Trabajo Fin de Grado
Grado en Ingeniería Electrónica, Robótica y Mecatrónica

Telemetría para un monoplaza de Formula Student

Autor: Rubén Militello González
Tutor: Ramón González Carvajal

Dpto. Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2022

Trabajo Fin de Grado
Grado en Ingeniería Electrónica, Robótica y Mecatrónica

Telemetría para un monoplaza de Formula Student

Autor:
Rubén Militello González

Tutor:
Ramón González Carvajal
Catedrático de Universidad

Dpto. de Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2022

Proyecto Fin de Carrera: Telemetría para un monoplaza de Formula Student

Autor: Rubén Militello González

Tutor: Ramón González Carvajal

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:
Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

El secretario del Tribunal

AGRADECIMIENTOS

Este trabajo de fin de grado simboliza una etapa de trabajo y sacrificio. Sin estas personas no hubiera sido posible lograr el cierre de esta etapa.

En primer lugar, agradecer el apoyo y la motivación que siempre he recibido por parte de mi familia.

En segundo lugar, agradecer a mis compañeros del equipo ARUS por esas noches interminables y esos momentos único vividos haciendo de la Formula Student una etapa increíble y única.

En tercer lugar, a mis amigos por el apoyo y la confianza siempre mostrada.

Por último, agradecer a todos los profesores involucrados en mi formación. Especialmente a Ramón González Carvajal por darme el apoyo con el desarrollo de este trabajo de fin de grado.

Rubén Militello González

Sevilla, 2022

RESUMEN

La Formula Student, también conocida como SAE Formula Student, es una competición automovilística entre estudiantes de universidades de todo el mundo donde cada equipo universitario diseña y construye un vehículo monoplaza con el que posteriormente compite. La Universidad de Sevilla participa en dicha competición a través del equipo ARUS Andalucía Racing Team en el desarrollo de un monoplaza de combustión y eléctrico.

En este contexto durante la temporada 2021/2022 se propone como objetivo el diseño propio de una telemetría para ambos monoplazas. Para ello se plantea el diseño de una placa de circuitos impresos con capacidad de poder transmitir toda la información necesaria en tiempo real de los monoplazas.

Dicha placa se fundamenta básicamente en la transmisión de datos a través de un bróker MQTT empleando un módulo con conectividad 4G LTE. Dicho modem será controlado a través de comandos AT los cuales serán transmitidos por un microcontrolador 'máster'. En este caso se han empleado como microcontrolador ESP32-WROOM-32D y como modem SIM7600E-H del fabricante SimCom. Podríamos destacar otras funcionalidades de la placa como puede ser el envío/recepción de tramas de bus CAN, lecturas analógicas y digitales y la activación de diversos sistemas que componen los monoplazas.

Todo el diseño hardware se ha desarrollado empleando Altium Designer (esquemáticos y layout) mientras que para el desarrollo software se ha empleado Visual Studio Code. Como bróker hemos empleado EMQX conocido como el bróker MQTT más escalable para aplicaciones IoT, tal y como procede en este caso.

Para visualizar esos datos en tiempo real se han desarrollado diferentes interfaces graficas paras diferentes plataformas. Podemos destacar el desarrollo de una interfaz gráfica para Windows empleando Visual Studio y una aplicación para Android empleando el lenguaje de programación 'Flutter' en Visual Studio Code.

Todo esto se ha llevado a cabo teniendo en cuenta el impacto económico que supone para el equipo. He de destacar el patrocinio de distribuidores de PCBs, componentes electrónicos, diferentes empresas del sector e incluso el patrocinio del Departamento de Ingeniería Electrónica sin el cual no hubiera sido posible poder dar forma a la telemetría empleada en ambos monoplazas durante la temporada 2021/2022.

Como conclusión podemos destacar que el resultado final ha sido satisfactorio y de gran ayuda para el equipo. Con esta telemetría hemos podido validar aquellos parámetros necesarios para poder conseguir un monoplaza fiable y competitivo por lo que ha supuesto un gran avance técnico dentro del equipo.

ABSTRACT

Formula Student, also known as SAE Formula Student, is an automotive competition among students from universities around the world, where each university team designs and builds a single-seater vehicle to compete with. The University of Seville participates in this competition through the ARUS Andalucía Racing Team in the development of both combustion and electric single-seater vehicles.

In this context, for the 2021/2022 season, the goal is set to design a telemetry system for both single seaters. To achieve this, the design of a printed circuit board is proposed, capable of transmitting all necessary real-time information from the single seaters. This board is primarily based on data transmission through an MQTT broker using a 4G LTE connectivity module. This modem will be controlled through AT commands, which will be transmitted by a 'master' microcontroller. In this case, the ESP32-WROOM-32D microcontroller and the SIM7600E-H modem from SimCom have been used. Other functionalities of the board can include sending/receiving CAN bus frames, analog and digital readings, and the activation of various systems that make up the single seaters.

The entire hardware design has been developed using Altium Designer (schematics and layout), while Visual Studio Code has been used for software development. As the broker, EMQX, known as the most scalable MQTT broker for IoT applications, has been used, as is appropriate in this case.

To visualize this real-time data, different graphical interfaces have been developed for various platforms. Notable among these are a graphical interface for Windows using Visual Studio, and an Android application using the 'Flutter' programming language in Visual Studio Code.

All of this has been accomplished while considering the economic impact on the team. It's worth highlighting the sponsorship from PCB distributors, electronic components suppliers, various companies in the sector, and even the sponsorship from the Department of Electronic Engineering, without which it would not have been possible to develop the telemetry used in both single seaters during the 2021/2022 season.

In conclusion, it can be highlighted that the result has been satisfactory and greatly beneficial for the team. With this telemetry system, it has been possible to validate the necessary parameters to achieve a reliable and competitive single seater, representing a significant technical advancement within the team.

... -translation by google-

ÍNDICE

Índice de Ilustraciones

NOTACIÓN

PCB	Placa de Circuito Impreso (Printed Circuit Board)
BMS	En casi todos los puntos
CAN	Red de Área de Controlador (Controller Area Network)
UART	Transmisor y Receptor Universal Asíncrono (Universal Asynchronous Receiver-Transmitter)
SMD	Montaje en Superficie
DAC	Convertidor Digital-Analógico (Digital-to-Analog Converter)
ADC	Convertidor Analógico-Digital (Analog-to-Digital Converter)
GND	Tierra (Ground)
VCC	Voltaje de Alimentación (Voltage Common Collector)
CAN	Red de Área de Controlador (Controller Area Network)
USB	Bus Universal en Serie (Universal Serial Bus)
PWM	Modulación de Ancho de Pulso (Pulse Width Modulation)
I2C	Interfaz de Circuitos Integrados (Inter-Integrated Circuit)
ESD	Descarga Electroestática (Electrostatic Discharge)
MOSFET	Transistor de Efecto de Campo de Óxido Metálico (Metal-Oxide-Semiconductor Field-Effect Transistor)
LTE	Long-Term Evolution (Evolución a Largo Plazo)
SIM	Módulo de Identidad del Abonado (Subscriber Identity Module)
QoS	Calidad de Servicio (Quality of Service)

INTRODUCCIÓN

La Formula Student, también conocida como SAE Formula Student, es una competición automovilística entre estudiantes de universidades de todo el mundo donde cada equipo universitario diseña y construye un vehículo monoplaza con el que posteriormente compite.

Ilustración 1: Participantes en Formula Student Germany 2022

Se trata de una competición de diseño por lo que no simplemente se valora el comportamiento dinámico del monoplaza sobre la pista. Los principales aspectos que se tienen en cuenta en la competición son: el diseño del monoplaza, validación, impacto medioambiental y viabilidad del proyecto.

La Universidad de Sevilla participó en dicha competición durante la temporada 2021/2022 tanto con el monoplaza de combustión con el monoplaza eléctrico. Este trabajo se centra en el diseño de la telemetría que ha ido integrada en ambos monoplazas durante dicha temporada. Se trata de un diseño válido para ambos monoplazas en el cual solo difiere la versión software (cada monoplaza tiene un software adaptado).

Ilustración 2: Monoplaza del equipo ARUS compitiendo

Durante dicha temporada ambas monoplazas han asistidos a las diferentes competiciones organizadas en los mejores circuitos de Europa. En dichas competiciones hemos conseguido diversos trofeos, destacando el tercer puesto en la categoría de monoplaza eléctrico en Formula Student Italy.

Ilustración 3: ARUS celebrando el tercer puesto obtenido en Formula Student Italy

REQUISITOS

Durante la temporada 2021/2022, el equipo ARUS (Andalucía Racing Team) se enfrentó al desafío de desarrollar una telemetría avanzada para validar los sistemas y el comportamiento dinámico de sus monoplazas en el contexto competitivo del Formula Student. A diferencia de los vehículos comerciales, donde la prioridad suele ser el confort y la utilidad cotidiana, el equipo ARUS tenía como objetivo principal configurar y afinar el comportamiento dinámico específico de sus monoplazas para el máximo rendimiento en la pista.

La telemetría, en este contexto, emerge como una herramienta esencial para la recopilación, el análisis y la interpretación de una diversidad de datos en tiempo real provenientes de los sensores estratégicamente ubicados en el monoplaza. Estos sensores, que abarcan desde mediciones de rendimiento del motor hasta información sobre la suspensión, la aerodinámica, la temperatura y la aceleración, capturan un panorama completo del funcionamiento del vehículo y su interacción con la pista.

La singularidad de esta tarea radica en que ARUS no busca simplemente un comportamiento dinámico único y estandarizado para sus monoplazas, sino que aspira a diseñar y configurar cada vehículo de manera específica, adaptando sus propiedades de rendimiento según las características únicas de la pista y los objetivos competitivos. La telemetría, entonces, se convierte en una herramienta esencial que permite la implementación precisa y la posterior verificación de estas configuraciones.

Durante la temporada mencionada, se establecieron una serie de requisitos fundamentales que se alineaban con los objetivos del equipo ARUS:

1. Captura en Tiempo Real de Datos: La telemetría debía ser capaz de capturar una amplia variedad de datos en tiempo real, abarcando desde información sobre el rendimiento del motor y la velocidad hasta la posición de la suspensión y la dirección.
2. Transmisión y Almacenamiento Eficiente: Dada la magnitud de los datos generados, se requería una solución eficiente para la transmisión de datos hacia una unidad central, así como un método seguro y accesible para el almacenamiento de estos datos con fines de análisis.
3. Interfaz de Usuario Intuitiva: La telemetría debía contar con una interfaz de usuario amigable y fácil de usar, que permitiera a los ingenieros y pilotos acceder y entender los datos recopilados, lo que facilitaría la toma de decisiones basadas en información precisa.
4. Flexibilidad y Personalización: La telemetría debía ser lo suficientemente flexible como para permitir ajustes y configuraciones personalizadas para adaptarse a los diferentes tests que tendrán lugar durante la temporada.
5. Preparación para el Futuro: La telemetría debía ser diseñada con escalabilidad en mente, permitiendo la incorporación de nuevos funcionalidades y actualizaciones sin requerir una revisión completa del sistema.

La implementación exitosa de la telemetría no solo permitió una validación más precisa del comportamiento dinámico, sino que también facilitó un análisis profundo de los datos para impulsar mejoras continuas en el rendimiento y la competitividad del equipo en el entorno desafiante del Formula Student.

ESTRUCTURA DEL DOCUMENTO

Los "Agradecimientos" iniciales dan voz al reconocimiento sincero hacia aquellos que han contribuido a la realización del trabajo, estableciendo un tono de gratitud y colaboración.

El par de secciones "Resumen" y "Abstract" cumplen con la función de ofrecer un vistazo general tanto en el idioma principal como en inglés, lo que garantiza que la esencia del trabajo sea accesible a una audiencia global.

El "Índice" aporta una visión panorámica de la organización del contenido. Esto permite a los lectores anticipar la estructura del documento y ubicar fácilmente las secciones de su interés.

La sección de "Introducción" establece un terreno sólido al presentar el monoplaza de Formula Student y delinear los objetivos fundamentales del proyecto, generando un contexto sustancial para los capítulos siguientes.

La "Contextualización" ofrece una visión detallada del escenario en el que se desenvuelve el proyecto, ofreciendo perspicacia sobre el monoplaza y cómo sus subsistemas se integran de manera crucial.

Las secciones que abordan la "Arquitectura Hardware" y el "Layout PCB" penetran en las complejidades técnicas, revelando el diseño y las decisiones detrás de los componentes electrónicos y la disposición de la placa de circuito impreso.

La sección de "Arquitectura Software" desentraña la trama del desarrollo de software, mientras que la atención dedicada a las "Interfaces gráficas de usuario" coloca al lector en la posición de los usuarios finales, describiendo las interfaces visuales que dan vida a la interacción.

Los "Datos Registrados" proporcionan sustancia tangible al presentar resultados cuantitativos y cualitativos derivados de pruebas y simulaciones, respaldando así las afirmaciones anteriores.

Las "Conclusiones" ponen de relieve los frutos de la investigación y el trabajo, encapsulando los logros y hallazgos clave. Esta sección culmina en una perspectiva reflexiva que brinda un sentido de cierre al relato.

La inclusión de "Mejoras Futuras" muestra la visión a largo plazo del proyecto, dejando la puerta abierta a posibles desarrollos y mejoras posteriores.

1 CONTEXTO

1.1 Monoplaza de Formula Student

1.2 Distribución electrónica

Para poder contextualizar el diseño de un monoplaza de formula student es necesario tener una visión general tanto del monoplaza de combustión como el del coche eléctrico.

En este caso, vamos a definir los diferentes subsistemas electrónicos que componen a ambos monoplazas:

1.2.1 ART22-E

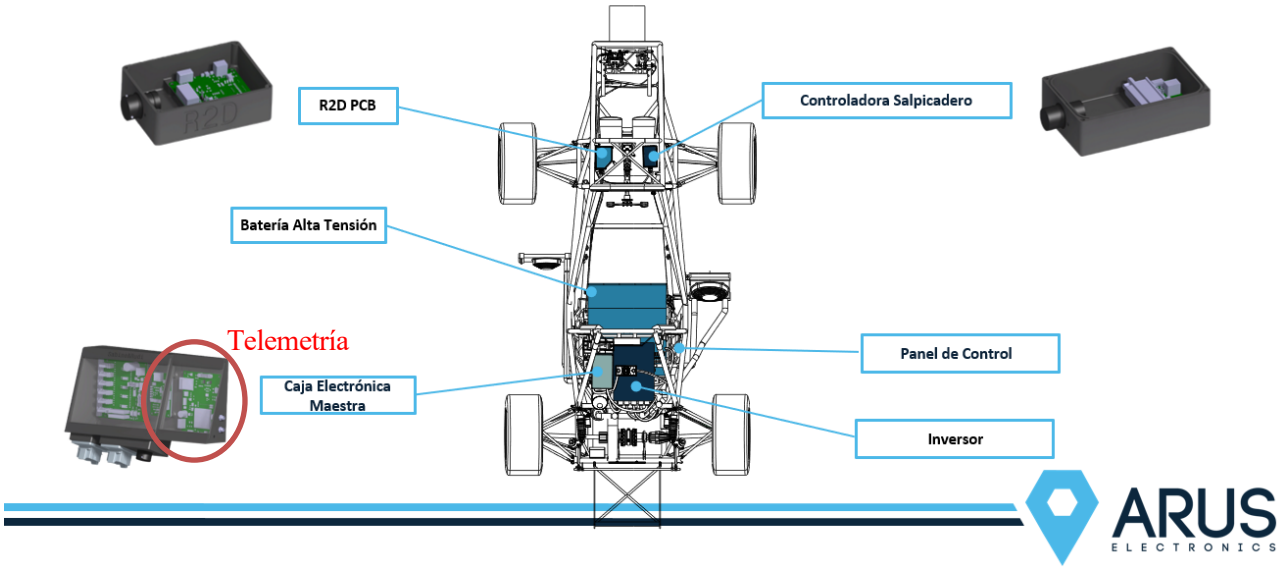


Ilustración 4: Visión general del monoplaza eléctrico

En la imagen anterior podemos ver la distribución de todos los subsistemas electrónicos integrados en el vehículo eléctrico. Podemos destacar los siguientes (los sistemas que conforman el tren de potencia se desarrollaran en los siguientes apartados):

R2D PCB: PCB de diseño propio la cual realiza el protocolo de 'Ready To Drive'. Dicho protocolo consiste el que el piloto debe pulsar el freno y una acción adicional para poder activar el sistema de alta tensión. Una vez haya realizado dicho protocolo un buzzer emite un sonido de activación durante 2s con el objetivo de notificar la activación del sistema de alta tensión.

Controladora Salpicadero: PCB encargada de leer por bus CAN los diferentes parámetros que se muestran por el salpicadero. Dichos parámetros mostrados son configurables en función de la información que necesite el piloto durante las pruebas dinámicas.

Panel de Control: Panel electrónico donde se tiene accesibilidad a las tensiones de la batería de alta y baja tensión, bus CAN, interruptores de activación de baja y alta tensión, referencia del chasis, conector de programación del inversor y del BMS.

Caja electrónica maestra: Se encuentran las PCBs de gestión de potencia, BSPD y telemetría.

PCB de gestión de potencia: diseño propio con el objetivo de distribuir la alimentación a los diferentes subsistemas del monoplaza, entre ellos se encuentran subsistemas de la imagen anterior y diversos sistemas

eléctricos como bomba de refrigeración, ventilador del radiador y luz de freno entre otros.

BSPD: Sistema de lógica no programable el cual debe abrir el circuito de shutdown si tiene lugar una frenada de más de 30 bar y si alguna de las siguientes condiciones: 'el pedal del acelerador está presionado más de un 25% o se está consumiendo más de 5kW de la batería de alta tensión'.

Telemetría: Sistema de diseño propio en el que se basa este Trabajo Fin de Grado. Descrito en los apartados previos y seguirá su desarrollo en los próximos apartados.

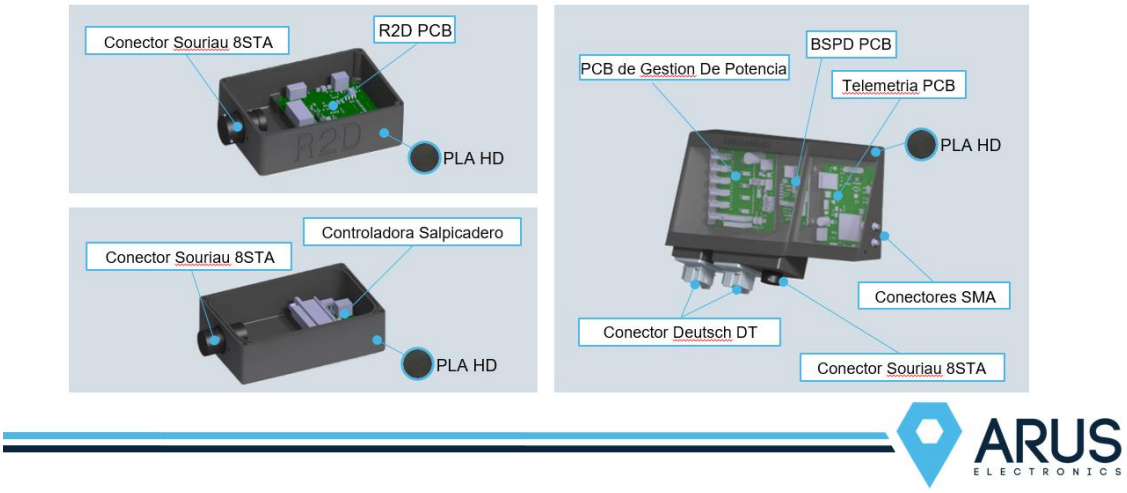


Ilustración 5: Resumen de las cajas electrónicas del monoplaza eléctrico

1.2.2 ART22-C

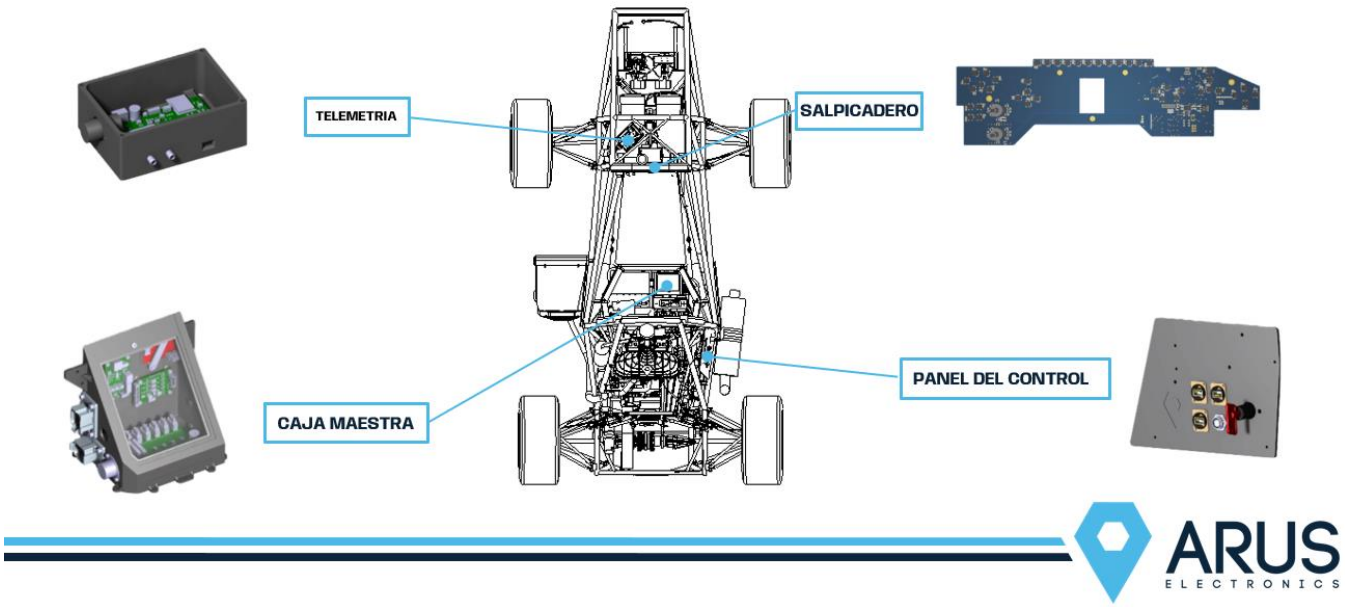


Ilustración 6: Visión general del monoplaza de combustión interna

En la imagen anterior podemos ver la distribución de todos los subsistemas electrónicos integrados en el vehículo eléctrico. Podemos destacar los siguientes (los sistemas que conforman el tren de potencia se desarrollaran en los siguientes apartados):

- Caja electrónica maestra: Se encuentran las PCBs de gestión de potencia, BSPD y control del embrague.
    - PCB de gestión de potencia: diseño propio con el objetivo de distribuir la alimentación a los diferentes subsistemas del monoplaza, entre ellos se encuentran subsistemas de la imagen

anterior y diversos sistemas eléctricos como bomba de refrigeración, ventilador del radiador y luz de freno entre otros.

