la relación entre ellos; los elementos que provocan una falla del sistema completo se configuran en serie mientras que si el sistema falla cuando hubiera una falla concurrente de varios elementos estos se configuran en paralelo. En la figura 14.1 se puede ver un diagrama en bloques de confiabilidad (RBD por sus siglas en inglés, *Reliability Block Diagram*). Un aspecto a destacar es que en el diseño inicial todos

los elementos del sistema se encuentran configurados en serie, dado que una falla en cualquiera de estos provoca el fallo del sistema completo. En el caso de encontrar puntos débiles o un MTBF muy bajo, la configuración del sistema se puede modificar para lograr una mayor robustez o redundancia.

Una vez confeccionado el diagrama en bloques de confiabilidad, se deben identificar las condiciones de operación de cada elemento, para luego calcular la tasa de falla de cada uno. La tasa de falla se define como una función  $\lambda(t)$  que depende del tiempo y presenta un valor esencialmente constante durante la mayor parte de la vida operativa del producto<sup>5</sup>.

La norma establece las siguientes funciones de tasa de fallas para distintos tipos de componentes:

- Microcontrolador:

$$\lambda_p = (C_1\pi_T + C_2\pi_E)\pi_Q\pi_L \text{ fallas}/10^6 \text{ horas}$$

- Diodo Schottky:

$$\lambda_p = \lambda_b\pi_T\pi_A\pi_R\pi_Q\pi_E \text{ fallas}/10^6 \text{ horas}$$

- Diodo Zener (voltage reference):

$$\lambda_p = \lambda_b\pi_T\pi_S\pi_C\pi_Q\pi_E \text{ fallas}/10^6 \text{ horas}$$

- MOSFET:

$$\lambda_p = \lambda_b\pi_T\pi_A\pi_Q\pi_E \text{ fallas}/10^6 \text{ horas}$$

- Resistencia:

$$\lambda_p = \lambda_b\pi_R\pi_Q\pi_E \text{ fallas}/10^6 \text{ horas}$$

- Capacitor cerámico:

$$\lambda_p = \lambda_b\pi_{CV}\pi_Q\pi_E \text{ fallas}/10^6 \text{ horas}$$

- Capacitor electrolítico:

$$\lambda_p = \lambda_b\pi_{CV}\pi_Q\pi_E \text{ fallas}/10^6 \text{ horas}$$

- Inductor:

$$\lambda_p = \lambda_b\pi_C\pi_Q\pi_E \text{ fallas}/10^6 \text{ horas}$$

- Conector:

$$\lambda_p = \lambda_b\pi_K\pi_P\pi_E \text{ fallas}/10^6 \text{ horas}$$

- PCB:

$$\lambda_p = \lambda_b [N_1\pi_C + N_2(\pi_C + 13)]\pi_Q\pi_E \text{ fallas}/10^6 \text{ horas}$$

- Cristal de cuarzo:

$$\lambda_p = \lambda_b\pi_Q\pi_E \text{ fallas}/10^6 \text{ horas}$$

- LED (emitter):

$$\lambda_p = \lambda_b\pi_T\pi_Q\pi_E \text{ fallas}/10^6 \text{ horas}$$

---

<sup>5</sup> La tasa de falla presenta una “curva de la bañera”, que se refiere a que para el inicio y el final de la vida útil del producto, el valor de  $\lambda(t)$  es significativamente más alto que el valor de estado estacionario.

Componente	$\lambda_b$	$\pi_Q$	$\pi_E$	$\pi_T$	$C_1$	$C_2$	$\pi_L$	$\pi_C$	$\pi_A$	$\pi_R$	$\pi_S$	$\pi_{CV}$
Microcontrolador	-	2	4	0.42	0.56	0.043	1	-	-	-	-	-
Diodo Schottky	0.027	2.5	5	2.1	-	-	-	-	1	1	-	-
Diodo Zener	0.002	8	9	3	-	-	-	1	-	-	1	-
MOSFET	0.012	5.5	9	2	-	-	-	-	0.7	-	-	-
Resistencia	0.00059	15	8	-	-	-	-	-	-	1	-	-
Capacitor cerámico	0.00072	10	9	-	-	-	-	-	-	-	-	1.6
Capacitor electrolítico	0.041	10	12	-	-	-	-	-	-	-	-	0.098
Inductor	0.00051	20	12	-	-	-	-	1	-	-	-	-
Cristal de cuarzo 32.768kHz	0.011	2.1	10	-	-	-	-	-	-	-	-	-
Cristal de cuarzo 16MHz	0.025	2.1	10	-	-	-	-	-	-	-	-	-
LED	0.00023	8	8	2.7	-	-	-	-	-	-	-	-

Tabla 14.1: Parámetros de confiabilidad de cada componente de acuerdo a la norma MIL-HDBK-217F.