- BSPD: Sistema de lógica no programable el cual debe abrir el circuito de shutdown si tiene lugar una frenada de más de 30 bar y el pedal del acelerador está presionado más de un 25%.
- Control del Embrague: Diseño propio para poder controlar la subida y bajada de marchas de nuestro monoplaza de combustión. Empleamos un actuador electromecánico para la subida y bajada de marcha y un servo motor para el acople/desacople del embrague.

- Salpicadero: PCB encargada de leer por bus CAN los diferentes parámetros que se muestran por el salpicadero. Dichos parámetros mostrados son configurables en función de la información que necesite el piloto durante las pruebas dinámicas.

- Panel de Control: Panel electrónico donde se tiene accesibilidad a las tensiones de la batería de baja tensión, bus CAN, interruptor de activación de baja tensión, conector de programación del embrague, PCB de gestión de potencia y conector de programación de la ECU del motor de combustión.

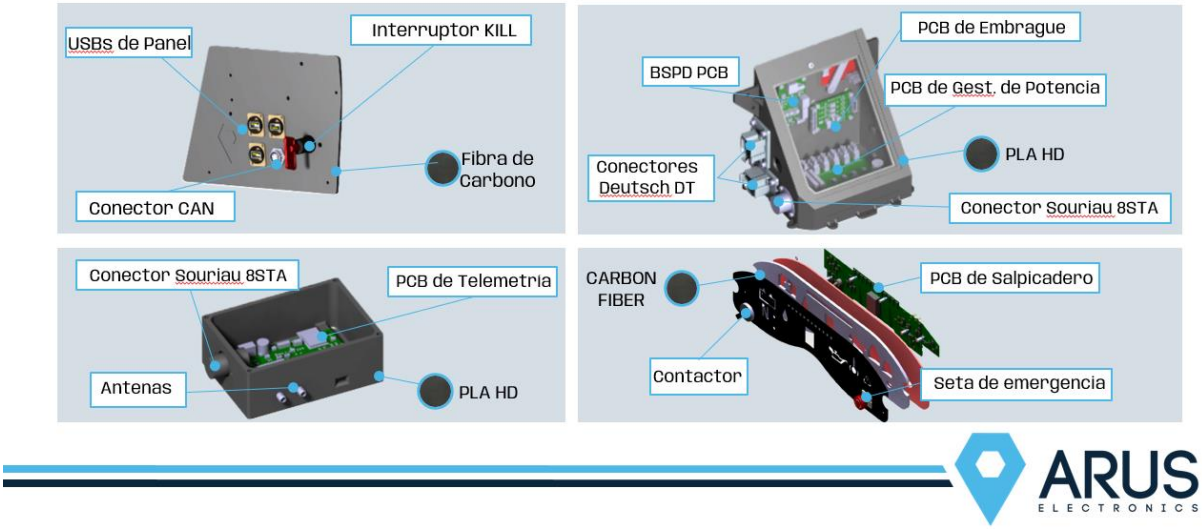


Ilustración 7: Resumen de las cajas electrónicas del monoplaza de combustión interna

1.3 Tren de Potencia

En este apartado nos centraremos en el tren de potencia del monoplaza eléctrico debido a que es el único de los dos que incluye sistemas electrónicos/eléctricos.

1.3.1 Inversor

Para el control del motor se emplea el inversor comercial Bamocar D3 de Unitek el cual puede emplearse hasta aplicaciones que requieran 150kW de potencia. Sin embargo, la normativa de Formula Student nos limita a 80kW. Además, este inversor posee bus CAN con el que se comunica con el resto subsistemas.

Ilustración 8: Bamocar D3 de Unitek

1.3.2 Batería de alta tensión

Este compuesto por una configuración de 108s2p de celdas de polímero de litio (LiFePo4). La tensión nominal de estas celdas es de 3,7V, con un mínimo de 3,3V y un máximo de 4,2V.

Teniendo en cuenta las características y la configuración anterior obtenemos una batería con las siguientes características:

$$\boldsymbol{V_{nominal} = 399,6V \mid V_{max} = 453,6V \mid V_{min} = 356,4V \mid P_{nominal} = 6,4Wh}$$

Para monitorizar todos los parámetros eléctricos la batería consta con un BMS de diseño propio. La normativa de Formula Student exige los parámetros mínimos que hay que monitorizar y con precisión debemos detectar fallos/errores en la batería de alta tensión. El BMS también gestiona la carga y balanceo de las diferentes celdas que componen nuestra batería. También gestiona los 3 relés que componen la batería de alta tensión.

La arquitectura del BMS es Maestra-Esclava debido a la gran cantidad de celdas que hay que monitorizar. Dicha arquitectura está compuesta por un maestro y nueves esclavos.

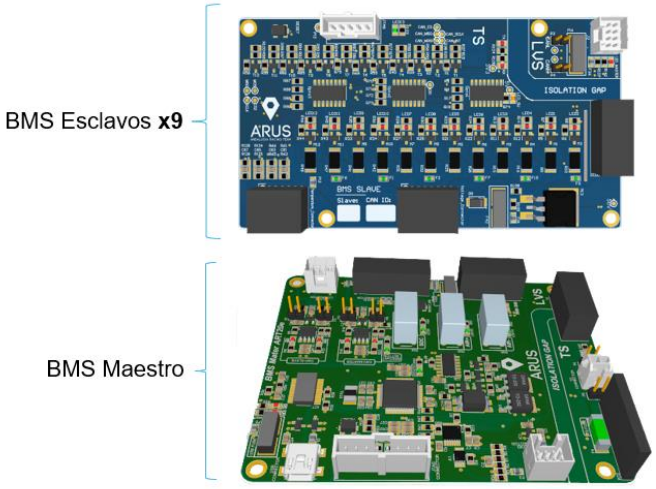


Ilustración 9: Arquitectura del BMS del ART22E

1.3.3 Motor trifásico

Ilustración 10: Emrax 228 Medio Voltaje (520 VDC)

Emrax 228 (motor síncrono de imanes permanentes y flujo axial) con las siguientes propiedades eléctricas:

Voltaje Máximo DC	520 V
Potencia Máxima	124 kW
Torque Máximo	230 Nm
Torque Nominal	Hasta 130 Nm
Eficiencia	92-98%

Tabla 1: Características principales del Emrax 228

1.4 Integración de subsistemas del monoplaza

Para la integración de los diferentes subsistemas que componen a los monoplazas empleamos diferentes protocolos de comunicación. Podríamos destacar principalmente tres de ellos, bus CAN, Universal Asynchronous Receiver-Transmitter (UART) y Universal Serial Bus (USB).

El protocolo principal de integración en nuestros monoplazas es el bus CAN (Controller Area Network). Es un protocolo de comunicaciones desarrollado por la firma alemana Robert Bosch GmbH para automoción, por lo que el uso en dicho sector está estandarizado y extendido.

1.4.1 Bus CAN (Control Area Network)

El bus CAN es un protocolo enfocado a la transmisión de mensajes. La información se descompone en los distintos mensajes a los que se le asigna un identificador y se encapsulan en tramas para su transmisión. Dentro del bus cada mensaje tiene un identificador (ID) asociado y cada nodo conectado al bus puede decidir si aceptar o rechazar dicho mensaje. [1]

El bus CAN es un protocolo diferencial por lo que la transmisión de mensajes se lleva a cabo a través de dos cables trenzados (La transmisión de señales diferencial proporciona protección frente a interferencias electromagnéticas). Las señales de estos cables se denomina CAN_H (CAN high) y CAN_L (CAN low)

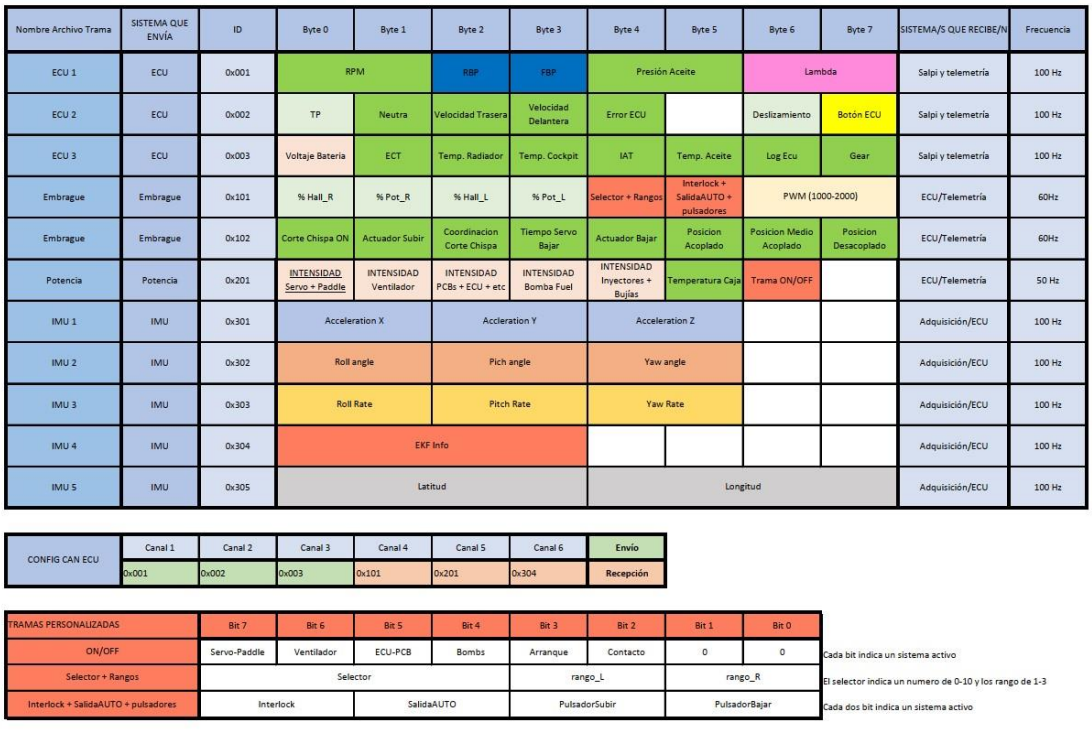
respectivamente. [1]

Los distintos nodos del bus CAN deben estar interconectados mediante un par de cables trenzados debido a que es un protocolo diferencial. El estándar CAN, a diferencia de otros estándares como el USB, no especifica ningún tipo de conector para el bus y por lo tanto cada aplicación puede tener un conector distinto. Sin embargo, hay varios formatos comúnmente aceptados como el conector D-sub de 9 pines, con la señal CAN_L en el pin 2 y la señal CAN_H en el pin 7. [1]

Podemos destacar dos tipos de bus CAN: [1]

- CAN de alta velocidad: también llamado CAN de alta velocidad, usa un único bus lineal terminado en cada extremo con sendas resistencias de 120 Ω (Es importante conseguir este valor de impedancia característico del BUS para evitar perturbaciones en la comunicación). Dicha configuración permite transmisiones de hasta 1 Mbit/s.
- Can de baja velocidad: Emplea un bus lineal, un bus en estrella o múltiples buses en estrella conectados por un bus lineal. El bus está terminado en cada nodo por una fracción de la resistencia de terminación total (próximo a 100 Ω, pero no inferior a 100 Ω). Esta configuración permite velocidades de hasta 125 kbit/s.

En nuestros monoplazas empleamos el bus CAN de alta velocidad con una velocidad configurada de 500kbit/s. En la siguiente imagen se muestra un ejemplo de la configuración de nuestro BUS en el monoplaza de combustión:



Nombre Archivo Trama	SISTEMA QUE ENVÍA	ID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	SISTEMA/S QUE RECIBE/N	Frecuencia
ECU 1	ECU	0x001	RPM		RBP	FBP	Presión Aceite		Lambda		Salpi y telemetría	100 Hz
ECU 2	ECU	0x002	TP	Neutra	Velocidad Trasera	Velocidad Delantera	Error ECU		Deslizamiento	Botón ECU	Salpi y telemetría	100 Hz
ECU 3	ECU	0x003	Voltaje Bateria	ECT	Temp. Radiador	Temp. Cockpit	IAT	Temp. Aceite	Log Ecu	Gear	Salpi y telemetría	100 Hz
Embrague	Embrague	0x101	% Hall_R	% Pot_R	% Hall_L	% Pot_L	Selector + Rangos	Interlock + SalidaAUTO + pulsadores	PWM (1000-2000)		ECU/Telemetría	60Hz
Embrague	Embrague	0x102	Corte Chispa ON	Actuador Subir	Coordinacion Corte Chispa	Tiempo Servo Bajar	Actuador Bajar	Posicion Acoplado	Posicion Medio Acoplado	Posicion Desacoplado	ECU/Telemetría	60Hz
Potencia	Potencia	0x201	INTENSIDAD Servo + Paddle	INTENSIDAD Ventilador	INTENSIDAD PCBs + ECU + etc	INTENSIDAD Bomba Fuel	INTENSIDAD Inyectores + Bujías	Temperatura Caja	Trama ON/OFF		ECU/Telemetría	50 Hz
IMU 1	IMU	0x301	Acceleration X		Accleration Y		Acceleration Z				Adquisición/ECU	100 Hz
IMU 2	IMU	0x302	Roll angle		Pich angle		Yaw angle				Adquisición/ECU	100 Hz
IMU 3	IMU	0x303	Roll Rate		Pitch Rate		Yaw Rate				Adquisición/ECU	100 Hz
IMU 4	IMU	0x304	EKF Info								Adquisición/ECU	100 Hz
IMU 5	IMU	0x305	Latitud				Longitud				Adquisición/ECU	100 Hz

CONFIG CAN ECU	Canal 1	Canal 2	Canal 3	Canal 4	Canal 5	Canal 6	Envío
	0x001	0x002	0x003	0x101	0x201	0x304	Recepción

TRAMAS PERSONALIZADAS	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
ON/OFF	Servo-Paddle	Ventilador	ECU-PCB	Bombs	Arranque	Contacto	0	0	Cada bit indica un sistema activo
Selector + Rangos	Selector				rango_L		rango_R		El selector indica un numero de 0-10 y los rango de 1-3
Interlock + SalidaAUTO + pulsadores	Interlock		SalidaAUTO		PulsadorSubir		PulsadorBajar		Cada dos bit indica un sistema activo

Ilustración 11: Configuración de nuestro bus CAN en el ART22C

En las siguientes imágenes podemos ver las dos topologías empleadas en ambos monoplazas:

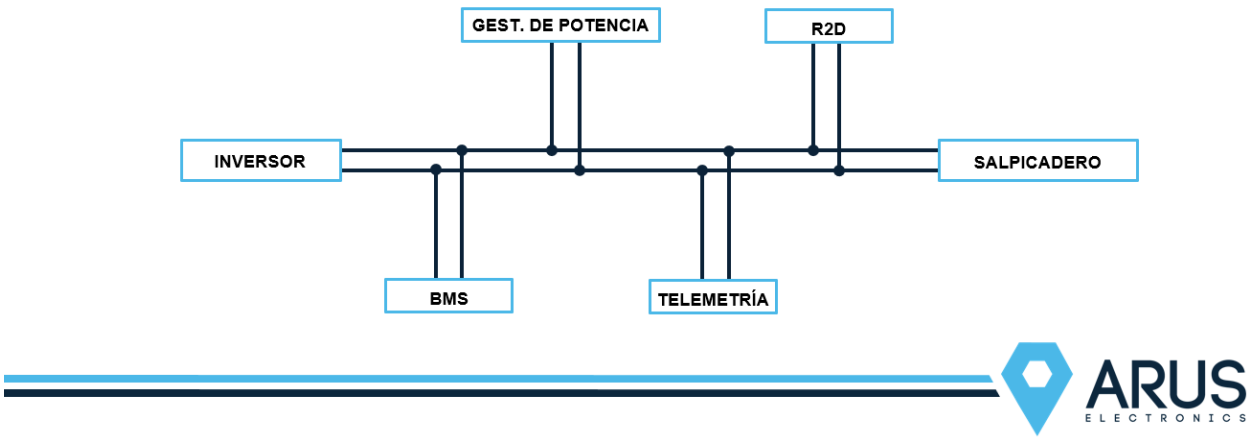


Ilustración 12: Topología del bus CAN del ART22E

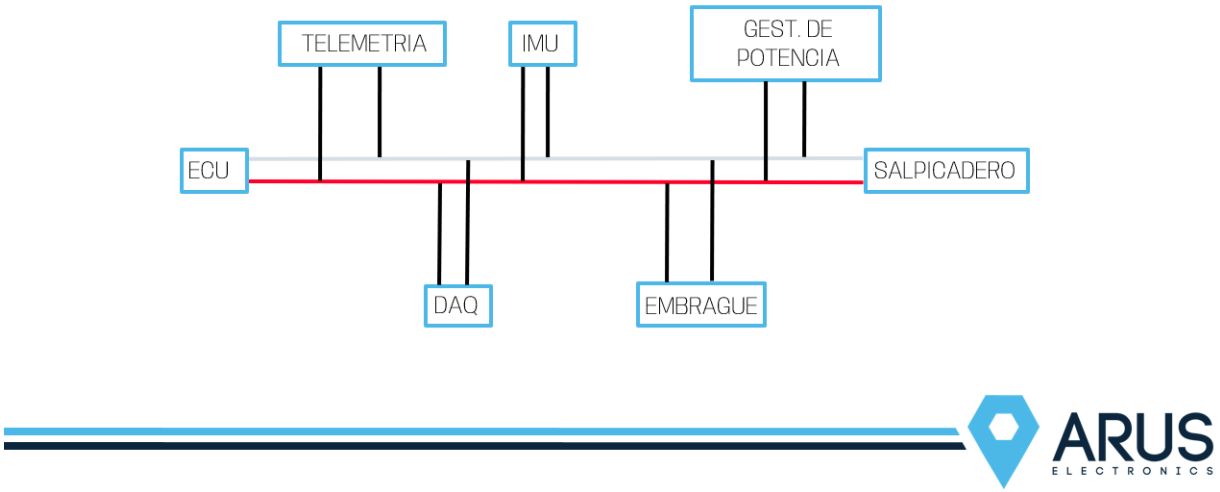


Ilustración 13: Topología del bus CAN del ART22C

1.4.2 Universal Asynchronous Receiver-Transmitter (UART)

UART es un protocolo simple de dos hilos para el intercambio de datos en serie. Asíncrono hace referencia a que los extremos de comunicación no comparten ningún reloj que los sincronicen. Por ello, para que el protocolo funcione ambos extremos deben estar configuradas a la misma velocidad de bits por segundo. Los bits de inicio y de parada se utilizan para indicar dónde empiezan y terminan los datos de usuario. Además, se puede utilizar un bit de paridad opcional para detectar errores de bit únicos. [2]

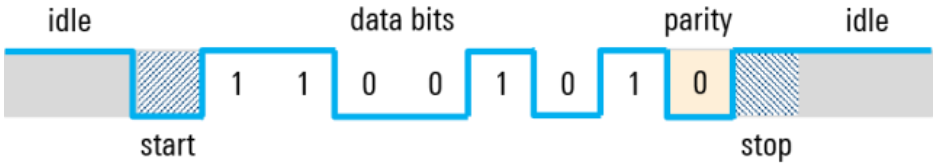


Ilustración 14: Formato de trama UART

Como en la mayoría de los sistemas digitales, un nivel de tensión alto significa un ‘uno lógico’, y un nivel de tensión bajo ‘un cero lógico’. Hay que destacar que cuando no se transmiten datos, la línea se mantiene en el estado alto por lo que podemos detectar con facilidad una línea o un transmisor averiado.