Componente	$\lambda_b$	$\pi_Q$	$\pi_E$	$\pi_C$	$\pi_K$	$\pi_P$	$N_1$	$N_2$
Conector DE-9	0.0012	-	21	-	4	2.4	-	-
Conector SD	0.0012	-	21	-	4	2.9	-	-
Conector USB	0.0012	-	21	-	4	4.7	-	-
PCB	0.000041	2	7	1.6	-	-	0	76

Tabla 14.2: Parámetros de confiabilidad de cada conector y PCB de acuerdo a la norma MIL-HDBK-217F.

En las tablas 14.1 y 14.2 se muestran los valores de cada uno de los coeficientes necesarios para el cálculo de la tasa de fallas de cada componente. Los coeficientes se obtuvieron del manual de la norma MIL-HDBK-217F a partir de las condiciones de entorno y aplicación. A partir de los valores obtenidos se obtuvo la tasa media de fallas a partir de las ecuaciones mencionadas.

En la tabla 14.3 se muestran los resultados para cada componente. Los elementos con mayor tasa de fallas son el microcontrolador, los diodos schottky, los MOSFET, y el PCB, por lo tanto se deben tener en cuenta en el caso de modificar el diseño de forma de mejorar la confiabilidad del sistema. Los capacitores también presentan una tasa de fallas significativa, lo cual es relevante dado que el sistema cuenta con decenas de capacitores. Es posible mejorar la confiabilidad empleando capacitores (y otros componentes pasivos) con una mayor clasificación de confiabilidad, como por ejemplo AEC-Q200 [4].

A continuación, se calcula la confiabilidad del sistema total. En el caso de un sistema con sus elementos en serie, se puede obtener la tasa de fallos total mediante:

$$\lambda_S(t) = \sum \lambda_i(t)$$

Componente	$[\lambda_P \text{ fallas}]/10^6 \text{ horas}$
Microcontrolador	0.8144
Diodo Schottky	0.70875
Diodo Zener	0.432
MOSFET	0.8316
Resistencia	0.0708
Capacitor cerámico	0.10368
Capacitor electrolítico	0.48216
Inductor	0.1224
Cristal de cuarzo 32.768kHz	0.231
Cristal de cuarzo 16MHz	0.525
LED	0.039744
Conector DE-9	0.24192
Conector SD	0.29232
Conector USB	0.47376
PCB	0.6369104

Tabla 14.3: Tasa media de fallas de cada componente.

Luego, el MTBF se puede obtener mediante:

$$MTBF = \frac{1}{\lambda_S}$$

La tasa de fallos del sistema resulta 14,1323 fallas/ $10^6$  horas y el MTBF resulta 8,078 años. Esta duración es superior al tiempo esperado de vida del producto, por lo cual se determinó que no es necesario efectuar cambios de diseño para mejorar la confiabilidad.

## 14.2. Estudio de confiabilidad de software

La confiabilidad de un software puede predecirse utilizando modelos de confiabilidad. Los modelos de confiabilidad del software emplean información sobre la cantidad de errores depurados durante el desarrollo de un programa, esta información se utiliza para caracterizar los parámetros del modelo que luego se pueden usar para predecir la cantidad de fallas u alguna otra medida de confiabilidad en el futuro. La confiabilidad del software se define como la probabilidad de que un software dado funcione durante un período de tiempo especificado, sin errores de software, cuando se utiliza dentro de los límites de diseño [5].

En el presente trabajo se empleó el modelo propuesto por Shooman [6, 7] para llevar a cabo el análisis de confiabilidad. Este modelo se basa en las siguientes suposiciones:

1. El número total de instrucciones de lenguaje de máquina en el software es constante.
2. El número de errores al inicio del proceso de pruebas de integración es constante y decrece a medida que los errores son corregidos. No se introducen nuevos errores durante el proceso de pruebas.

3. La diferencia en los errores presentes al inicio y los errores corregidos acumulados representa los errores residuales.

A partir de estas suposiciones se obtiene la siguiente relación:

$$e_r(x) = e_t - e_c(x)$$

Donde:

$x$  es el tiempo de depuración desde el inicio del periodo de integración.

$e_t$  es la cantidad de errores al inicio del proceso normalizado por la cantidad total de instrucciones de lenguaje de máquina ( $I_T$ ).

$e_c(x)$  es la cantidad de errores corregidos en el instante  $x$  normalizado por  $I_T$ .

$e_r(x)$  es la cantidad de errores residuales en el instante  $x$  normalizado por  $I_T$ .

Asumiendo que la tasa de fallas es proporcional a la cantidad de errores residuales se obtiene:

$$\lambda_S(t) = K_S \cdot e_r(x)$$

Donde:

$t$  es el tiempo de operación del sistema.

$K_S$  es una constante de proporcionalidad.

$\lambda_S(t)$  es la tasa de fallas en el instante  $t$ .