Este protocolo se emplea en ambos monoplazas, pero cabe destacar la utilidad en el BMS de diseño del monoplaza eléctrico. En dicho monoplaza poseemos un conector para poder visualizar todos los valores de nuestra batería de alta tensión (requisito de la normativa de Formula Student).

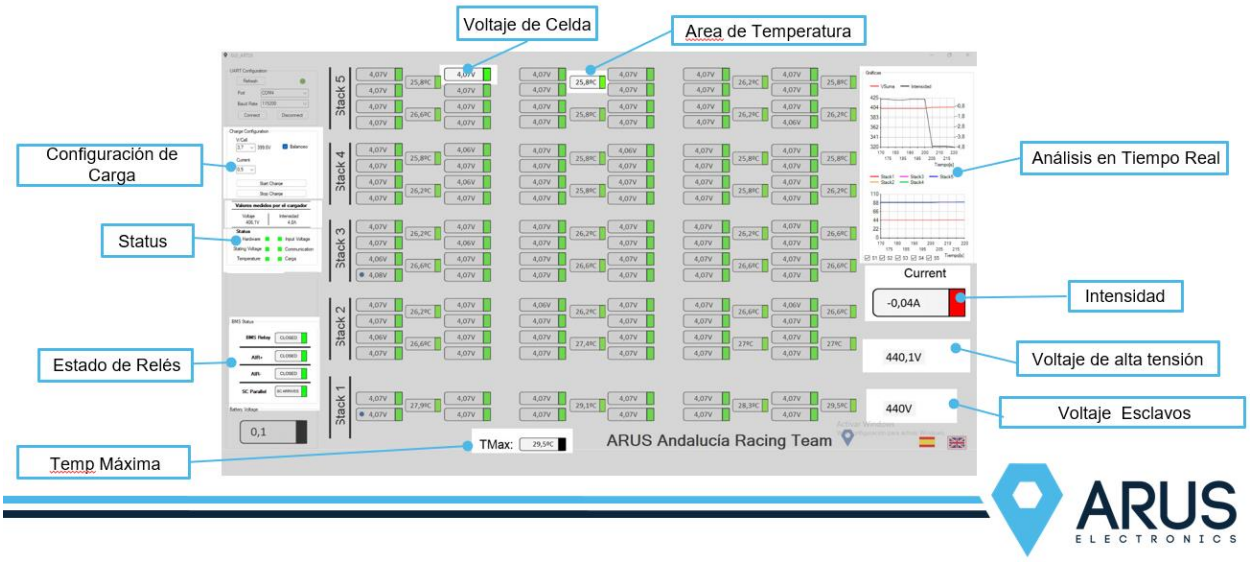


Ilustración 15: Interfaz gráfica del BMS del ART22E

2 ARQUITECTURA HARDWARE

2.1 Introducción

En primer lugar, se define a alto nivel la funcionalidad de cada bloque y la nomenclatura de las distintas señales tal y como muestra la siguiente ilustración:

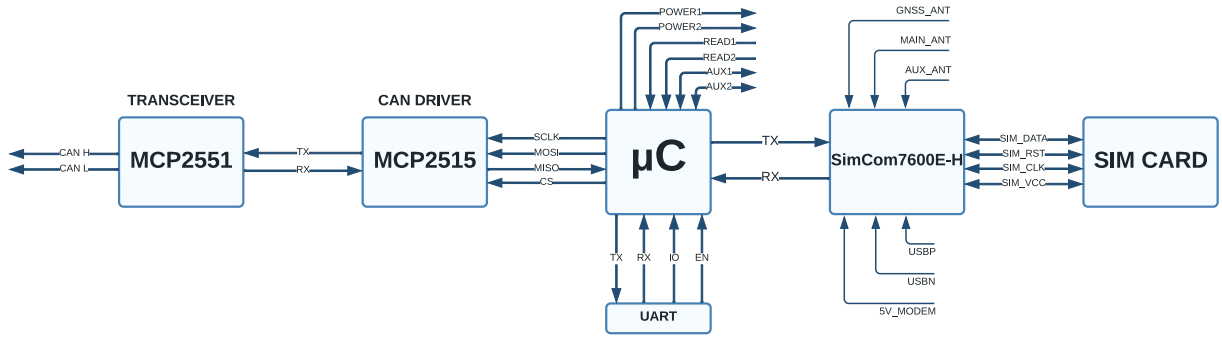


Ilustración 16: Arquitectura Hardware Telemetría

Teniendo en cuenta la imagen anterior podemos definir como tres funcionalidades principales que realiza la PCB:

- Envío/Recepción de tramas de bus CAN
- Envío/Recepción de datos a un bróker MQTT
- Funcionalidades del microcontrolador:
    - Activación de celdas de potencia
    - Activación de pines de salida/entrada
    - Integra la comunicación con los drivers CAN y de conectividad LTE

Para llevar a cabo dichas funcionalidades se han recurrido a diferentes componentes hardware:

- Envío/Recepción de tramas de bus CAN:
    - Driver bus CAN: MCP2515 de Microchip Technology es un controlador (CAN) que implementa la especificación CAN, versión 2.0B. Es capaz de transmitir y recibir en ambos estándares mencionados anteriormente. Posee dos máscaras de aceptación y seis filtros de aceptación que se utilizan para filtrar mensajes no deseados, por lo tanto, reduciendo la sobrecarga de los distintos subsistemas conectados al bus. El MCP2515 se comunica con nuestro microcontrolador a través del bus SPI (del inglés Serial Peripheral Interface). Dicho bus es un estándar hoy en día para controlar cualquier dispositivo electrónico que permita una comunicación síncrona.
    - Transceiver CAN: El MCP2551 es un transceptor CAN de alta velocidad que sirve como interfaz entre un controlador de protocolo CAN y el bus físico. El MCP2551 proporciona la capacidad de transmisión y recepción para el controlador de protocolo CAN (Funcionando a velocidades de hasta 1 Mb/s).
- Envío/Recepción de datos a un bróker MQTT:
    - Modulo Multibanda: SIM7600E-H de SIMCom Wireless Solutions. Módulo multibanda LTE-TDD/LTE-FDD/HSPA+ y GSM/GPRS/EDGE en un tipo SMT que admite LTE CAT1. Tiene

una fuerte capacidad de extensión con interfaces ricas que incluyen UART, USB2.0, I2C, GPIO. En nuestro caso la comunicación con el microcontrolador la integramos a través de UART empleando los diferentes comandos AT definidos por el fabricante. En este caso podríamos decir que el módulo es un esclavo de nuestro microcontrolador principal.

- Funcionalidades del microcontrolador:
    - ESP32-WROOM-32D: Microcontrolador de Expressif Systems que posee algunas de las siguientes características:
        - CPU: microprocesador de 32-bit Xtensa LX6 de doble núcleo
        - Memoria: 520 KiB SRAM
        - Wi-Fi: 802.11 b/g/n
        - Bluetooth: v4.2 BR/EDR y BLE
        - Interfaces periféricas:
        - 12-bit SAR ADC de hasta 18 canales
        - 2 × 8-bit DACs
        - 4 × SPI
        - 2 × interfaces I²C
        - 3 × UART

Teniendo en cuenta todas las funcionalidades mencionadas anteriormente, podemos decir que es un microcontrolador que cumple con todos los requisitos planteados. Además, otro motivo para la elección de este se debe a que posee conectividad Wi-Fi por lo que se pueden realizar pruebas de depuración tal como envío/recepción al bróker MQTT simplemente compartiendo conectividad desde un punto de acceso sin consumir datos de la tarjeta SIM.

2.2 Tarjeta Sim

El extracto del esquemático mostrado hace referencia al diseño de la interfaz USIM y del acondicionamiento necesario de las señales que componen dicho protocolo USIM. Respecto al acondicionamiento de las señales se ha empleado teniendo en cuenta el manual de diseño hardware del fabricante, donde se detalla la aplicación de la interfaz USIM. Se ha añadido una protección ESD tal y como indica dicha guía de diseño hardware.

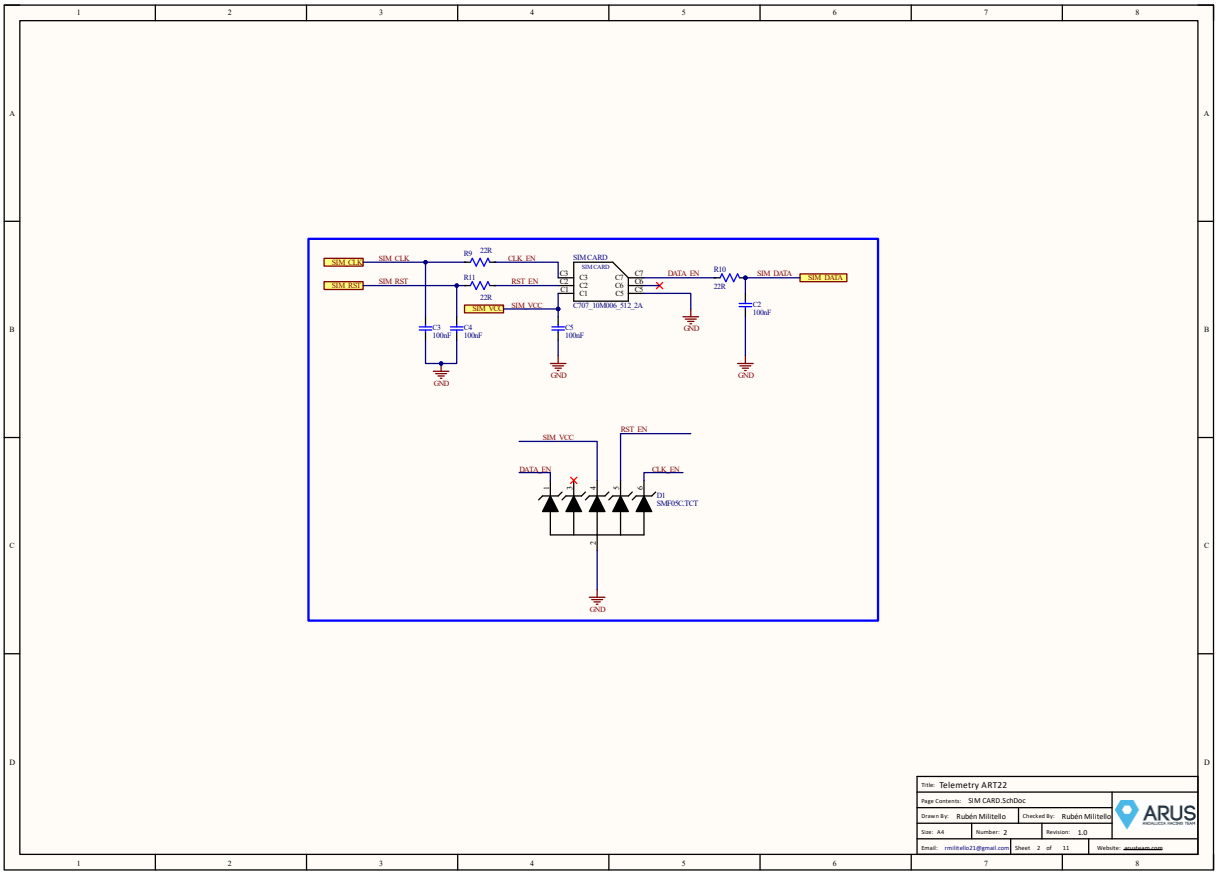


Ilustración 17: Esquemático del circuito de interfaz USIM

La serie de estándares ISO/IEC 7816 e ISO/IEC 7810 definen las diferentes características mecánicas y eléctricas de las tarjetas inteligentes, en este caso se trata de una tarjeta SIM (acrónimo en inglés de Subscriber Identity Module). En el apartado de la norma 7816-2 se especifican las dimensiones y ubicación de los contactos tal y como muestra la siguiente imagen:

Ilustración 18: Ubicación y contactos de una tarjeta de circuito integrado

Teniendo en cuenta la imagen anterior en nuestro diseño empleamos los contactos:

- C1: Contacto de alimentación de la interfaz USIM.
- C2: Contacto de reset, utilizada para reiniciar las comunicaciones de la tarjeta.
- C3: Proporciona a la tarjeta una señal de reloj, de la que se deriva la temporización de las

comunicaciones de datos.

- C5: Contacto de la referencia de voltaje.
- C7: Entrada o salida para datos seriales al circuito integrado dentro de la tarjeta.

Se emplean cinco contactos para la implementación de la interfaz ISA (parte 2 de la ISO / IEC 7816). La aplicación de nuevas interfaces (opcional), implica el uso de los contactos C4 y C8 (interfaz USB), así como un cambio en el uso del C6.

El conector elegido es el C707 10M006 522 2A de Amphenol MCP. Dicho conector es compatible con una Mini-SIM (especificada por el estándar: ISO/IEC 7810:2003, ID-000) la cual posee un largo de 25mm, ancho de 15mm y un espesor de 0.76mm.

Ilustración 19: Imagen del conector utilizado: C707 10M006 522 2A

2.3 Antenas de PCB

Para el diseño electrónico de las distintas antenas ubicadas en la PCB se han tenido en cuenta también la guía de diseño hardware. El fabricante recomienda el uso de una antena principal y una auxiliar debido a las bandas altas en el diseño de LTE-TDD, como band38, band40 y band41. Debido a la alta pérdida de inserción del cable de RF y las líneas de diseño, la sensibilidad del receptor de estas bandas arriba correrá el riesgo de cumplir con la autenticación sin la antena auxiliar.

Recomienda además un circuito de enlace de referencia por si el fabricante de las antenas tiene algún requisito de en el circuito de conexión entre la PCB y la antena. Dicho acondicionamiento se ha incluido en el diseño, aunque finalmente no se hayan implementado físicamente en la PCB.

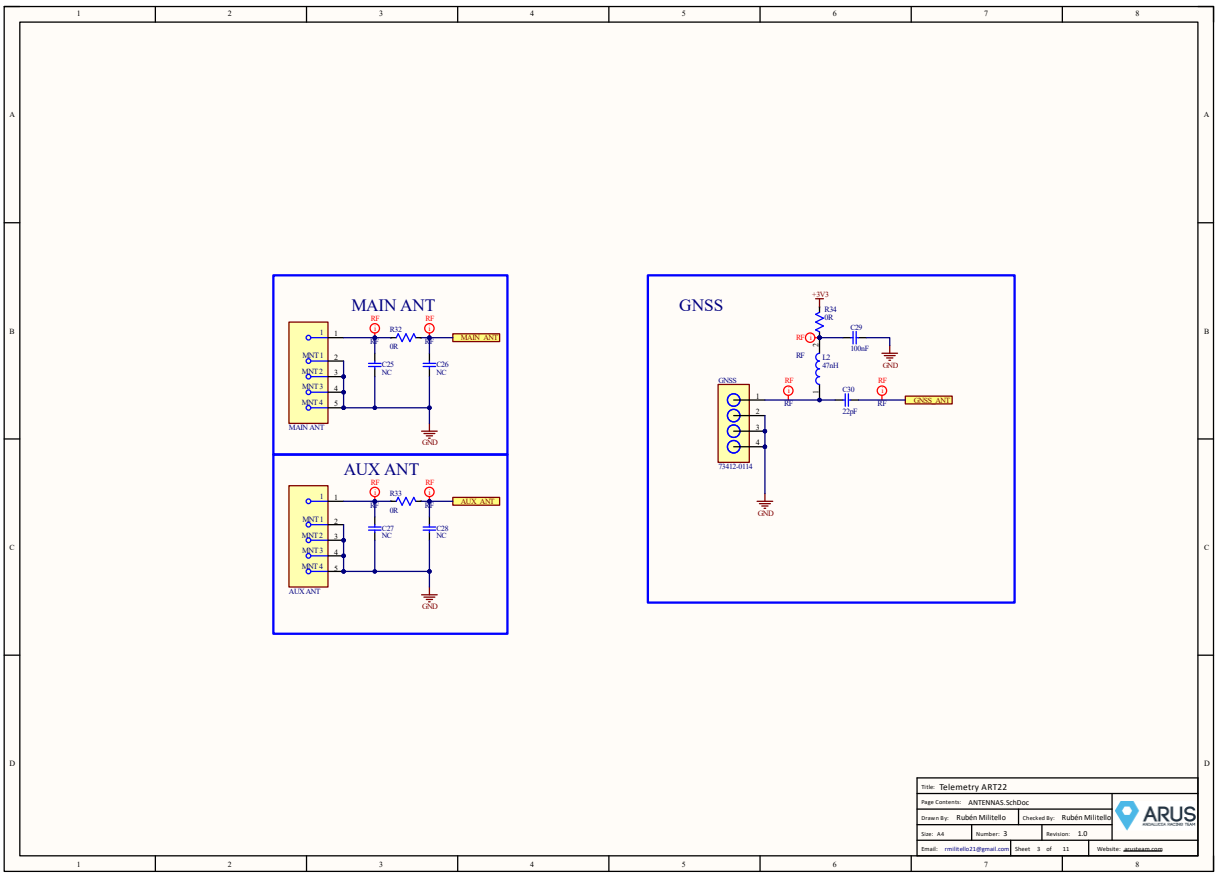


Ilustración 20: Esquemático del diseño de las antenas de PCB

Otra recomendación para tener en cuenta es el control de impedancia en las pistas de señales de la antena. Se recomienda un valor de 50Ω de impedancia. Para el control de impedancia se ha recurrido a la herramienta que tiene Altium Designer integrada. [3]

#	Name	Material	Type	Weight	Thickness	Dk
	Top Overlay		Overlay			
	Top Solder	Solder Resist	Solder Mask		0.02032mm	3.8
1	Top Layer		Signal	1oz	0.035mm	
	Dielectric 1		Prepreg		0.18034mm	4.6
2	GND Layer		Signal	1/2oz	0.0175mm	
	Dielectric 2	FR-4	Core		1.065mm	4.6
3	VCC Layer		Signal	1/2oz	0.0175mm	
	Dielectric 3		Prepreg		0.18034mm	4.6
4	Bottom Layer		Signal	1oz	0.035mm	
	Bottom Solder	Solder Resist	Solder Mask		0.02032mm	3.8
	Bottom Overlay		Overlay			

Ilustración 21: Captura de Altium Designer del stack layer manager

En la imagen superior se muestra el 'Stack Layer Manager' de la PCB. Como podemos apreciar en dicha imagen se definen todos los parámetros característicos de las cuatro capas que componen la PCB. Para obtener toda esta información se contactó con el fabricante y nos proporcionó un 'Stack Layer' para control de impedancia tal y como muestra la captura.

Una vez definido nuestro stack layer empleamos la calculadora de impedancias para obtener un valor de 50Ω. En la siguiente imagen se muestran los resultados obtenidos:

Top Ref	Bottom Ref	Width (W1)	Impedance (Z0)	Deviation	Delay...
	2 - GND Layer	0.29069mm	50	0.01%	6.2861...
1 - Top Layer	3 - VCC Layer	0.22254mm	49.99	0.02%	7.2055...
2 - GND Layer	4 - Bottom Layer	0.22254mm	49.99	0.02%	7.2055...
3 - VCC Layer		0.29069mm	50	0.01%	6.2861...

Ilustración 22: Ancho de pistas obtenidas para 50Ω

Podemos destacar dos anchos de pistas diferentes:

- 0.22254mm para las capas internas de la PCB
- 0.29069mm para las capas externas de la PCB

Posteriormente hemos creado un 'net label' llamado RF y hemos asociado una regla que cumpla el ancho calculado previamente en aquellos puntos que se indican en el esquemático a través de dicha 'net label' (Ilustración 21).

Como antena principal y auxiliar hemos empleado el modelo 132203-18 de Amphenol RF. Es un conector SMA de PCB 'through hole' el cual tiene una impedancia de 50 y además posee clasificación IP67 por lo que hace estanca la caja electrónica. En la siguiente imagen podemos ver el conector:

Ilustración 23:Conector de PCB para las antenas principal y auxiliar

En el caso de la antena con conectividad GNSS hemos recurrido a 73412-0114 de Molex. Se trata de un conector SMD micro coaxial vertical con una impedancia de 50Ω. En la siguiente podemos ver el conector:

Ilustración 24: Conector empleado para la antena de conectividad GNSS

2.4 Fuentes de alimentación

En la siguiente captura podemos observar las cuatro etapas de alimentación diferentes que tenemos en la PCB:

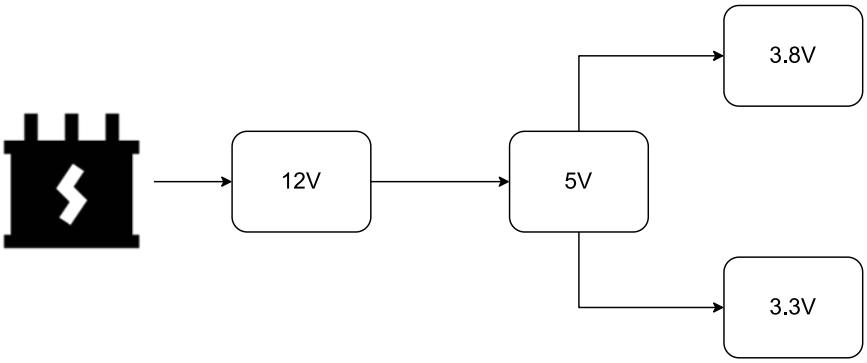


Ilustración 25: Diagrama de las etapas de alimentación

La alimentación principal es subministrada por la batería de baja de tensión del monoplaza. Para obtener las distintas alimentaciones auxiliares (5V,3.8V y 3.3V) que son necesaria para los demás componentes hemos planteado el siguiente diseño que se muestra en la Ilustración 26:

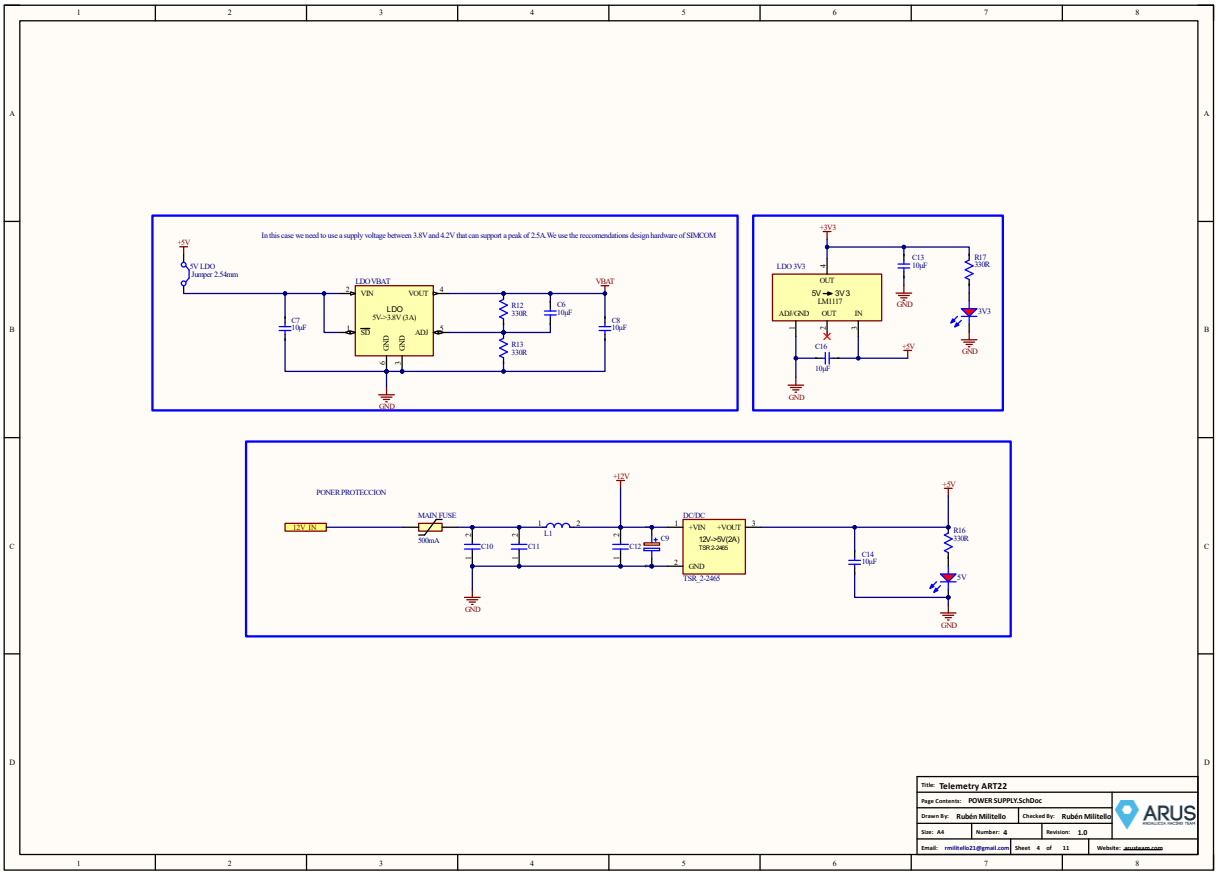


Ilustración 26: Esquemático del diseño de las etapas de alimentación

Puesto que el valor de tensión de la alimentación principal es variable es necesario elegir un regulador de tensión que tenga un gran rango de alimentación de entrada. En este caso se ha recurrido al convertidor CC/CC no asilado TSR 2-2450 fabricado por TRACO Power. Este tiene un rango de entrada de 6.5V-36V y proporciona una salida de 5V con una corriente de hasta 2A.

El motivo principal por el que hemos elegido un regulador conmutado es debido a la eficiencia del 96% (sin disipador) que asegura el fabricante en su hoja de datos y por su gran precisión en la tensión de salida. Los convertidores de esta tecnología son más costosos al igual que todos los componentes pasivos que son necesario en su acondicionamiento [4].

En esta etapa la alimentación de entrada es compartida con el resto de los subsistemas de baja tensión del monoplaza por lo que se ha puesto un fusible con su porta fusible como protección debido a que es una alimentación compartida. El resto del acondicionamiento es el recomendado por el fabricante en el datasheet [4]. Se trata de un filtro pi compuesto por la bobina L1 y los condensadores C11 y C12. Además, recomienda añadir C10, C9 y en la salida C14.

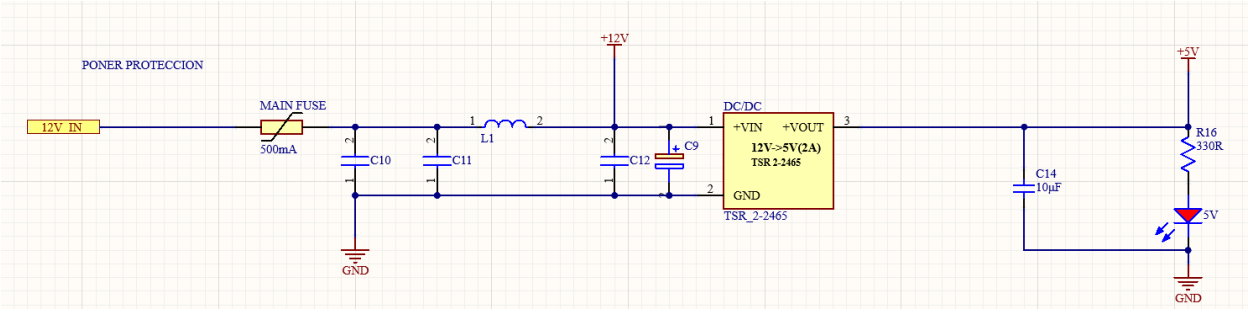


Ilustración 27: Etapa de alimentación 12V a 5V

Para la etapa de alimentación de 5V a 3.3V hemos empleado un regulador lineal ya que la potencia que debe disipar es mucho menor ya que tiene una entrada estable de 5V proveniente de la etapa anterior de alimentación. Hemos recurrido al regulador lineal de 5V a 3.3V LM1117MP-3.3/NOPB de Texas Instruments.

En este caso el acondicionamiento es más reducido ya que consiste simplemente en añadir en la entrada el condensador C16 y C13 a la salida para compensar la pérdida de corriente en caso de un cambio instantáneo en la corriente de salida, mejorando la respuesta transitoria de la carga.

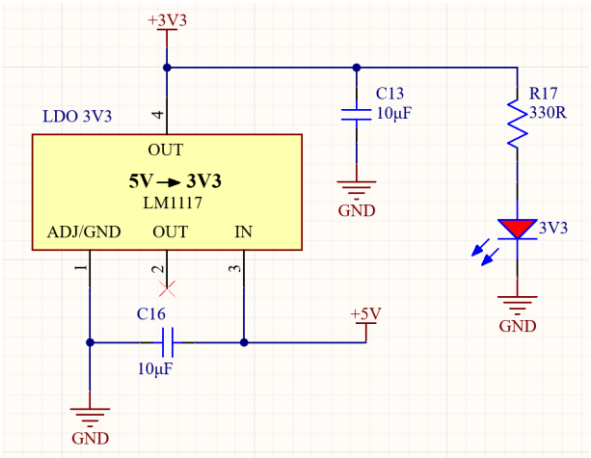


Ilustración 28: Etapa de alimentación 5V a 3.3V

Respecto al encapsulado hemos optado por un SOT223 de los diferentes que ofrece el fabricante. Para justificar el uso de dicho encapsulado hemos realizado los siguientes cálculos [5]:

$$\text{Potencia Disipada por el regulador lineal: } P_D = [(V_{in} - V_{out}) * I_{out}] + (V_{in} * I_{GND})$$

$$Donde\ V_{in} = 5V, V_{out} = 3.3V, I_{out} = 200mA, I_{GND} = 10mA$$

$$P_D = [(5 - 3{,}3) * 0{,}2] + (5 * 0{,}01) = 0{,}39W$$

Si calculamos ahora la temperatura de funcionamiento más alta en el regulador para una carga supuesta de 200mA:

$$T_J = T_A + (R_{\theta JA} * P_D)$$

$$Donde\ T_A = 30º(Suponemos\ temperatura\ en\ verano), R_{\theta JA} = 61{,}6\ \frac{º}{W}\ y\ P_D = 0{,}39W$$

$$T_J = 35ºC + (61.6º * 0{,}39) = 59{,}024ºC$$

Teniendo en cuenta que en la hoja de datos del fabricante tenemos una temperatura máxima de funcionamiento de 150ºC por lo que nuestro encapsulado seleccionado es correcto para la potencia máxima disipada supuesta para la etapa de 3.3V (200mA se estima que sería un consumo sobredimensionado para esta etapa).

Para la etapa de alimentación VBAT (la cual debe estar entre 3.4V y 4.2V) hemos recurrido también a un regulador lineal de 5V a 3.8V (voltaje nominal recomendado por el fabricante). Dicha etapa se encarga de la alimentación del módulo SIM7600E-H el cual sufre pico de hasta 2A cuando se emplea las señales GSM/GPRS tal y como indica la siguiente ilustración:

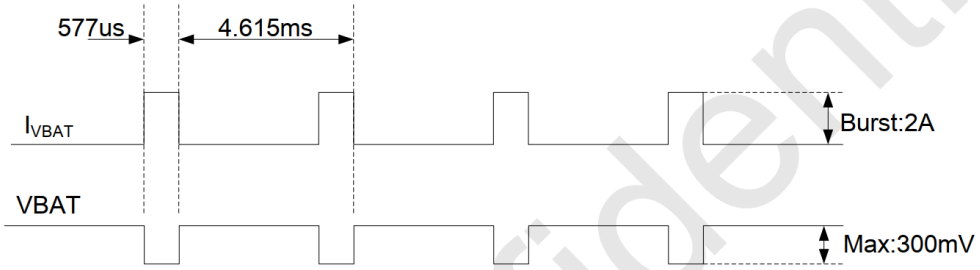


Figure 5: VBAT voltage drop during burst emission (GSM/GPRS)

Note: The test condition: The voltage of power supply for VBAT is 3.8V, Cd=100 µF tantalum capacitor (ESR=0.7Ω) and Cf=100nF (Please refer to Figure 6—Application circuit).

Por ello debemos recurrir a un regulador lineal el cual su tensión de salida no caiga más de 0,4V cuando la carga es de hasta 2A. En nuestro caso lo ideal sería obtener conectividad 4G/LTE, pero debido a que en muchas ocasiones realizamos diversas pruebas en circuitos de motorsport en los cuales no tenemos una gran conectividad por lo que deberíamos conformarnos con GSM, aunque nuestra tasa disminuya drásticamente.

Debido a lo comentado anteriormente, hemos recurrido al regulador lineal ajustable LP3856ESX-ADJ/NOPB de Texas Instruments. Se trata de un regulador lineal con una mínima caída de tensión a la salida el que soporta hasta 3A a la salida.

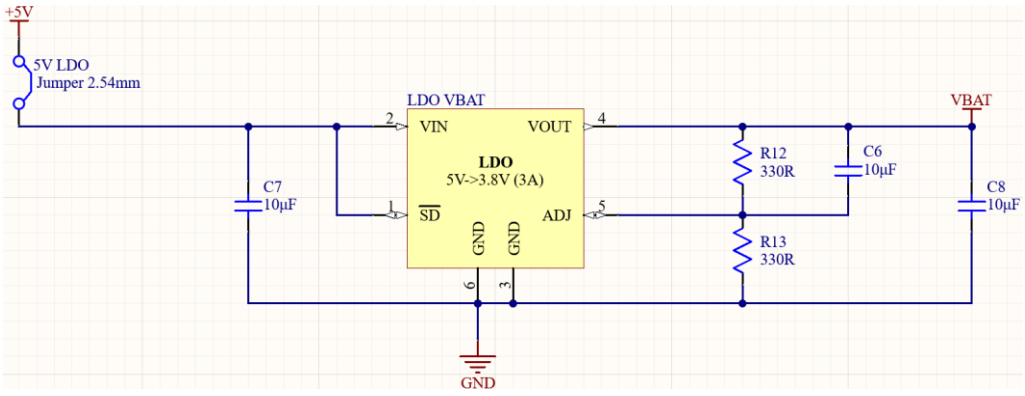


Ilustración 29: Etapa de alimentación 5V a 3.8V

Al igual que el circuito de la etapa de 3.3V el acondicionamiento se resumen en un condensador en la tensión de alimentación y otro en la tensión de salida. A diferencia del anterior se trata de un regulador ajustable por lo que se le ha añadido Rx Ry y Cz para obtener el valor de tensión deseado (en este caso 3.8V). Tal y como indica la hoja de datos del fabricante la tensión de salida viene dada por la siguiente ecuación [6]:

$$V_{OUT} = 1.216 * (1 + \frac{R1}{R2})$$

Respecto al encapsulado hemos optado por un DDPAK/TO-263-5 de los diferentes que ofrece el fabricante. Para justificar el uso de dicho encapsulado hemos realizado los siguientes cálculos [6]:

$$\text{Potencia Disipada por el regulador lineal: } P_D = [(V_{in} - V_{out}) * I_{out}] + (V_{in} * I_{GND})$$

$$Donde\ V_{in} = 5V, V_{out} = 3.8V, I_{out} = 200mA, I_{GND} = 10mA$$

$$P_D = [(5 - 3{,}8) * 2] + (5 * 0{,}01) = 2{,}45W$$

Si calculamos ahora la temperatura de funcionamiento más alta en el regulador para una carga supuesta de 200mA:

$$T_J = T_A + (R_{\theta JA} * P_D)$$

$$Donde\ T_A = 30º(Suponemos\ temperatura\ en\ verano)\ y\ P_D = 0{,}29W$$

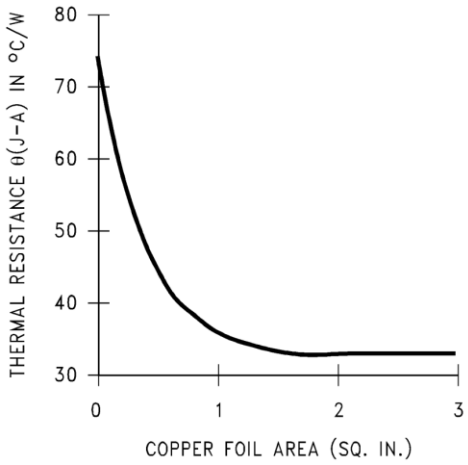


Ilustración 30: Resistencia térmica en función del área

$$Teniendo\ en\ cuenta\ la\ gráfica\ anterior\ podemos\ suponer\ una\ R_{\theta JA}\ de\ 65ºC/W$$

$$T_J = 35ºC + (65º * 2{,}45) = 194{,}25ºC$$

Si calculamos el valor deseado de nuestra resistencia térmica que deberíamos utilizar para estar al límite de los 150º, obtendríamos el siguiente valor:

$$R_{\theta JA} = \frac{T_{Rmax}}{P_D} = \frac{T_J - T_{Amax}}{P_D} = \frac{150 - 40}{2{,}45} \approx 44{,}89\frac{º}{C}$$

En este caso podemos observar que simplemente con hace un poco más grande la superficie de cobre a 0,5 pulgadas al cuadrado. En este caso como nuestro consumo nominal no es 2A (El pico de intensidad tiene un periodo de 4,6 ms y una duración de 577µs) y además nuestro modo de funcionamiento es con conectividad LTE en el cual el consumo máximo es de 0,6A.

2.5 Conversor USB-UART

El FT232RL es un popular circuito integrado (IC) de interfaz USB a serie UART desarrollado por FTDI (Future Technology Devices International). Se utiliza comúnmente para proporcionar conectividad USB a microcontroladores, sistemas integrados y diversos dispositivos que requieren una interfaz de comunicación UART. [7]

Características clave del FT232RL [7]:

1. Conectividad USB: El FT232RL permite que los dispositivos se comuniquen con una computadora host u otros dispositivos habilitados para USB. Actúa como un puente entre la interfaz USB y la interfaz UART.
2. Interfaz UART: El integrado admite la comunicación UART, que significa Receptor/Transmisor Asincrónico Universal. Proporciona un protocolo de comunicación serie simple y confiable.
3. Tasas de bits: El FT232RL admite una amplia gama de velocidades de transferencia de datos, desde 300 baudios hasta 3 Mbaudios, lo que brinda flexibilidad para diversas aplicaciones.
4. EEPROM Integrada: El FT232RL tiene una EEPROM integrada (Memoria Programable de Solo Lectura Eléctricamente Borradable) que almacena información específica del dispositivo y configuraciones. Los fabricantes pueden utilizar esto para almacenar números de serie únicos, identificadores de productos y otra información relevante.