Para obtener estimadores de los coeficientes de este modelo, durante el proceso de depuración del código se llevó un registro de los errores encontrados y corregidos mediante el software de control de versiones git<sup>6</sup> utilizando un repositorio remoto de Github<sup>7</sup>. Así, es posible contabilizar los errores y calcular las tasas de fallas parciales  $\lambda_i$ , los estimadores de errores totales  $\widehat{e_{ti}}$ , y los estimadores de la constante de proporcionalidad  $\widehat{K_{Si}}$ . Se utilizan las siguientes definiciones:

$$\lambda_i = \frac{e_t(t_i) - e_t(t_{i-1})}{t_i - t_{i-1}}$$

$$\widehat{e_{ti}} = \frac{\frac{\lambda_i}{\lambda_{i-1}} e_t(t_{i-1}) - e_t(t_i)}{\frac{\lambda_i}{\lambda_{i-1}} - 1}$$

$$\widehat{K_{Si}} = \frac{\lambda_i I_T}{\widehat{e_{ti}} - e_t(t_i)}$$

En la tabla 14.4 se pueden ver los valores de  $\lambda_i$ ,  $\widehat{e_{ti}}$  y  $\widehat{K_{Si}}$  calculados a partir de los parámetros  $e_t$ ,  $e_c$  y  $e_r$ . El proceso de depuración se llevó a cabo a lo largo de 8 semanas.

Los valores de  $\widehat{e_{ti}}$  y  $\widehat{K_{Si}}$  se promedian para obtener  $\widehat{K_S} = 6,77$  y  $\widehat{K_S} = 93,49$ . Luego, el valor de  $\lambda_{tot}$  se obtiene como  $\widehat{K_S}/I_T \cdot e_r(8) = 0,00434$  (donde el valor de  $I_T$  es de 21520). El valor de MTBF se obtiene como  $1/\lambda_{tot} = 230,18$  horas = 9,59 días. Este valor excede la duración de uso continuo del producto, por lo tanto la confiabilidad alcanzada en el software se adecúa a las necesidades del producto.

---

<sup>6</sup><https://git-scm.com/>

<sup>7</sup><https://github.com/>

Semanas	$e_t$	$e_c$	$e_r$	$\lambda_i$	$\widehat{e_{ti}}$	$\widehat{K_{Si}}$
1	10	5	5	0.0595	-	-
2	21	8	13	0.0655	-100.00	-11.65
3	24	18	6	0.0179	25.13	341.59
4	25	22	3	0.0060	25.50	256.19
5	30	25	5	0.0298	23.75	-102.48
6	36	31	5	0.0357	0.00	-21.35
7	38	34	4	0.0119	39.00	256.19
8	42	41	1	0.0238	34.00	-64.05

Tabla 14.4: Parámetros del modelo de Shooman de confiabilidad de software.

## 15. Conclusiones

### 15.1. Objetivos Alcanzados

- El prototipo logró satisfacer y superar las expectativas del cliente.
- Los módulos de software pudieron ser probados independientemente entre ellos. Esto fue posible gracias al alto grado de modularidad del código desarrollado.
- Solamente una prueba de módulos de hardware en el prototipo necesitó una modificación en el circuito esquemático, los demás aprobaron sin inconvenientes.
- El prototipo superó todas las pruebas de validación que se propuso inicialmente que sean ejecutadas.
- Gracias a la validación exitosa del prototipo, se realizó una prueba de integración entre el prototipo y el karting que fue también exitosa, sin requerir modificaciones especiales.

### 15.2. Recomendaciones para futuros diseños

- Para futuros prototipos, incluir un pequeño circuito de protección para poder conectar el programador al mismo que tiempo que se alimenta la placa por otro método. De esta forma, se evita tener que desconectar y conectar el cable de programación constantemente, y además se puede depurar el software mientras la placa está alimentada.
- Elegir capacitores de mantenimiento de otra tecnología de tamaño más reducido para poder reducir el tamaño de la carcasa.
- Ofrecer al cliente la posibilidad de utilizar otros medios de almacenamiento que no sean una tarjeta SD. Por ejemplo, una tarjeta MicroSD cuya ventaja es que permitiría reducir el tamaño final del producto. Otra opción es utilizar una memoria no volátil no removible, lo cual además de ahorrar espacio sería menos propenso a fallas por vibraciones.

## 16. Referencias

- [1] International Organization for Standardization, *ISO 11898-2:2016, Road vehicles – Controller area network (CAN) – Part 2: High-speed medium access unit*, 2016.
- [2] International Organization for Standardization, *ISO 16750-3:2007, Road vehicles – Environmental conditions and testing for electrical and electronic equipment – Part 3: Mechanical loads*, 2007.
- [3] U.S. Department of Defense, *MIL-HDBK-217F: Reliability Prediction of Electronic Equipment*, Technical Report, U.S. Department of Defense, 1991.
- [4] Automotive Electronics Council, *AEC-Q200: Stress Test Qualification for Passive Components*, Revision D, Automotive Electronics Council, 2019.
- [5] B. S. Dhillon and C. Singh, *Engineering Reliability: New Techniques and Applications*, Wiley, New York, 1981, Chapter 5.
- [6] M. L. Shooman, *Operational Testing and Software Reliability Estimation During Program Development, 1973 IEEE Symposium on Computer Software Reliability*, IEEE, New York, 1973, pp. 51-57.
- [7] M. L. Shooman, *Software Reliability: Measurement and Models, Proceedings of the 1975 Annual Reliability and Maintainability Symposium*, IEEE, New York, 1975.
- [8] Altium, *Pi Filter Designs for Power Supplies*, <https://resources.altium.com/p/pi-filter-designs-power-supplies>.
- [9] FTDI, *AN\_146 USB Hardware Design Guidelines for FTDI ICs*, Application Note, FTDI.
- [10] NXP Semiconductors, *K64 Sub-Family Reference Manual*, 2021, <https://www.mouser.com/datasheet/2/813/K64P144M120SF5RM-1074828.pdf>.
- [11] NXP Semiconductors, *Kinetis K64F Sub-Family Data Sheet*, 2016, <https://www.nxp.com/docs/en/data-sheet/K64P144M120SF5.pdf>.
- [12] NXP Semiconductors, *FRDM-K64F User Guide*, 2016, [https://os.mbed.com/media/uploads/GregC/frdm-k64f\\_ug\\_rev0.1.pdf](https://os.mbed.com/media/uploads/GregC/frdm-k64f_ug_rev0.1.pdf).
- [13] NXP Semiconductors, *FRDM-K64F Schematic*, 2016, <https://web.eece.maine.edu/~zhu/book/FRDM-K64F/FRDM-K64F-SCH-E4.pdf>.
- [14] NXP Semiconductors, *AN10911 - SD(HC)-memory card and MMC interface conditioning*, <https://www.mouser.com/catalog/specsheets/an10911.pdf>.
- [15] Siemens, *On-Board Communication via CAN without Transceiver*, [https://www.mikrocontroller.net/attachment/28831/siemens\\_AP2921.pdf](https://www.mikrocontroller.net/attachment/28831/siemens_AP2921.pdf).
- [16] International Electrotechnical Commission, *IEC 60529: Degrees of protection provided by enclosures (IP Code)*, 2001, <https://texa-co.ir/wp-content/uploads/2017/04/IEC-60529.pdf>.
- [17] CAN Data Logger source code, *Release v0.1.0*, GitHub, [https://github.com/PabloSML/CAN\\_Data\\_Logger/releases/tag/v0.1.0](https://github.com/PabloSML/CAN_Data_Logger/releases/tag/v0.1.0).

## 17. Anexos

### 17.1. Esquemáticos

En las figuras 17.1, 17.2 y 17.3 se observan los esquemáticos desarrollados para la construcción del prototipo.