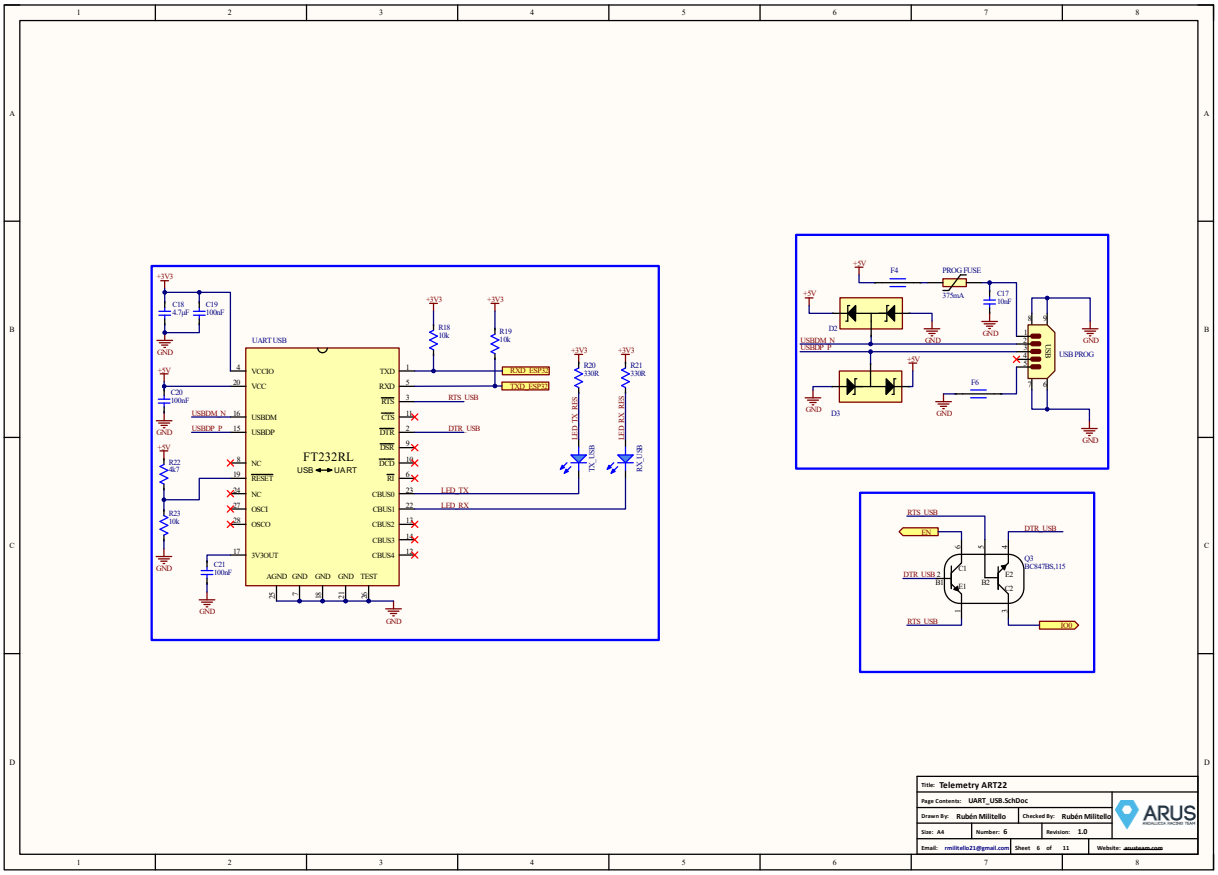


Ilustración 31: Esquemático del diseño del conversor USB-UART

A lo que refiere al esquemático podemos resumir el diseño en los siguientes puntos:

1. Conexión USB:

   Para empezar, se asignaron los pines D+ y D- del FT232RL a los pines correspondientes del conector USB. Estos son los canales de datos bidireccionales para la comunicación USB. Aseguré que el conector USB estuviera bien identificado en el esquemático.

   Se añadieron varios diodos TVS para mayor protección eléctrica ya que el conector USB se manipula con frecuencia y podemos tener una descarga ESD. Los diodos TVS son empleados en aplicaciones de protección contra ESD para desviar la corriente generada por la descarga electrostática lejos de los componentes electrónicos sensibles.

   Cuando una descarga ESD se produce en el circuito protegido, el diodo TVS se activa instantáneamente, proporcionando una ruta de baja resistencia para que la corriente fluya hacia tierra. Esto evita que la energía de la descarga dañe los componentes electrónicos al limitar el voltaje que llega a ellos.

2. Pines de alimentación:

   Conecté el pin VCC del FT232RL a la fuente de alimentación del circuito, que era de 5V en este caso. En el caso del pin VCCIO lo conecté a 3.3V ya que la transmisión. El pin GND se conectó al plano de tierra para establecer la referencia de voltaje común en todo el circuito. Por último, añadieron diversos condensadores de desacoplo de valores típicos.

3. Pines UART:

   Enlacé el pin TXD del FT232RL al pin RX del microcontrolador con el que estaba comunicando, y el pin RXD del FT232RL al pin TX del microcontrolador. Esta configuración permitiría la comunicación serial entre los dos dispositivos. Además, se añadieron unas resistencias pull-up a 3.3V.

4. Pines de control y programación:

   EN (Enable) y GPIO0: El pin EN es el pin de habilitación del ESP32. Se debe conectar a 3.3V para que el ESP32 funcione.

   El pin GPIO0 se utiliza para poner el ESP32 en modo de programación o arranque normal. Para programar, GPIO0 debe estar conectado a tierra (GND); para arranque normal, debe estar conectado a 3.3V.

   Gracias a los transistores (BJTs) empleados y conectando las señales de EN y IO0 a las diferentes entradas de estos podemos programar directamente el microcontrolador.

5. Pines GPIO:

   Decidí utilizar algunos pines GPIO para controlar LED indicadores (en este caso los pines CBUS0 y CBUS1). Con esto se puede depurar de manera visual si el integrado está recibiendo y transmitiendo ya que cada uno de ellos se encienden cuando las señales TX y RX están funcionando. Los configuré como salidas y los conecté a través de resistencias limitadoras de corriente.

6. Pin Reset:

   En este caso se hizo un divisor resistivo empleando las resistencias R22 de 4.7kΩ y R23 de 10kΩ. Ya que una tensión de referencia de 5V obtenemos un valor de 3.4V. Debido a que el pin de ‘reset’ se activa con un nivel lógico bajo (respetando todos los umbrales eléctricos del datasheet) estamos forzando que nunca se reinicie.

2.6 Celdas De Potencia y Entradas aisladas

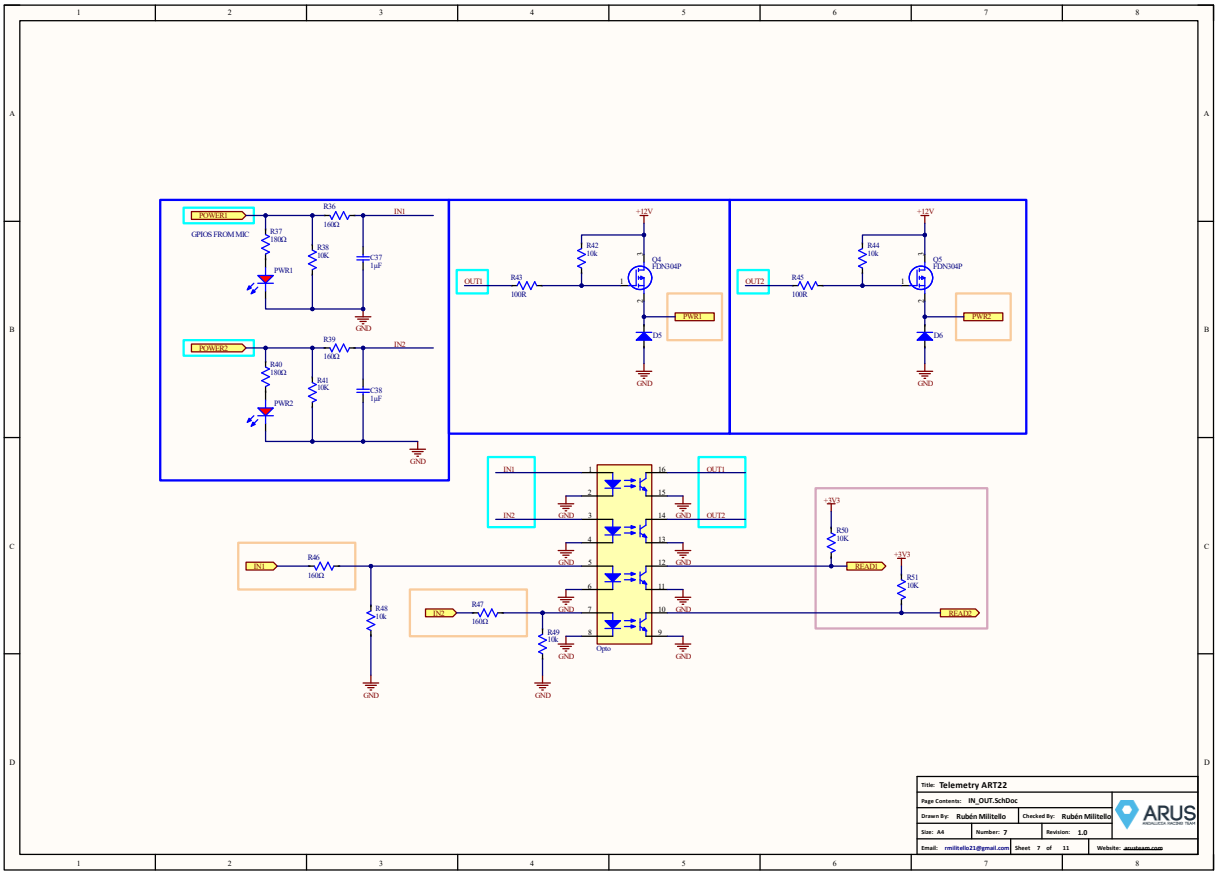


Ilustración 32: Hoja de esquemático de las celdas de potencia y entradas aisladas

Este diseño de esquemático puede ser divido en dos bloques principalmente:

- Celdas de potencia: Permite alimentar ciertos subsistemas del monoplaza a 12V.
- Entradas aisladas: Nos permite leer entradas de tensión configurable a través del microcontrolador.

Las celdas de potencia (2 incluidas en esta PCB) consisten en un optoacoplador el cual es controlado por un pin del microcontrolador. Dicho optoacoplador controla la puerta de un transistor tipo P (como un interruptor el cual da 12V o alta impedancia).

Con el pin del microcontrolador a nivel lógico alto polarizamos el led (el cual a su vez debe ser capaz de polarizar un fototransistor), y como consecuencia introduce un valor lógico ‘0’ en la puerta del MOSFET tipo P, activando los 12V de salida. En el caso de que el microcontrolador de un nivel lógico bajo, el led no se polarizará y por consecuencia el fototransistor tampoco. Al tener una ‘pull up’ a 12V en la puerta del MOSFET tipo P, tendremos a la salida alta impedancia. En el siguiente esquema se muestra el funcionamiento lógico:

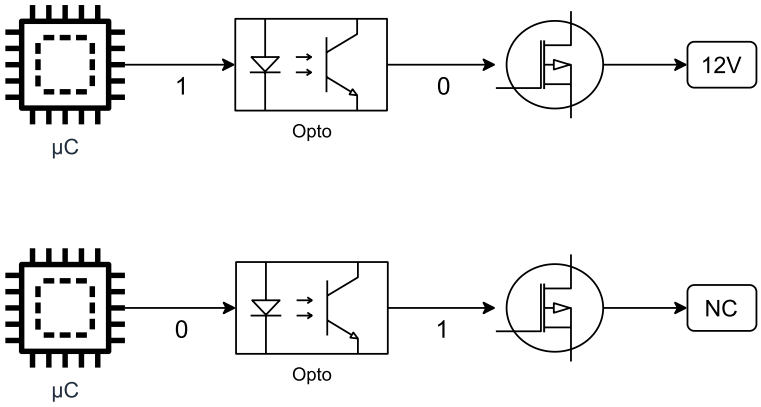


Ilustración 33: Esquema lógico celda de potencia

Se ha añadido un led en cada salida del microcontrolador para visualmente poder depurar si cada se ha activado de manera correcta (Posee gran utilidad para depurar el correcto funcionamiento cuando a través de la interfaz gráfica se activa cualquiera de las dos salidas).

Se ha dimensionado el valor de las resistencias para obtener una corriente de polarización del orden de 12mA (según la hoja de datos permite como valor absoluto hasta 50mA). [8]

En el caso del MOSFET tipo P se ha dimensionado todas las resistencias para respetar el rateo máximo que indica el fabricante en la hoja de datos. [9]

Las entradas aisladas nos permiten leer diferentes puntos de tensión presente en diferentes subsistemas del monoplaza con el objetivo de poder monitorizar diferentes funcionalidades del monoplaza. Podemos monitorizar hasta dos etapas del shutdown para verificar el estado de algún subsistema y visualizar el estado del circuito de shutdown a través de la interfaz gráfica. En este caso tenemos una lógica inversa de lectura. Es decir, un valor de 12V a la entrada del optoacoplador supone un '1' lógico para el microcontrolador y viceversa.

Un valor de tensión en la entrada de 5V,12V (es ajustable con la resistencia serie que posee el led del optoacoplador) supone polarizar el led y por consecuencia polarizar el fototransistor, lo que supone un '0' lógico en la lectura del microcontrolador. Para una lectura de 0V obtendremos un '1' lógico en el microcontrolador.

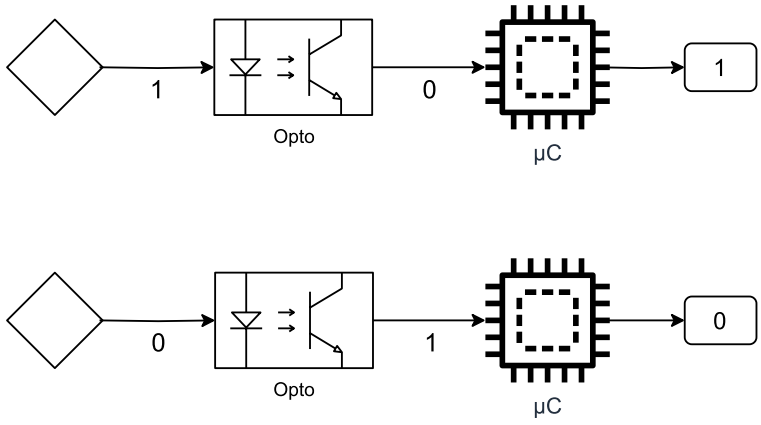


Ilustración 34: Esquema lógico de las medidas aisladas

2.7 Microcontrolador

El ESP32-WROOM-32D es un módulo específico del ESP32 fabricado por Espressif Systems. Este módulo presenta las siguientes características: [10]

1. Microcontrolador de doble núcleo: Al igual que el ESP32-WROOM-32D genérico, este módulo contiene un microcontrolador de doble núcleo Tensilica LX6 que ofrece una velocidad de reloj de hasta 240 MHz.
2. Conectividad inalámbrica: El módulo ofrece conectividad Wi-Fi 802.11 b/g/n y Bluetooth v4.2 (Bluetooth Low Energy), lo que permite la comunicación inalámbrica con otros dispositivos y acceso a Internet.
3. Memoria: El ESP32-WROOM-32D (M113DH3200PH3Q0) generalmente viene con una memoria flash integrada de tamaño variable, que puede oscilar entre 4 MB y 16 MB.
4. Periféricos y E/S: Al igual que el ESP32-WROOM-32D estándar, este módulo también tiene una variedad de interfaces periféricas, como UART, I2C, SPI, I2S, ADC, DAC, PWM, GPIO y más, lo que facilita la conexión y el control de diversos sensores y actuadores.
5. Bajo consumo de energía: Este módulo, al ser parte de la familia ESP32, también ofrece eficiente consumo de energía, lo que lo hace adecuado para dispositivos portátiles y con batería.
6. Sistema operativo y programación: Puede ejecutar el sistema operativo FreeRTOS y es compatible con el framework de desarrollo Arduino, lo que facilita la programación y desarrollo de aplicaciones.
7. Antena integrada: Este módulo también incluye una antena integrada, lo que facilita su diseño y reduce el espacio necesario para la implementación.

En esta parte del esquemático diseñamos todo el acondicionamiento necesario para el correcto funcionamiento del microcontrolador. En este caso hemos recurrido al ESP32-WROOM-32D como el microcontrolador principal de nuestra PCB.

En primer lugar, destacamos los dos pulsadores para reiniciar y activar el modo ‘bootloader’ del microcontrolador. Se ha añadido un filtro anti-rebote para evitar la interpretación de pulsaciones no reales por parte del microcontrolador.

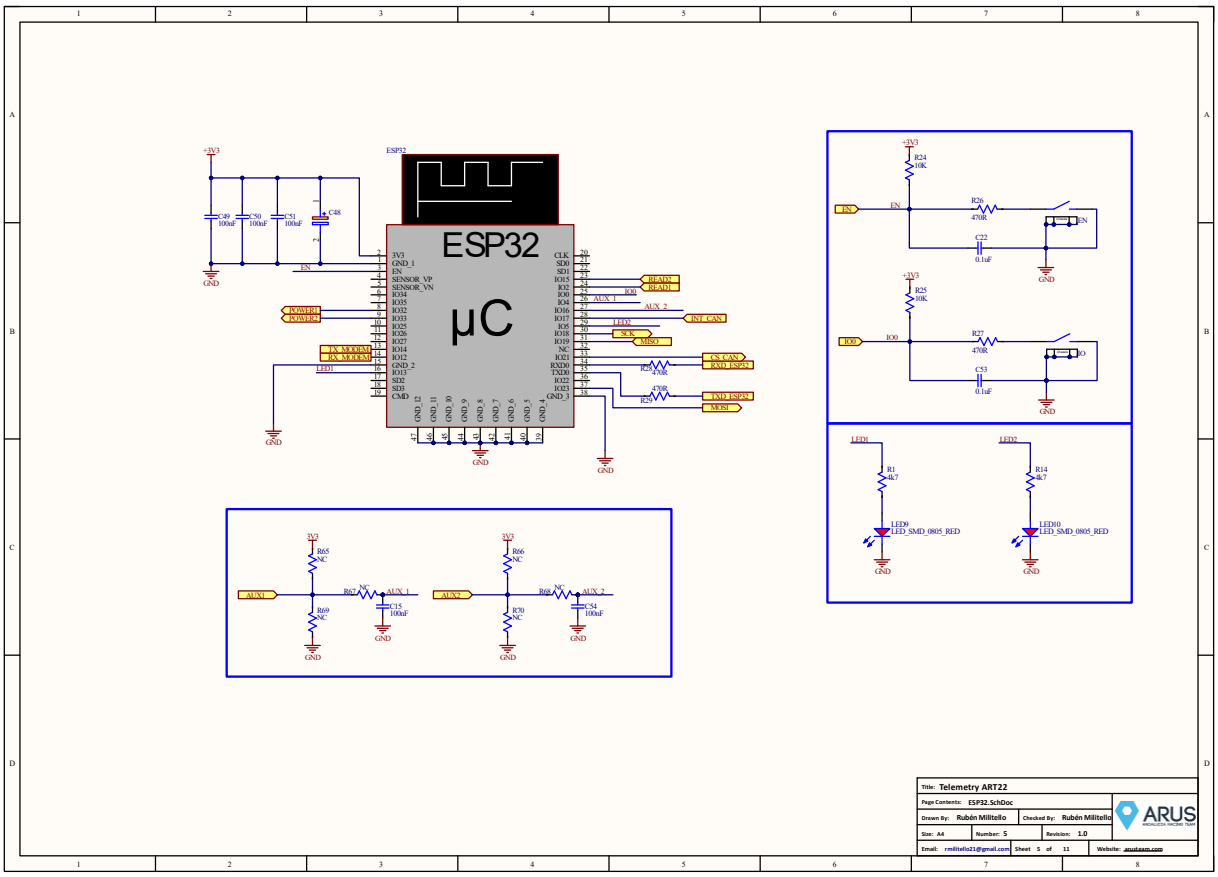


Ilustración 35: Hoja esquemático del microcontrolador

1. Botón de bootloader (Boot button): El ESP32-WROOM-32D (M113DH3200PH3Q0) suele incluir un botón de bootloader que se utiliza para colocar el módulo en modo de arranque especial. Este modo permite la carga del firmware o la reprogramación del ESP32-WROOM-32D utilizando herramientas de programación.
2. Botón de reset (Reset button): El módulo también cuenta con un botón de reset que se utiliza para reiniciar el microcontrolador. Al presionar este botón, se borra el estado actual del ESP32 y se reinicia el programa o firmware.
3. Condensadores de desacoplo (Decoupling capacitors): En el diseño del circuito del ESP32-WROOM-32D, se recomienda utilizar condensadores de desacoplo para asegurar una fuente de energía estable y reducir el ruido en el suministro de energía. Estos condensadores se colocan cerca de la fuente de alimentación del módulo y se conectan entre los pines de alimentación (VCC) y tierra (GND) para filtrar y estabilizar las fluctuaciones de voltaje.

Los botones de bootloader y reset son útiles durante el proceso de programación y depuración del módulo, ya que permiten controlar el inicio y reinicio del ESP32-WROOM-32D.

2.8 Driver Can y Transceiver

La hoja de esquemático del bus CAN queda resumida de la siguiente manera:

1. MCP2515: Es el controlador CAN y está conectado al microcontrolador a través de la interfaz SPI. El MCP2515 se encarga de gestionar el protocolo CAN, incluida la transmisión y recepción de mensajes.
2. Transceptor SN65HVD251D: Este es el transceptor CAN que se encarga de la capa física de la comunicación CAN. Proporciona la interfaz entre el controlador CAN y el cableado del bus CAN.
3. Cristal: El l MCP2515 requiere un cristal externo para generar el reloj necesario para su funcionamiento.
4. Condensadores de desacoplamiento: Se colocan condensadores de desacoplamiento cercanos al MCP2515, SN65HVD251D y al cristal para filtrar el ruido y garantizar una alimentación estable.
5. Resistores de terminación: Al igual que en las descripciones anteriores, el bus CAN requiere resistencias de terminación de 120 ohmios en ambos extremos del bus para evitar problemas de reflexión de señal.
6. Fuente de alimentación: Tanto el MCP2515 como el SN65HVD251D necesitan una fuente de alimentación adecuada para su correcto funcionamiento. Asegúrate de suministrar la tensión de alimentación especificada por los fabricantes.
7. Otros componentes: Dependiendo del diseño específico, es posible que haya otros componentes adicionales, como resistencias de pull-up o pull-down, filtros y protecciones contra sobretensiones y descargas electrostáticas.