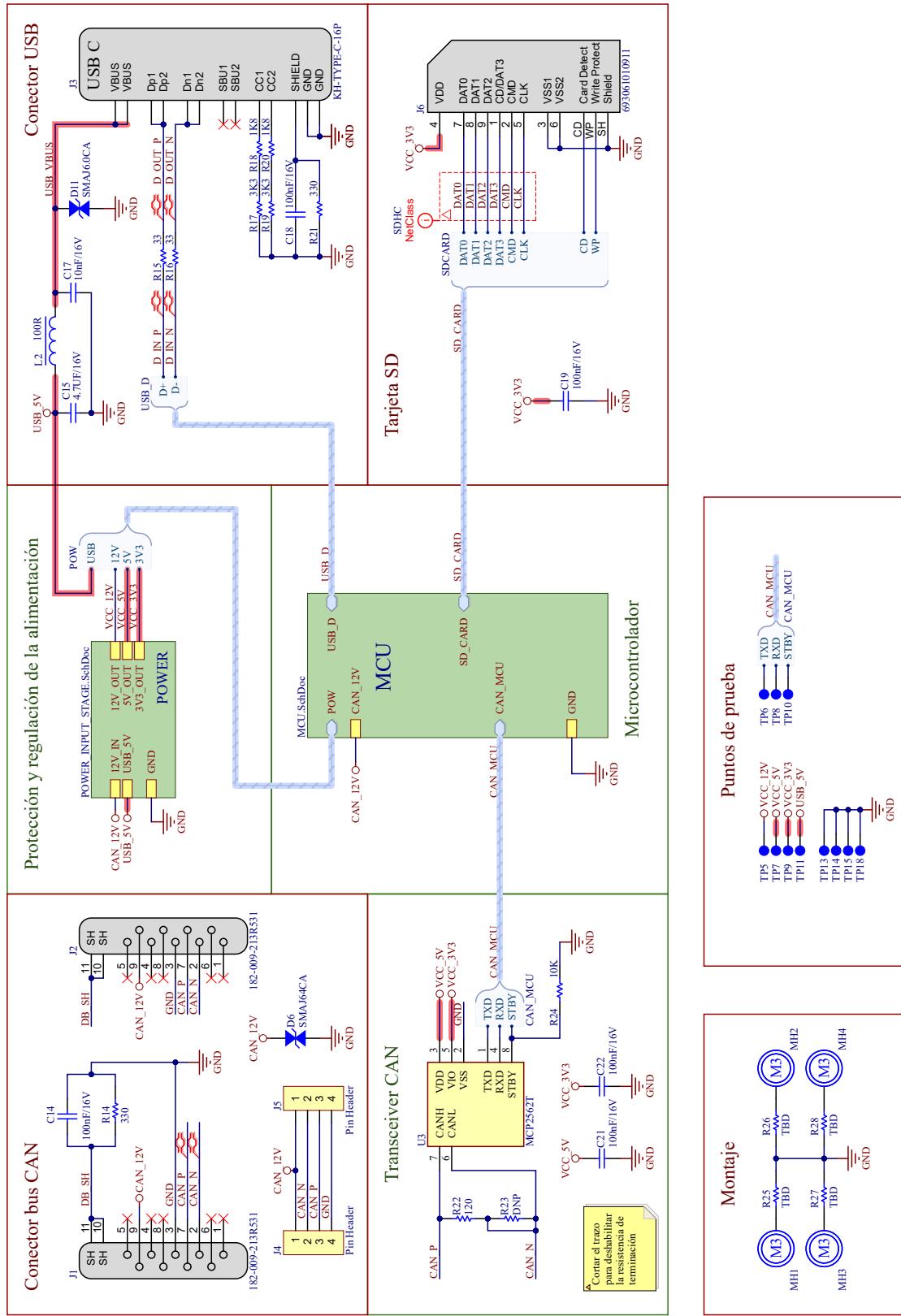
### 17.2. Planos de PCB

En la figura 17.4 se pueden observar los planos del PCB diseñado. Las capas corresponden a la Top Layer (esquina superior izquierda), Bottom Layer (esquina superior derecha), GND Plane 1 (esquina inferior izquierda), y GND Plane 2 (esquina inferior derecha).

### 17.3. Listado de Partes y Componentes (BOM)

En las tablas 17.1 y 17.2 se puede ver el listado de partes y componentes (BOM por sus siglas en inglés, *Bill Of Materials*) que corresponden a la manufactura del PCB.

En la tabla 17.3 se puede observar el listado de partes y componentes que corresponden a la manufactura del prototipo.



## Power Input Stage

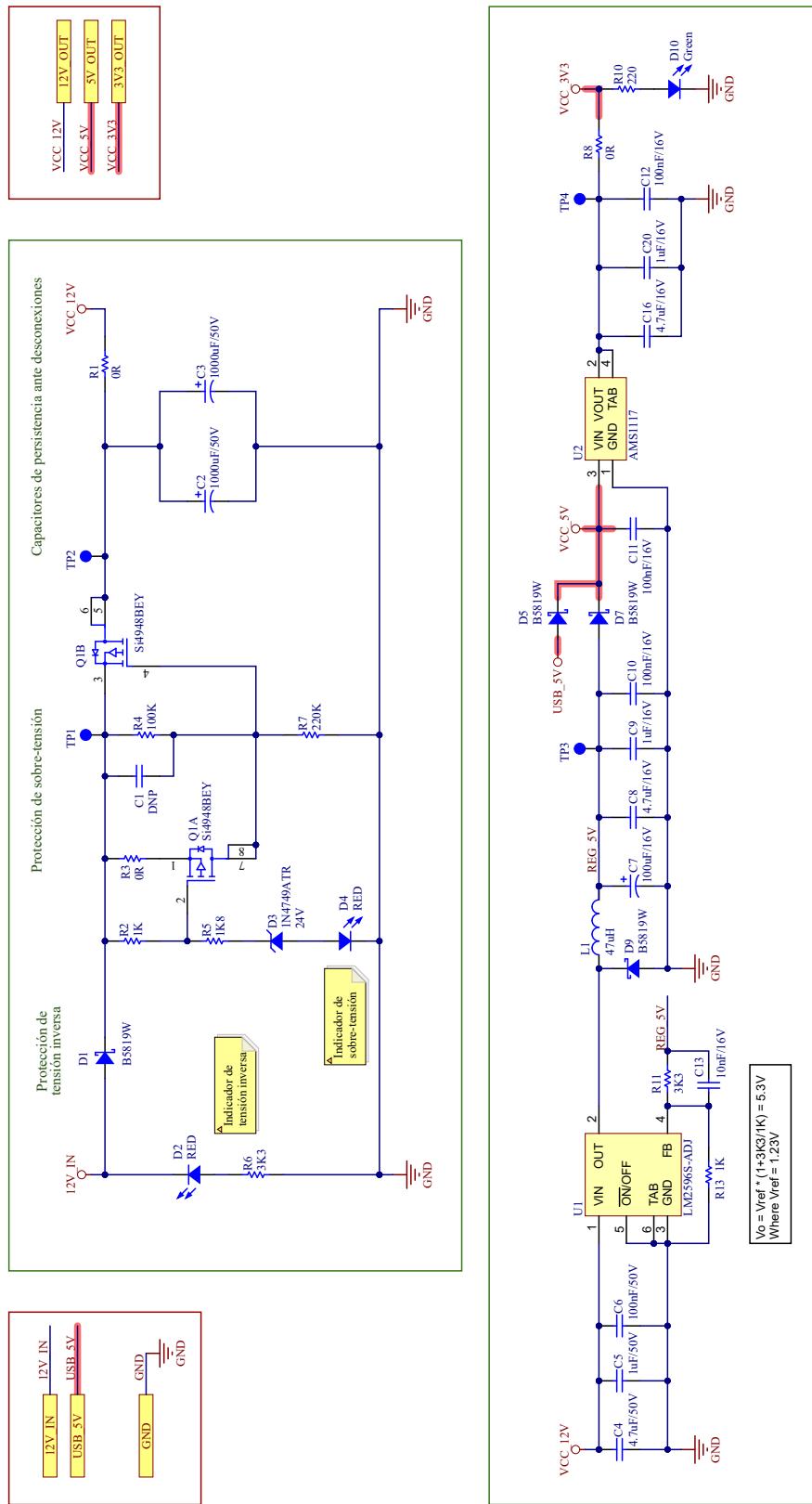


Figura 17.2: Esquemático de la etapa de entrada de alimentación.