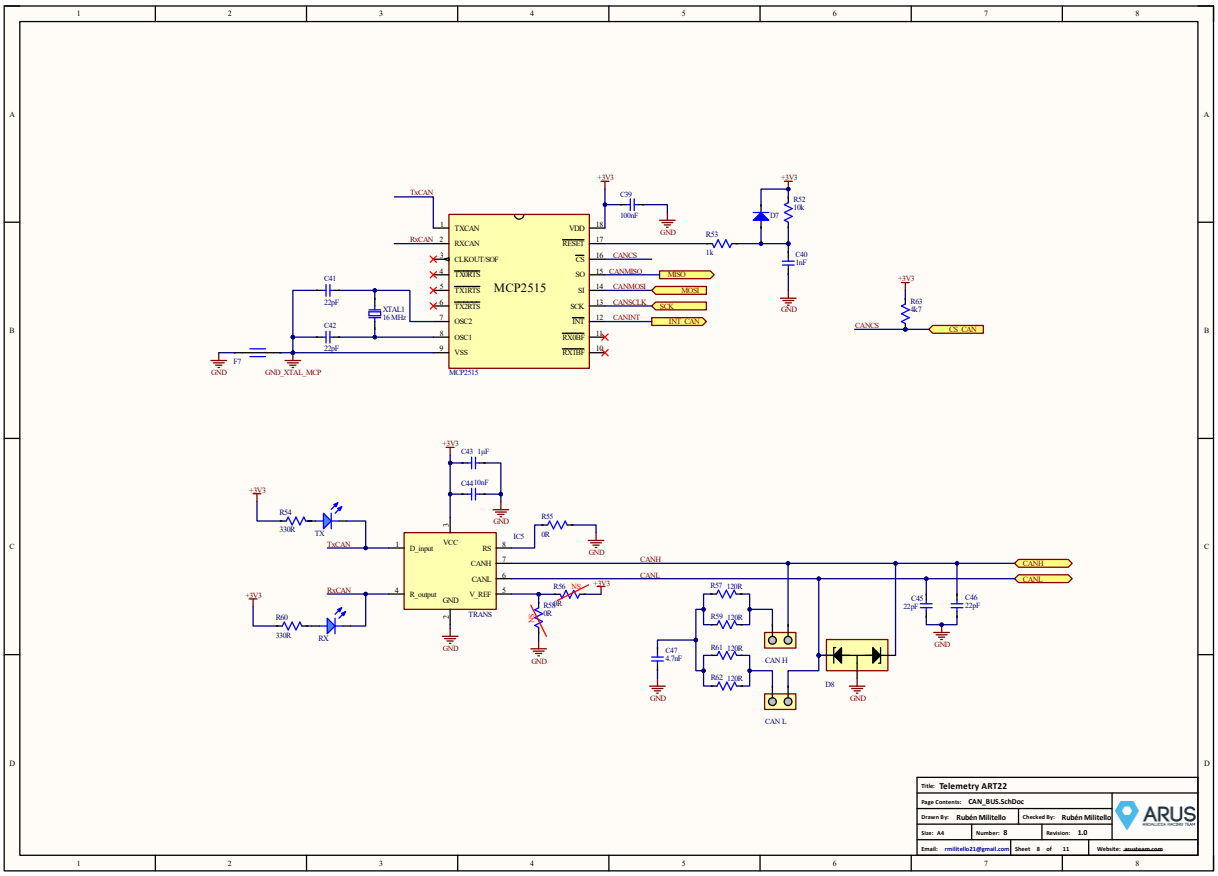


Ilustración 36: Hoja de esquemático del bus CAN

2.9 Conector PCB

El conector empleado es el M80-5001442 de Harwin. Es un conector macho para PCB Ángulo recto HARWIN serie Datamate J-Tek de 14 vías, 2 filas, paso 2.0mm.

En la siguiente tabla queda resumida el pin -out del conector de la PCB:

Designador	Nombre	Descripción
1	IN2	Entrada aislada
2	IN1	Entrada aislada
3	PWR2	12V de salida
4	PWR1	12V de salida
5	CANL	Señal diferencial negativa
6	CANH	Señal diferencial positiva
7	NC	N/A
8	AUX1	Entrada digital
9	AUX2	Entrada digital
10	GND_IN	Referencia
11	GND_IN	Referencia
12	GND_IN	Referencia
13	GND_IN	Referencia
14	12V_IN	Alimentación

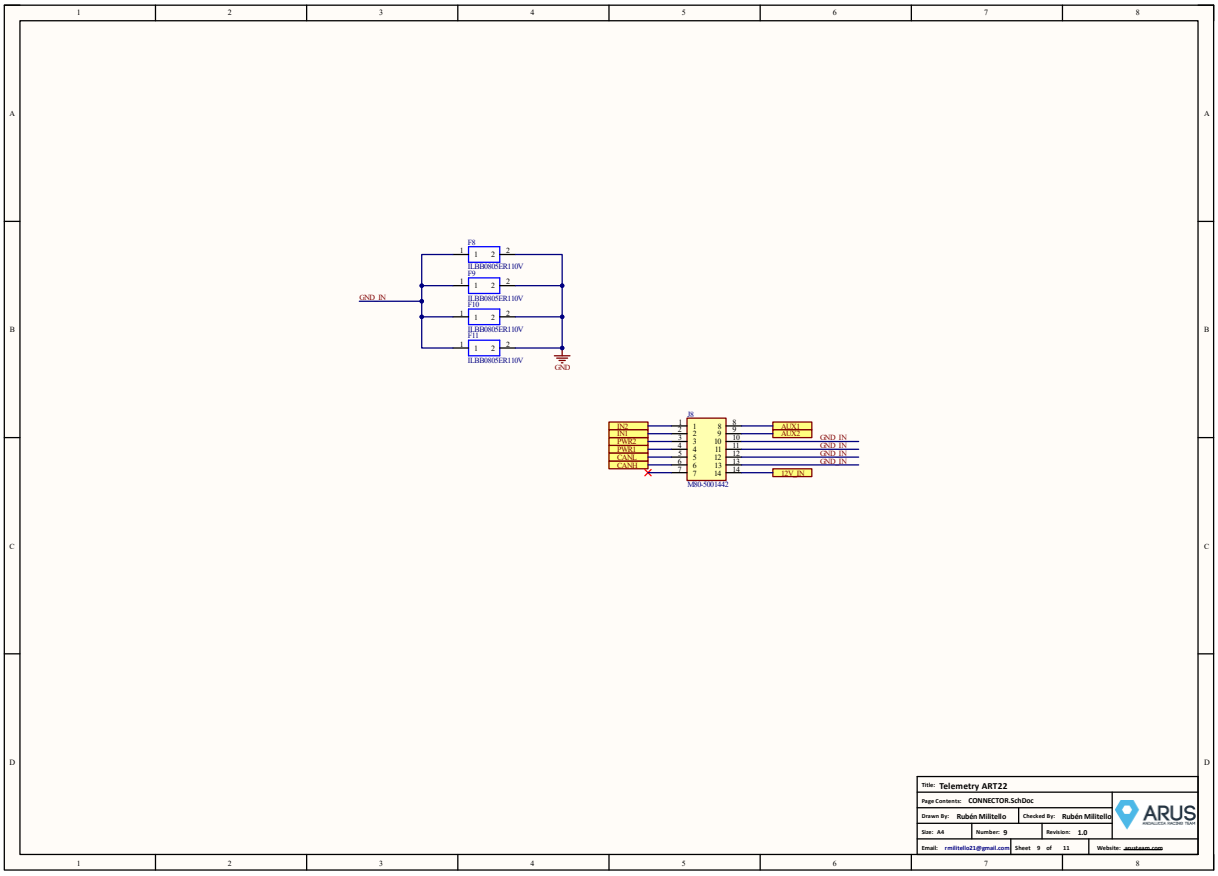


Ilustración 37: Hoja de esquemático del conector de la PCB

2.10 SIM7600E-H

El SIM7600E-H es un módulo celular fabricado por SIMCom Wireless Solutions, una empresa líder en soluciones y módulos de comunicación inalámbrica. Forma parte de la serie SIM7600 y está diseñado para proporcionar una conectividad celular confiable para diversas aplicaciones.

Las características principales del SIM7600E-H son las siguientes: [11]

1. Conectividad Celular: El módulo admite redes 2G (GSM/GPRS), 3G (UMTS/HSPA) y 4G LTE (LTE-FDD/LTE-TDD). Esto le permite conectarse a una amplia gama de redes celulares en todo el mundo, lo que lo hace adecuado para despliegues globales y regionales.
2. Comunicación de Datos: El SIM7600E-H permite la comunicación de datos a través de varias interfaces, como UART, USB y GPIO, lo que le permite integrarse con diferentes dispositivos principales como microcontroladores, computadoras de placa única u otros sistemas integrados.
3. Sistema de Posicionamiento Global (GPS): Algunas variantes de la serie SIM7600 vienen con funcionalidad GPS integrada. Esto permite el seguimiento de ubicación y capacidades de posicionamiento, útiles para aplicaciones que requieren servicios basados en la ubicación.
4. Soporte para Voz y SMS: El módulo puede admitir llamadas de voz y mensajes de texto SMS, lo que lo convierte en una solución de comunicación completa para ciertas aplicaciones.
5. Interfaz de Tarjeta SIM: El módulo generalmente tiene una interfaz para insertar una tarjeta SIM estándar, lo que le permite autenticarse con la red celular y acceder a los servicios de datos.

Resumen Detallado del Esquemático para el Módulo Simcom 7600E-H:

1. Acondicionamiento del Módulo Simcom 7600E-H:

   El diseño comprende estrategias clave para optimizar el rendimiento y la confiabilidad del módulo Simcom 7600E-H:

   Condensadores de Desacople: Se implementan condensadores de desacople estratégicamente conectados en paralelo a la fuente de alimentación. Esto contrarresta eficazmente el ruido y las fluctuaciones en la tensión de alimentación, garantizando un suministro eléctrico estable y limpio.

   Filtro de Alimentación: Siguiendo las directrices de diseño del fabricante, se introduce un filtro en la trayectoria de alimentación. Esto ayuda a reducir las interferencias electromagnéticas y asegura que la señal de alimentación llegue al módulo en condiciones óptimas.

   Botón de Reinicio con Filtro Anti-Rebote: Se incorpora un botón de reinicio diseñado con un circuito de filtro anti-rebote. Este diseño inteligente minimiza la posibilidad de activaciones no deseadas debido a fluctuaciones temporales en la señal del botón.

2. Indicadores LED:

   Indicador de Estado de Red: El esquemático incluye un primer LED que actúa como indicador visual del estado de la red del módulo. Este LED proporciona una retroalimentación rápida y clara sobre si el módulo está conectado a una red o no.

   Indicador de Alimentación: Un segundo LED se utiliza para señalar si el módulo está recibiendo alimentación eléctrica. Esto permite una rápida verificación de la disponibilidad de energía sin la necesidad de comprobar manualmente conexiones eléctricas.

3. Control de Encendido (Power Key):

   Flexibilidad de Encendido y Apagado: Para administrar el encendido y apagado del módulo, se ha integrado un control mediante jumper. Esto facilita el proceso de manipulación y brinda la posibilidad de desconectar el módulo según las necesidades.

   Habilitación Permanente: Existe la opción de establecer una habilitación permanente del módulo

mediante la soldadura de la resistencia R15. Esta característica puede ser útil en escenarios donde se requiere que el módulo permanezca activo continuamente.

4. Modo de Vuelo (Flight Mode):

   Gestión del Modo Avión: Se proporciona un mecanismo para activar el modo avión en el módulo Simcom 7600E-H. Esta funcionalidad es análoga al control de encendido (Power Key) y permite poner el módulo en un estado de baja energía para cumplir con los requisitos de restricciones de comunicación.

5. Traductor de Voltaje para Comandos AT:

   Superación de Diferencias de Tensión: Dado que la salida UART del Sim7600E-H opera a 1.8V y el microcontrolador ESP32 opera a 3.3V, se ha implementado el integrado TXB0108 de Texas Instruments. Este componente facilita la comunicación entre ambos dispositivos al realizar una conversión de niveles de tensión bidireccional de 8 bits. Su detección automática de dirección garantiza una interfaz eficaz y segura.

6. Pines de Depuración:

   Facilitando la Depuración: El esquemático incluye pines designados para la interfaz I2C y la interfaz UART del módulo. Estos puntos de acceso permiten una conexión conveniente para tareas de depuración y comunicación con el módulo. Proporcionan una vía para monitorear y ajustar el comportamiento del módulo durante el desarrollo y las pruebas.

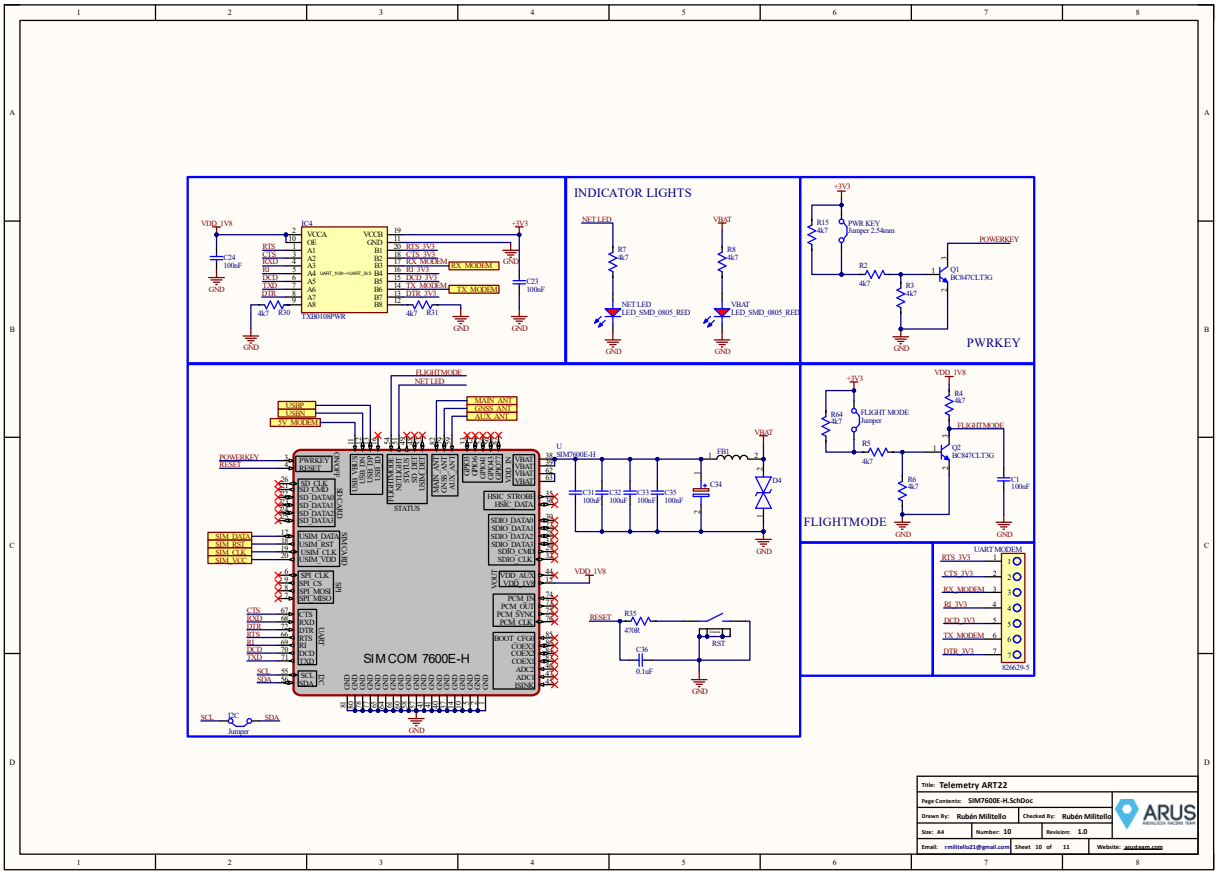


Ilustración 38: Esquemático para el módulo Simcom 7600E-H

2.11 Interfaz USB del SIMCOM 7600E-H

En la fase inicial del diseño, se procedió a la asignación precisa de los pines D+ y D- del FT232RL, garantizando una correspondencia adecuada con los pines correspondientes en el conector USB. Estos pines son esenciales, ya que constituyen los canales de datos bidireccionales fundamentales para la comunicación USB. Se prestó una atención meticulosa a la representación correcta del conector USB en el esquemático, asegurando así su identificación sin ambigüedades.

Conscientes de la exposición frecuente del conector USB a manipulaciones, se implementó una estrategia de protección eléctrica integral. Para este propósito, se incorporaron diodos TVS, cuya función principal radica en salvaguardar los componentes electrónicos ante las amenazas de descargas electrostáticas (ESD). Estos diodos son una elección óptima en situaciones que requieren protección contra ESD, ya que derivan eficazmente la corriente generada por estas descargas, desviándola de manera segura lejos de los elementos electrónicos sensibles.

El principio detrás de los diodos TVS es sencillo pero efectivo: en caso de una descarga ESD en el circuito protegido, el diodo TVS se activa de inmediato, estableciendo una vía de baja resistencia para el flujo de corriente hacia la tierra. Este mecanismo impide que la energía liberada por la descarga cause daños a los componentes electrónicos al limitar el voltaje que alcanza dichos componentes.

De manera adicional, se incorporó un fusible como una capa suplementaria de protección. Este componente juega un papel crucial al interrumpir la corriente eléctrica cuando esta supera un nivel predeterminado. Al hacerlo, el fusible desempeña un papel crucial en la prevención de situaciones como cortocircuitos y sobrecorrientes, que podrían tener consecuencias perjudiciales para el circuito.

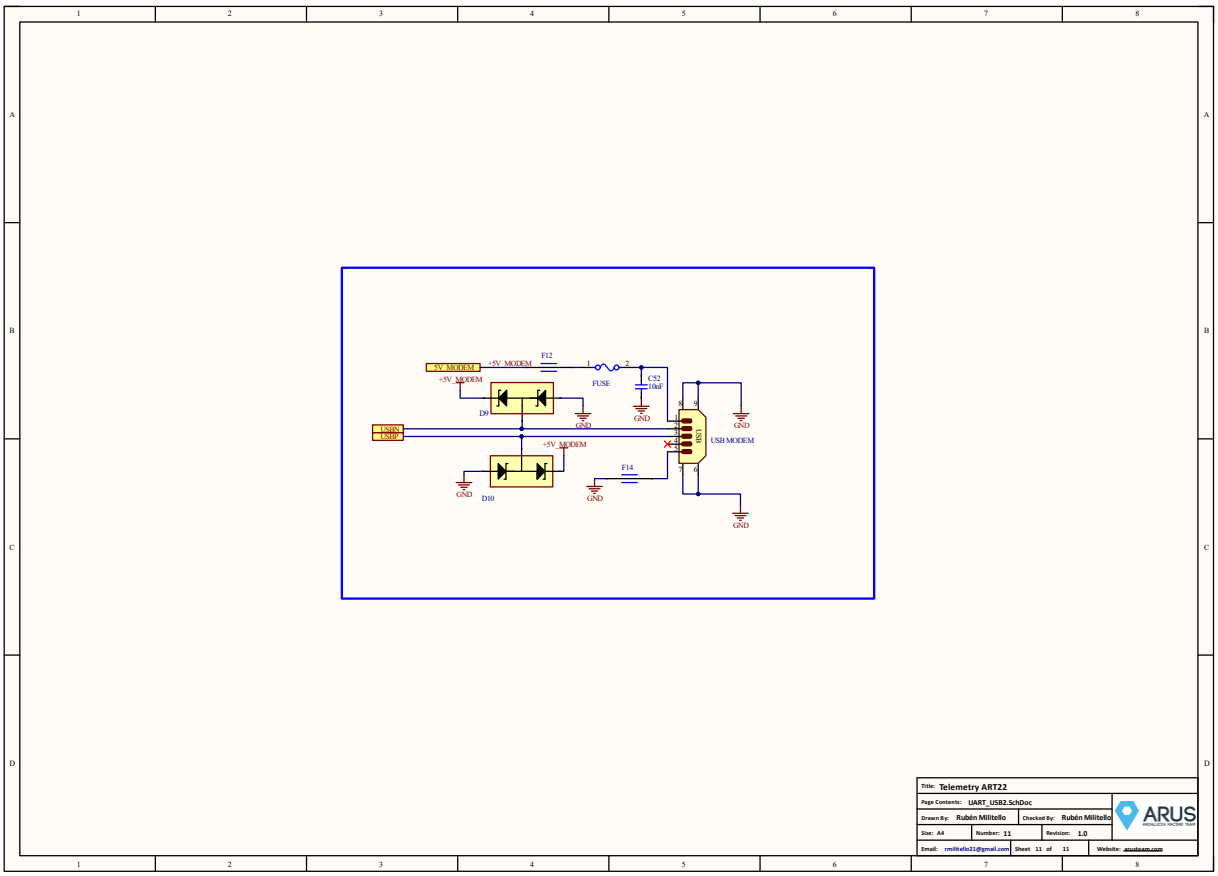


Ilustración 39: Interfaz USB del SIMCOM 7600E-H

3 LAYOUT PCB

A continuación, podemos observar una vista en 3D de la PCB:

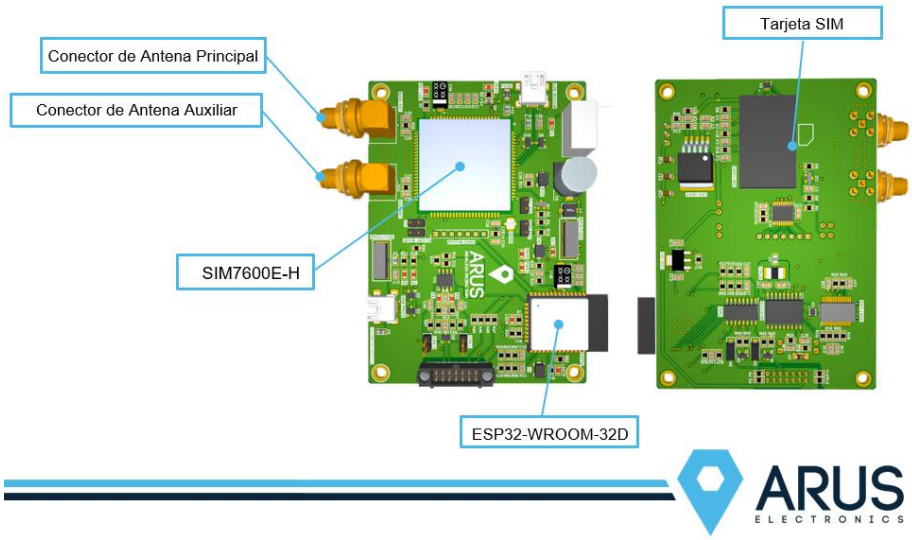


Ilustración 40: Vista superior e inferior de la PCB

3.1 Resumen del Layout

Desde el principio, nos enfocamos en la distribución de componentes de manera que las rutas de señal fueran lo más cortas y directas posible. Esto no solo mejora la eficiencia de las señales, sino que también ayuda a prevenir problemas de interferencia y ruido.

Hemos asignado capas internas específicas para el plano de tierra y el plano de alimentación, lo que proporciona una base sólida para todas nuestras señales. Esto reduce la interferencia electromagnética y garantiza un entorno más limpio para nuestras comunicaciones de telemetría.

Para las señales de alta velocidad, hemos utilizado pistas de impedancia controlada para asegurarnos de que las señales se transmitan de manera precisa y sin distorsiones. Esto es especialmente importante para mantener la integridad de los datos en aplicaciones de telemetría.

Las señales críticas, como los relojes y los buses de datos, han sido enrutadas de manera diferencial y cuidadosa para minimizar cualquier interferencia cruzada. Esto asegura que nuestras mediciones de telemetría sean precisas y consistentes.

Antes de finalizar el diseño, realizamos múltiples revisiones exhaustivas y verificaciones cruzadas para garantizar que todas las conexiones fueran correctas y que no hubiera errores de enrutamiento. También realizamos simulaciones para asegurarnos de que nuestras señales cumplieran con nuestras expectativas de rendimiento.

3.2 Stack Layer

El proceso de diseñar una placa de circuito impreso (PCB) es una fase crítica en el desarrollo de sistemas electrónicos. El stackup layer, que se refiere a la organización y disposición de las capas en una PCB, juega un papel fundamental en la funcionalidad y rendimiento del diseño. En este documento técnico, se presenta un enfoque exhaustivo para un stackup layer de 4 capas destinado a un sistema electrónico complejo que integra elementos como un microcontrolador, un modem LTE, conectores USB, antenas, indicadores LED y otros componentes. A continuación, se detallan las capas en su disposición vertical:

Capa de Enrutamiento (Top Layer): La capa superior de mi PCB, también llamada "Top Layer", está destinada al enrutamiento de pistas de señal. En esta capa se alojan los componentes electrónicos y se establecen conexiones esenciales para el funcionamiento del circuito. Aquí, el diseño de las pistas es crucial para garantizar la integridad de la señal y la funcionalidad del sistema. [12]

Capa de Plano de Tierra (GND Plane): Inmediatamente debajo de la capa de componentes se encuentra la capa de "GND Plane", que actúa como una conexión a tierra continua. Esta capa proporciona una referencia de voltaje estable y actúa como una ruta de retorno para las corrientes. Mantener una conexión a tierra adecuada es esencial para reducir el ruido y asegurar un rendimiento confiable. [12]

Capa de Alimentación (Power Plane): La siguiente capa es la de "Power Plane", donde se enrutan las pistas de alimentación. Aquí, se establecen trazados para llevar la energía a los diferentes componentes del circuito. Dependiendo de las necesidades del diseño, pueden existir planos de alimentación separados para diversos voltajes o corrientes. [12]

Capa de Enrutamiento (Bottom Layer): La capa inferior de la PCB, también conocida como "Bottom Layer", está destinada nuevamente al enrutamiento de señales. Aquí se completan las conexiones y trazas que no se pueden acomodar en las capas anteriores. Esta capa permite una mayor flexibilidad en el diseño y el enrutamiento. [12]

En conjunto, estas capas forman un "stack layer" estratégico que mejora la integridad de la señal, reduce el ruido y proporciona una distribución efectiva de energía en todo el circuito. Esta disposición de capas se ha diseñado para cumplir con los requisitos específicos del proyecto y asegurar un rendimiento correcto del sistema. [12]

3.2.1 Capa de Enrutamiento (Top Layer)

La primera capa del stackup en el diseño de sistemas electrónicos puede ser definida como la etapa inicial de ubicación estratégica de los componentes principales y el enrutamiento de señales críticas. En esta fase, se dedica un esfuerzo meticuloso a colocar los componentes esenciales en posiciones estratégicas que garanticen la accesibilidad y la optimización del funcionamiento del sistema. Además, se aborda el enrutamiento de las señales, un aspecto crucial para asegurar una transmisión de datos fluida y confiable entre los distintos componentes.

La primera capa del stackup desempeña un papel fundamental al dar cabida a los componentes principales que tienen un papel integral en la funcionalidad global del sistema. Elementos clave como el microcontrolador, el modem LTE, conectores USB y de antenas, así como otros componentes de mayor envergadura, encuentran su ubicación en esta capa. La elección estratégica de dónde colocar estos componentes está guiada por múltiples factores, como la necesidad de acceso, la interacción con señales de alta frecuencia y la eficiencia térmica.

Uno de los aspectos primordiales en esta capa es la cuidadosa disposición de los componentes para minimizar las longitudes de traza de señales críticas. Las longitudes de traza pueden afectar la integridad de las señales, especialmente en circuitos de alta frecuencia. Al ubicar los componentes de manera cercana y eficiente, se reducen las longitudes de traza, lo que disminuye las posibilidades de pérdida de señal o degradación del rendimiento.

Otro aspecto crucial es la reducción de las posibles interferencias electromagnéticas. Dado que los componentes en esta capa interactúan entre sí y con las señales de alta frecuencia, es esencial diseñar su disposición de manera que las interferencias electromagnéticas se minimicen. Esto se logra considerando las trayectorias de las señales y la orientación de los componentes en función de sus campos electromagnéticos.

Asimismo, esta capa puede albergar indicadores visuales, como LEDs, que brindan una retroalimentación visual directa sobre el estado del sistema. La colocación estratégica de estos elementos en la primera capa permite una observación rápida y sencilla del estado de funcionamiento.

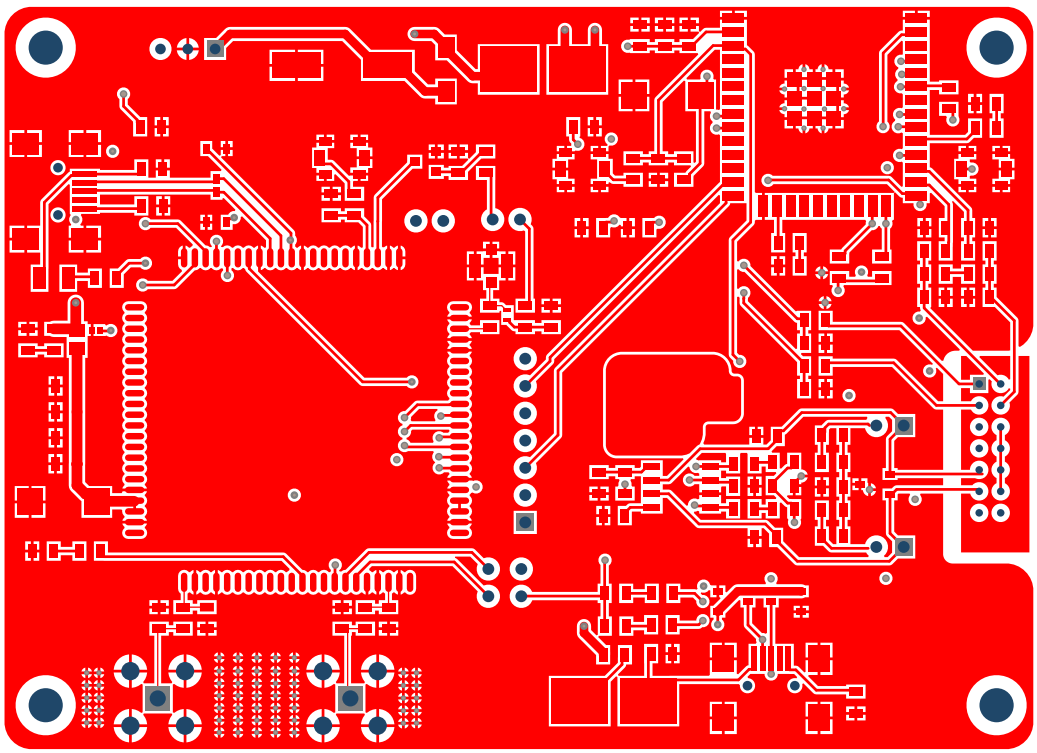


Ilustración 41: Capa Superior

3.2.2 Capa de Plano de Tierra (GND Plane)

La segunda capa de diseño en sistemas electrónicos, conocida como plano de tierra, desempeña un papel crítico en la estabilidad electromagnética y la gestión térmica del circuito. Definida como un plano continuo que abarca la totalidad de la superficie de la PCB, esta capa cumple una serie de funciones esenciales para asegurar el rendimiento óptimo y la confiabilidad del sistema.

En primer lugar, el plano de tierra se establece como una referencia de voltaje sólida y estable para las señales que circulan en la capa superior del diseño. Proporciona un nivel de voltaje de referencia contra el cual se miden las señales eléctricas. Esta referencia es crucial para garantizar la integridad de las señales, especialmente en circuitos que operan a altas frecuencias. La presencia de una referencia de voltaje sólida reduce la posibilidad de fluctuaciones y distorsiones en las señales, lo que podría comprometer el rendimiento y la precisión del sistema.

Uno de los aspectos más destacados del plano de tierra es su baja impedancia de retorno. Esto juega un papel fundamental en la minimización del ruido electromagnético. Cuando las señales eléctricas fluyen a través de las pistas de la capa superior, la corriente de retorno busca el camino de menor impedancia para regresar al punto de origen, que es el plano de tierra. La baja impedancia de retorno del plano de tierra garantiza que la corriente de retorno siga una ruta eficiente y de baja resistencia, lo que reduce la posibilidad de interferencias electromagnéticas que podrían afectar negativamente el rendimiento de circuitos sensibles o de alta frecuencia. [12]

Además de su función en la estabilidad electromagnética, el plano de tierra también desempeña un papel vital en la gestión térmica del sistema. La disipación de calor es un desafío crítico en la electrónica, ya que los componentes pueden generar calor durante su funcionamiento normal. El plano de tierra actúa como un eficiente disipador térmico al estar en contacto directo con la PCB y los componentes montados sobre ella. Esto ayuda a distribuir y disipar el calor de manera uniforme, evitando el sobrecalentamiento y contribuyendo a mantener la temperatura de los componentes dentro de límites seguros.

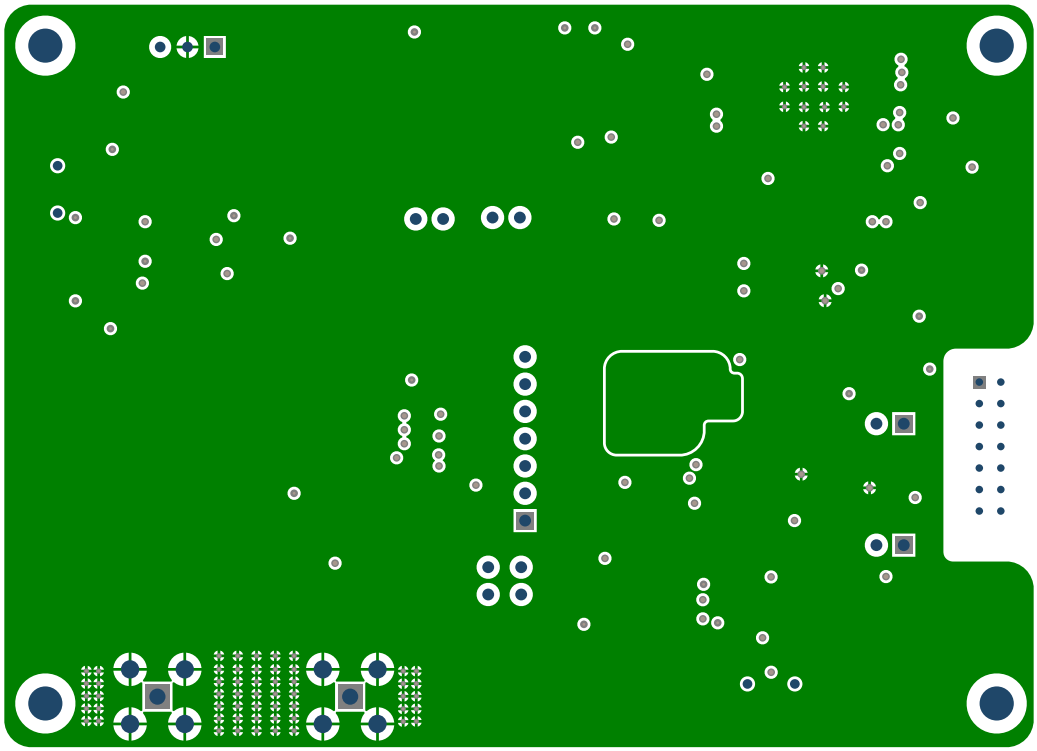


Ilustración 42: Capa plano GND

3.2.3 Capa de Alimentación (Power Plane)

La tercera capa de diseño en sistemas electrónicos se enfoca en la distribución eficiente de energía a través de

planos de alimentación. Este aspecto es crítico para garantizar el funcionamiento estable y confiable de los componentes y circuitos dentro del sistema. En esta capa, se implementa una estrategia de diseño que consiste en dividir los planos de alimentación en diferentes niveles de voltaje requeridos por el sistema.

En este caso, se mencionan varios niveles de voltaje, como 3.3V, 5V, 3.8V y 12V, que son esenciales para el funcionamiento de diferentes partes del sistema. La precisión en el establecimiento de estos planos de alimentación es fundamental para lograr una distribución eficiente de la energía. Cada nivel de voltaje tiene requisitos específicos y sensibilidades que deben ser atendidos de manera adecuada.

La distribución de energía comienza desde la fuente de alimentación y se extiende a través de rutas estratégicas a lo largo de la placa de circuito impreso (PCB) o el sustrato en el que se montan los componentes. Cada plano de alimentación se conecta de manera estratégica a los componentes correspondientes que operan a ese nivel de voltaje. Esto asegura que cada componente reciba la cantidad de energía requerida para su funcionamiento óptimo. [12]

Un aspecto importante en esta capa de diseño es la minimización de interferencias electromagnéticas entre diferentes niveles de voltaje. Cuando los niveles de voltaje se encuentran en cercanía física en la PCB, existe la posibilidad de que las señales eléctricas generadas por un nivel puedan interferir con las señales de otro nivel, lo que podría llevar a problemas de funcionamiento o degradación del rendimiento. Al separar estratégicamente los planos de alimentación y establecer barreras de aislamiento adecuadas, se reduce el riesgo de interferencias electromagnéticas no deseadas.

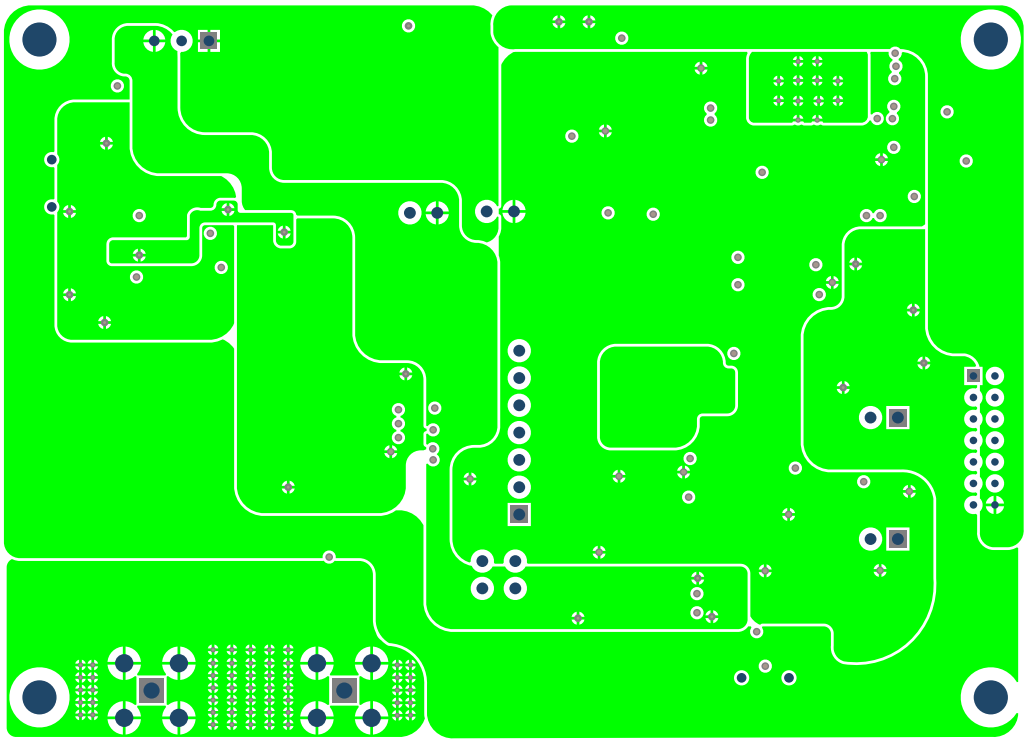


Ilustración 43: Capa Planos VCC

3.2.4 Capa de Enrutamiento (Bottom Layer)

La cuarta capa del stackup en el diseño de sistemas electrónicos cumple un papel esencial al albergar componentes adicionales y señales de baja frecuencia. Esta capa se planifica con minuciosidad para garantizar la funcionalidad óptima y la integridad de las señales, a la vez que se considera la interacción entre los elementos presentes en esta capa y las capas previas del diseño.

En esta capa, se incorporan diversos componentes que desempeñan roles específicos en el sistema. Ejemplos de estos componentes pueden ser el conector de la tarjeta SIM, el controlador CAN, optoacopladores y otras señales secundarias. Cada uno de estos elementos tiene una función única en el sistema global y requiere una ubicación estratégica para su correcto funcionamiento y su interacción con otros componentes.