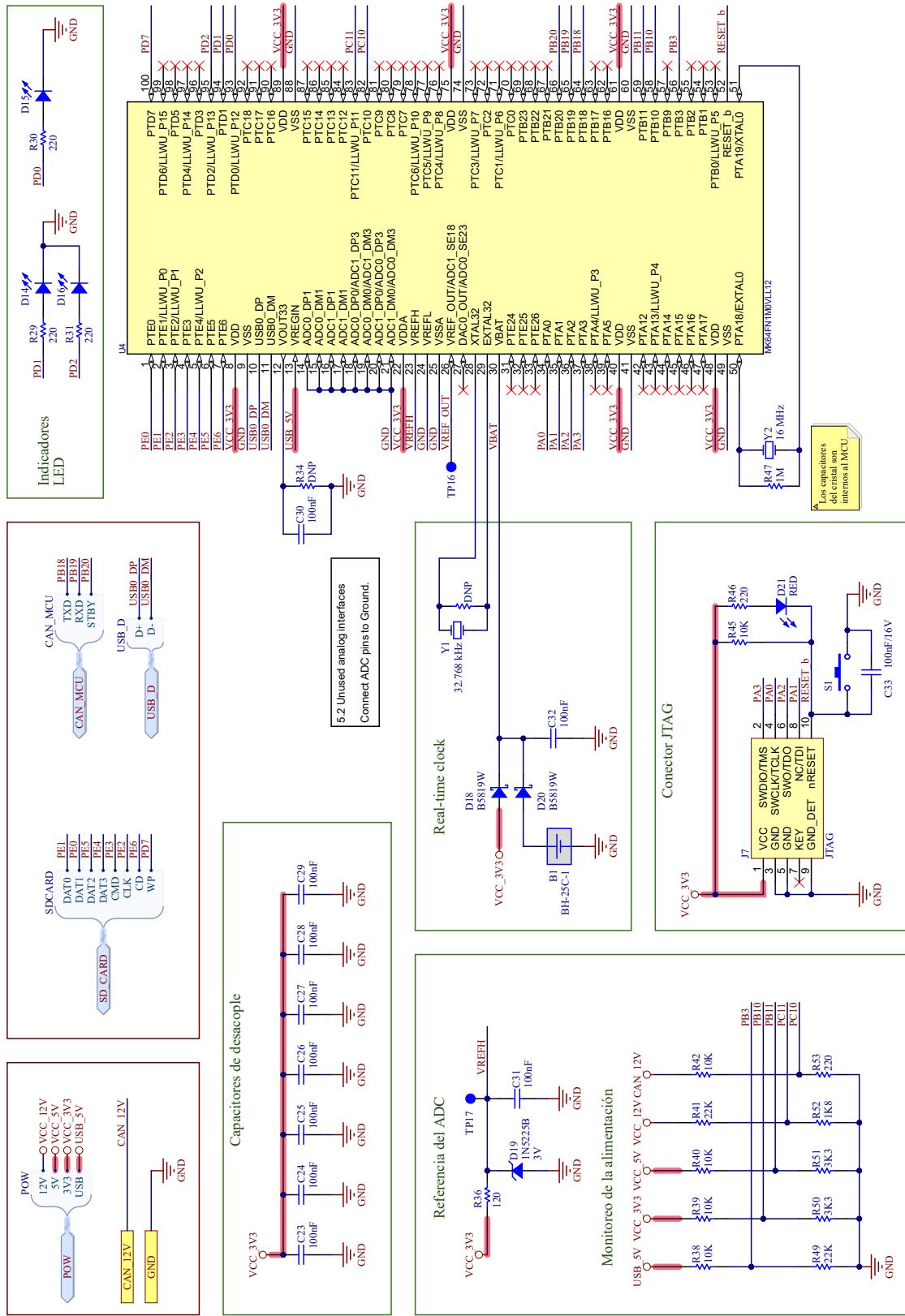


Figura 17.3: Esquemático del microcontrolador y circuitos periféricos.

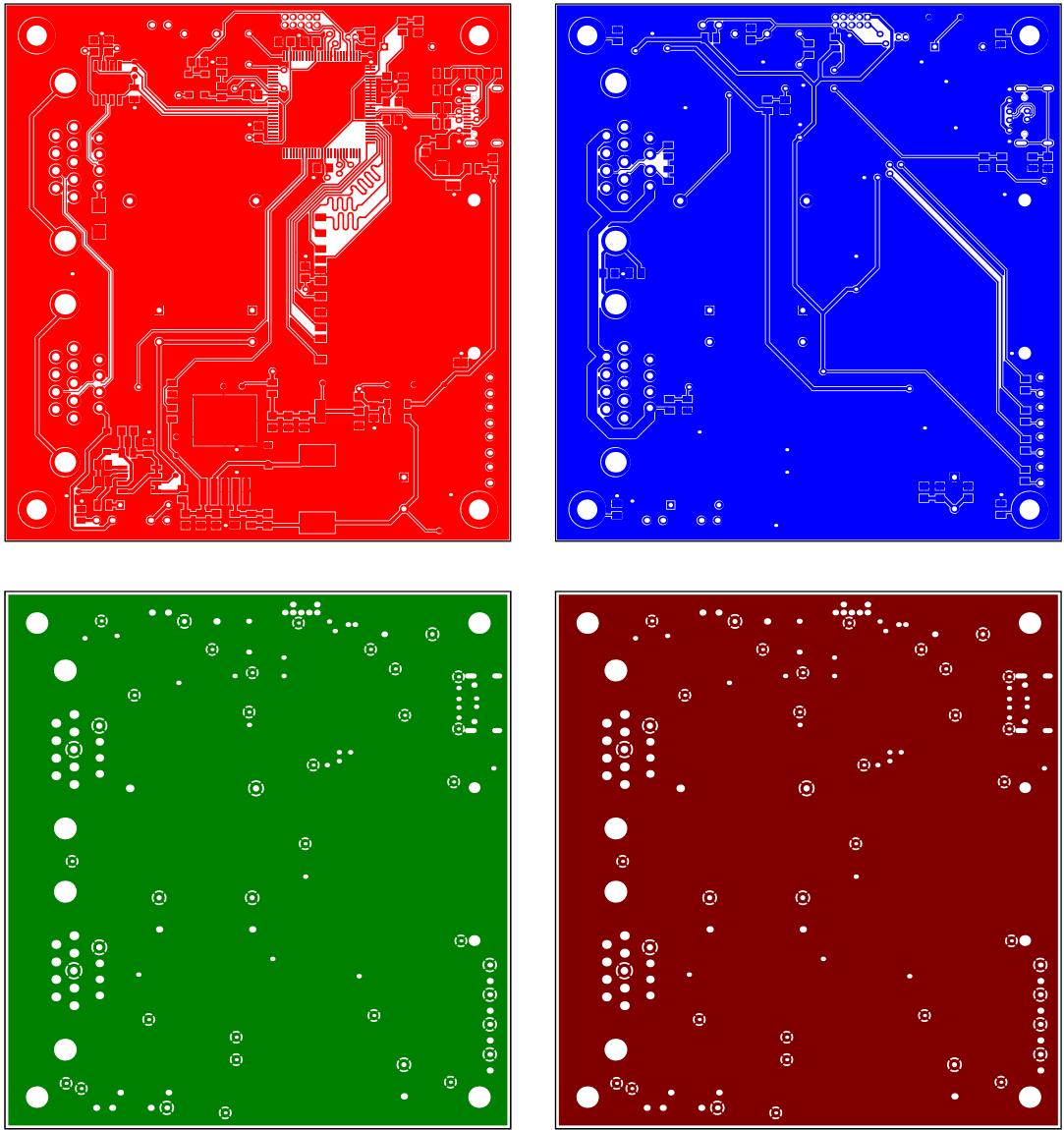


Figura 17.4: Planos del PCB diseñado. Las capas corresponden a la Top Layer (esquina superior izquierda), Bottom Layer (esquina superior derecha), GND Plane 1 (esquina inferior izquierda), y GND Plane 2 (esquina inferior derecha).

Cantidad	Valor	Empaquetado	Descripción	Designador
2	1000uF/50V	Radial 12.5mm	Polarized Capacitor	C2, C3
1	100uF/16V	Radial 12.5mm	Polarized Capacitor	C7
1	4.7uF/50V	0805	SMD 0805 Capacitor	C4
3	4.7uF/16V	0805	SMD 0805 Capacitor	C8, C15, C16
1	1uF/50V	0805	SMD 0805 Capacitor	C5
2	1uF/16V	0805	SMD 0805 Capacitor	C9, C20
1	100nF/50V	0805	SMD 0805 Capacitor	C6
19	100nF/16V	0805	SMD 0805 Capacitor	C10, C11, C12, C14, C18, C19, C21, C22, C33, C23, C24, C25, C26, C27, C28, C29, C30, C31, C32
2	10nF/16V	0805	SMD 0805 Capacitor	C13, C17
3	0R	0805	SMD 0805 Resistor	R1, R3, R8
2	1K	0805	SMD 0805 Resistor	R2, R13
1	100K	0805	SMD 0805 Resistor	R4
4	1K8	0805	SMD 0805 Resistor	R5, R18, R20, R52
6	3K3	0805	SMD 0805 Resistor	R6, R11, R17, R19, R50, R51
2	22K	0805	SMD 0805 Resistor	R41, R49
1	220K	0805	SMD 0805 Resistor	R7
6	220	0805	SMD 0805 Resistor	R10, R29, R30, R31, R46, R53
2	330	0805	SMD 0805 Resistor	R14, R21
2	33	0805	SMD 0805 Resistor	R15, R16
2	120	0805	SMD 0805 Resistor	R22, R36
7	10K	0805	SMD 0805 Resistor	R24, R34, R38, R39, R40, R42, R45
1	100R	0805	Ferrite Bead	L2
1	47uH	DRQ470	Inductor	L1

Tabla 17.1: Listado de partes y componentes (parte 1 de 2).

Cantidad	Valor	Empaquetado	Descripción	Designador
6	Red	TH 3mm LED	Red 3mm LED	D2, D4, D14, D15, D16, D21
1	Green	TH 3mm LED	Green 3mm LED	D10
6	B5819W	SOD-123	Schottky Diode	D1, D5, D7, D9, D18, D20
1	1N4749ATR	DO-35	Zener Diode 24V	D3
1	SMAJ64CA	DO-214AC	TVS Diode	D6
1	SMAJ6.0CA	DO-214AC	TVS Diode	D11
1	1N5225B	DO-35	Zener Diode 3V	D19
1	BH-25C-1	CR2032 Battery Holder	CR2032 Battery Holder	B1
2	182-009-213R531	DE-9 - Female	DE-9, Female, Right Angle	J1, J2
1	KH-TYPE-C-16P	USB4105-GF-A-060	USB-C Connector SMT 12 Pin	J3
2	Pin Header 1x4 (2.54mm)	HEADER-4	Pin Header 4-pos	J4, J5
1	693061010911	SD Card Connector	SD Card Reader	J6
1	Pin Header 2x5 (1.27mm)	JTAG-10PIN	JTAG header	J7
1	Si4948BEY	8-SOIC	MOSFET 2P-CH 60V 2.4A	Q1
1	Tact Switch 6mm 2-pin	6mm tactile switch	Tactile switch	S1
1	LM2596S-ADJ	TO-263-5	Step-Down Voltage Regulator	U1
1	AMS1117	SOT-223	1A LDO Voltage Regulator	U2
1	MCP2562T	8-SOIC	High-Speed CAN Transceiver	U3
1	MK64FN1M0VLL12	QFP50P1600X1600X 160-100N	ARM Cortex®-M4- based Microcontroller	U4
1	32.768 kHz	Radial 3x8mm	Crystal Oscillator	Y1
1	16 MHz	HS-49S TH	Crystal Oscillator	Y2
1	PCB CANDLE	PCB	PCB del prototipo	PCB1

Tabla 17.2: Listado de partes y componentes (parte 2 de 2).

Cantidad	Componente
1	CANDLE Board
1	Carcasa inferior
1	Carcasa superior
4	Tornillo, diámetro de 1/8”, longitud de 2”
4	Tuerca, rosca de 1/8”

Tabla 17.3: Listado de partes y componentes para el ensamblaje del prototipo.

#### 17.4. Códigos de Software

En [17] se encuentra disponible el código fuente de la versión v0.1.0 del software implementado para el prototipo, con el cual las pruebas de validación finales fueron realizadas.