El enrutamiento de las señales en esta capa se lleva a cabo con un enfoque meticuloso. Se planifica cuidadosamente para evitar cruces no deseados entre trazas y para mantener distancias adecuadas entre las rutas de señales que podrían transportar señales sensibles. La separación entre trazas es especialmente crucial para evitar interferencias electromagnéticas y crosstalk, que son fenómenos que podrían afectar negativamente la integridad de las señales y, por lo tanto, el funcionamiento del sistema. [12]

La disposición de los componentes en esta capa se realiza con una visión global del sistema y su interacción. Se considera cómo estos componentes pueden interactuar entre sí y con los elementos presentes en las capas anteriores. Esta consideración es fundamental para prevenir posibles interferencias electromagnéticas o incompatibilidades que puedan surgir debido a la proximidad de ciertos componentes. [12]

Es importante mencionar que esta cuarta capa del diseño también puede tener consideraciones térmicas, ya que algunos componentes pueden generar calor durante su funcionamiento. La disposición de estos componentes y la planificación del flujo de aire pueden contribuir a mantener una temperatura adecuada y prevenir problemas de sobrecalentamiento.

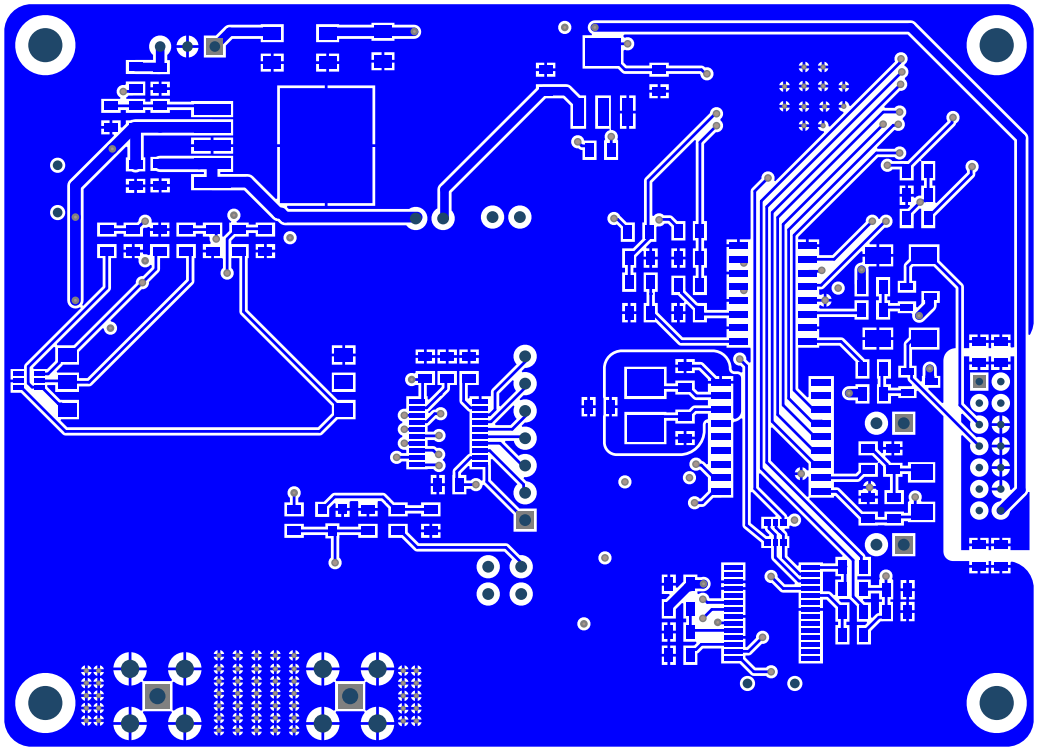


Ilustración 44: Capa Inferior

3.2.5 Aspecto del layout destacables

Tras analizar de manera general el enrutamiento de las cuatro capas que conforman la PCB, es fundamental profundizar en las consideraciones específicas que han sido implementadas en este proceso. Estas consideraciones se han centrado en optimizar la integridad de la señal, mitigar el ruido y asegurar un rendimiento óptimo del circuito.

1. Plano de Tierra para el Reloj:

   Con el objetivo de reducir la susceptibilidad a interferencias electromagnéticas y acoplar el ruido generado por el reloj, se ha integrado un plano de tierra dedicado para esta señal crítica. La disposición de un plano de tierra específico posibilita el retorno eficiente de las corrientes de alta frecuencia, contribuyendo a minimizar problemas de jitter y asegurando una sincronización precisa en todo el circuito. Además, componentes de filtrado en forma de ferritas se han insertado en la trayectoria del reloj para una mayor atenuación de las perturbaciones de alta frecuencia. [13]

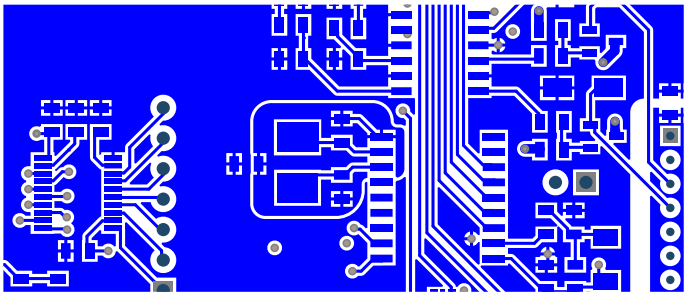


Ilustración 45: Plano de referencia propio para el rejoj

2. Biasing Shielding para Antenas:

   Las antenas principales y auxiliares de la PCB se benefician de un enfoque de protección mediante "biasing shielding". Este método implica la disposición de planos de tierra circundantes a las antenas para minimizar la exposición a interferencias externas y garantizar un desempeño confiable de las comunicaciones inalámbricas. La configuración de estos planos de tierra adicionales disminuye el impacto de las señales indeseadas y mejora la relación señal-ruido, lo cual es crucial para la calidad de las transmisiones y recepciones. [13]

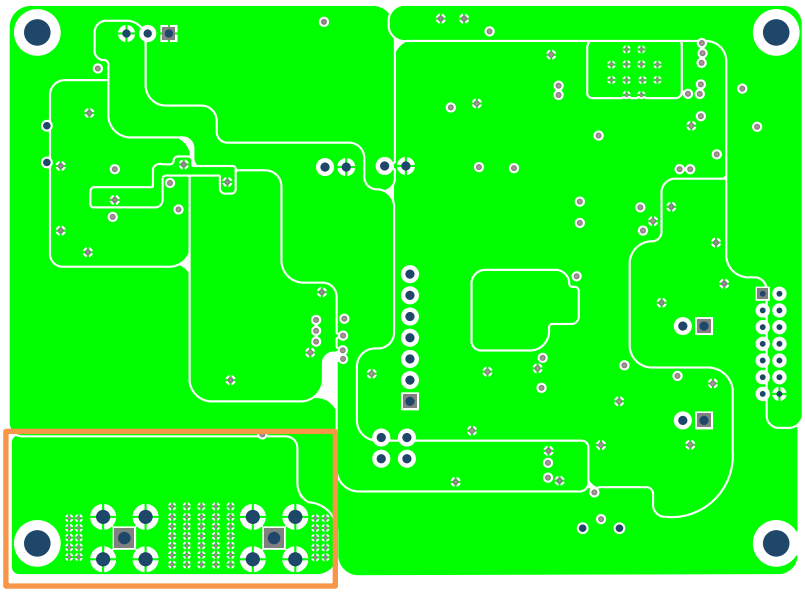


Ilustración 46: Biasing Shielding

3. Plano de Tierra "Sucia" y Ferritas:

   Para abordar las perturbaciones electromagnéticas en la región del conector, se ha implementado un plano de tierra localizado que rodea dicho conector. Este plano de tierra "sucio" actúa como una barrera contra interferencias, proporcionando un camino de retorno preferente para las corrientes de alta frecuencia. Para fortalecer aún más la supresión de ruido, se han incorporado componentes de filtrado

en forma de ferritas en las conexiones entre este plano de tierra y la PCB principal. Este enfoque reduce la propagación de ruido indeseado hacia el resto del circuito.

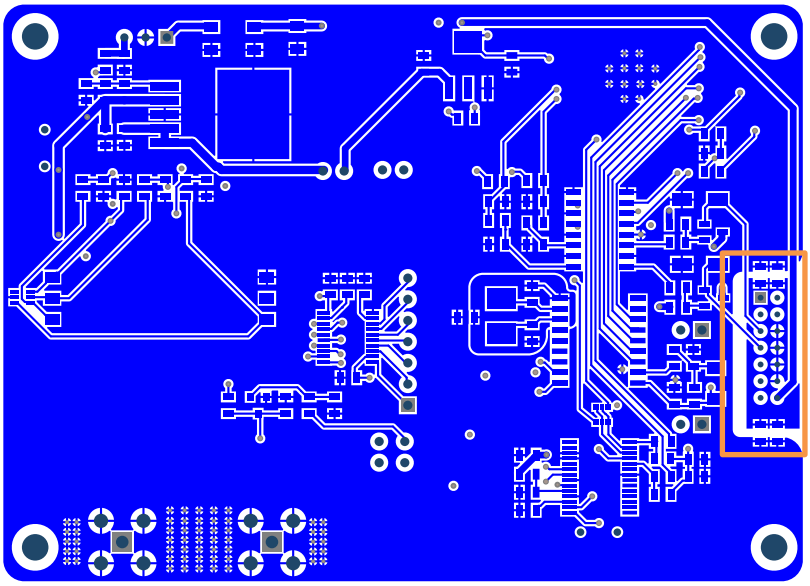


Ilustración 47: Plano de tierra sucia

4. Ubicación de Antena Integrada:

   Siguiendo las recomendaciones de diseño del fabricante del ESP32, [14] la antena integrada se ha posicionado estratégicamente para favorecer la emisión y recepción eficiente de señales. Al ubicar la antena en una posición libre, se minimizan las interferencias generadas por componentes y trazados cercanos, lo que resulta en una mejor calidad de señal y un rendimiento de comunicación más estable.

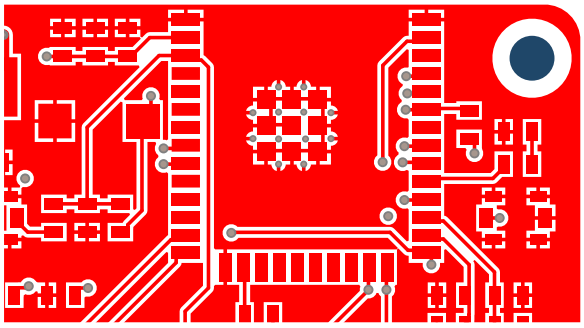


Ilustración 48: Colocación Microcontrolador

5. Ruteo Simétrico de Señales Diferenciales del CAN:

   En el caso específico de las señales diferenciales del Controller Area Network (CAN), se ha aplicado un enfoque de ruteo simétrico. Esto implica mantener un par de señales, una con inversión respecto a la otra, con trazados y longitudes idénticos. Este enfoque minimiza la diafonía y mejora la inmunidad al ruido en las comunicaciones diferenciales, esenciales para la integridad de las señales en entornos de alta interferencia electromagnética.

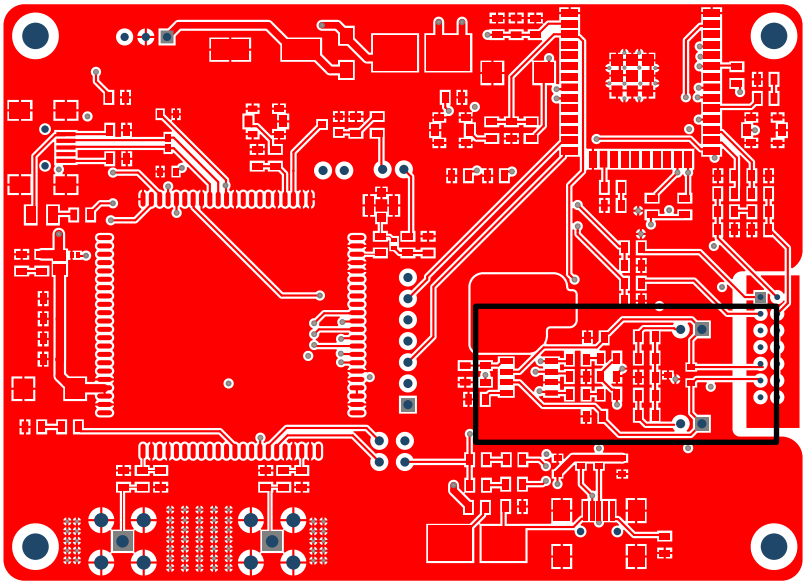


Ilustración 49: Ruteo simétrico bus CAN

6. Planos de VCC y Consideraciones de Bucles de Corriente:

   Para mantener la distribución uniforme de la alimentación, se han implementado planos de VCC estratégicamente dispuestos en la PCB. Se ha prestado especial atención a los bucles de corriente, minimizando su área y asegurando que sean lo más compactos posible. Esto reduce la inductancia y evita la generación de campos magnéticos que podrían afectar la integridad de la señal y la calidad del rendimiento del circuito.

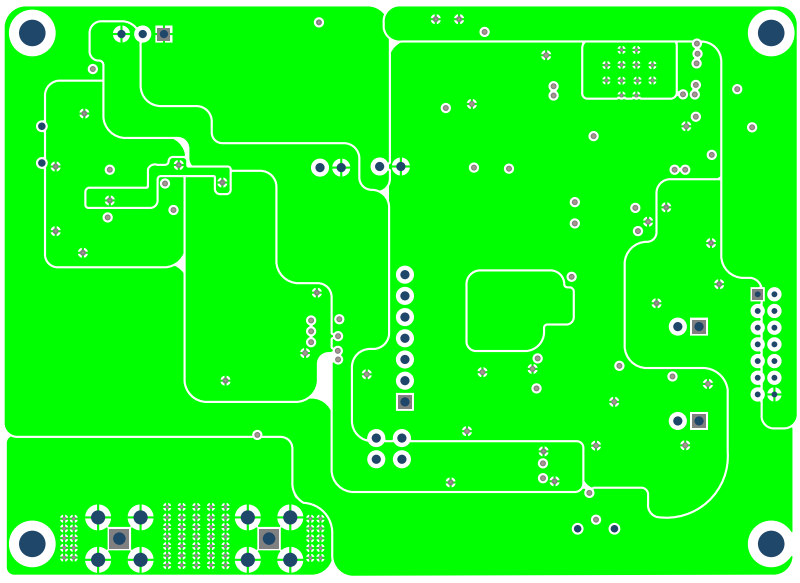


Ilustración 50: Planos de VCC

En resumen, estas consideraciones específicas en el ruteo de la PCB representan una estrategia completa para optimizar la funcionalidad, la integridad de la señal y la inmunidad al ruido en el diseño del circuito. Cada una de estas medidas ha sido implementada con el objetivo de garantizar un rendimiento excepcional y confiable en el contexto de las aplicaciones inalámbricas y de comunicación de alta frecuencia abordadas en este proyecto.

4 ARQUITECTURA SOFTWARE

4.1 Introducción

Esta PCB posee la capacidad de leer datos de una diversidad de sensores emplazados en el monoplaza, además de recolectar información vital de la red de Control de Área de Red (CAN) del vehículo. Mediante la implementación de un sistema de mensajería MQTT (Message Queuing Telemetry Transport), los datos captados se transmiten hacia un bróker centralizado, desde el cual pueden ser analizados y visualizados en tiempo real tanto a través de una Interfaz Gráfica de Usuario (GUI) de escritorio como en una aplicación móvil Android. Este enfoque no solo facilita la monitorización en tiempo real del rendimiento del monoplaza, sino que también concede la habilidad de almacenar los datos para análisis a futuro, mejorando la toma de decisiones y la optimización de los diferentes subsistemas.

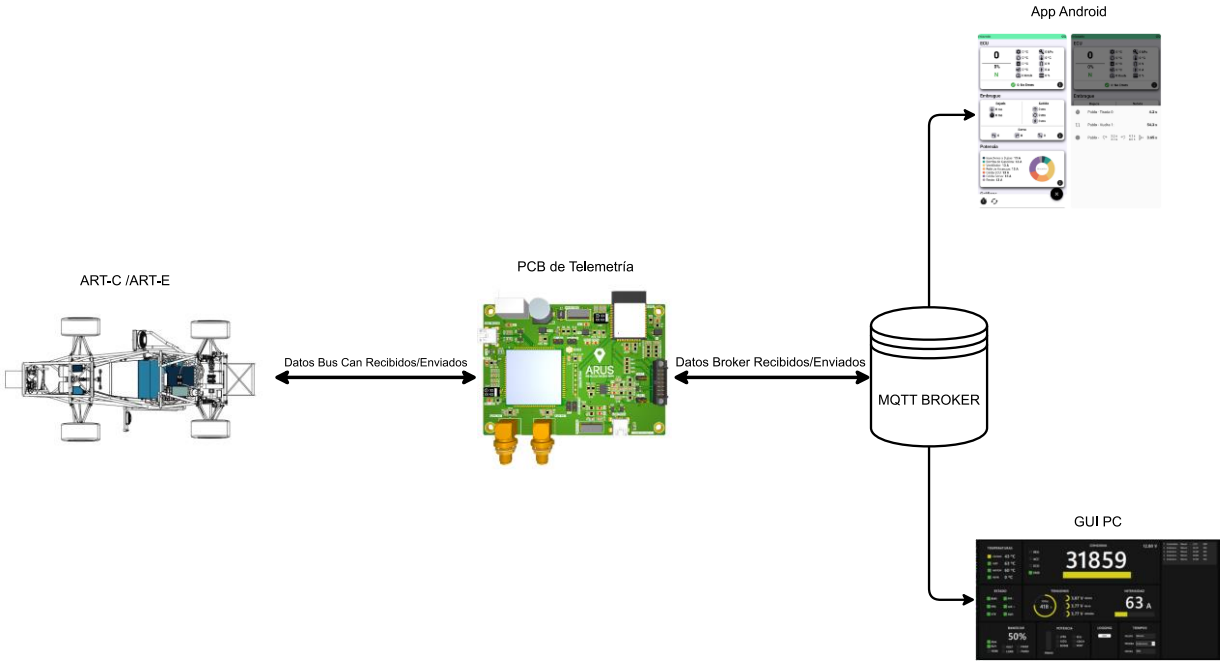


Ilustración 51: Arquitectura SW a nivel funcional

4.2 Desarrollo SW del código

Dentro de esta sección específica del documento, se despliega de manera precisa un compendio abarcador de las capacidades funcionales clave que han sido implementadas en el código integral. La finalidad primordial de este resumen es asegurar que los lectores obtengan una comprensión clara y concisa de cómo las diversas funcionalidades interactúan y operan en conjunto en el código en cuestión.

Con el propósito de permitir una exploración más exhaustiva y detallada de la implementación, se ha incorporado el código completo en el apéndice designado bajo la denominación de "Código Completo". Dicha aproximación ofrece el beneficio de otorgar a los interesados la capacidad de sumergirse en todos los aspectos técnicos con un alto nivel de profundidad.

Es importante resaltar las características funcionales más sobresalientes del código, mientras evita adentrarse en exceso en los detalles más técnicos. Al mismo tiempo, la inclusión del anexo sirve como un recurso esencial para aquellos que buscan un entendimiento aún más completo y detallado del código en su totalidad.

1. Inclusión de Librerías:

En esta sección, se incluyen las bibliotecas necesarias para el funcionamiento del código. `SoftwareSerial.h` es una biblioteca que permite la comunicación serie en pines digitales arbitrarios del microcontrolador. Esto es especialmente útil en situaciones donde se necesita más de una comunicación serie.

2. Definición de Comunicación Serial con el Módulo GSM:

   Se utiliza la biblioteca `SoftwareSerial` para crear una instancia llamada `SerialAT`, que se utiliza para comunicarse con el módulo GSM. Los pines 2 y 3 se utilizan para la comunicación RX y TX respectivamente. Esto permite enviar comandos AT y recibir respuestas del módulo LTE.

3. Selección de Modo (GPRS o WiFi) para TinyGSM:

   Con las definiciones `TINY_GSM_USE_GPRS` y `TINY_GSM_USE_WIFI`, decides qué tipo de conectividad usarás para la comunicación. En este caso, `TINY_GSM_USE_GPRS` está habilitado, lo que significa que el dispositivo se conectará a través de la red celular utilizando GPRS.

4. Definición de Credenciales y Configuración GPRS:

   En esta parte, defines los detalles necesarios para establecer la conexión GPRS. `apn`, `gprsUser` y `gprsPass` son los datos de acceso para la red celular. Estos detalles son cruciales para que el módulo se autentique en la red y pueda enviar y recibir datos.

5. Definición de Detalles MQTT:

   Aquí defines los detalles relacionados con MQTT. `broker` es la dirección IP o el nombre de dominio del servidor MQTT, mientras que `mqttUsername` y `mqttPassword` son las credenciales de autenticación para conectarse al servidor. Estos detalles permiten al dispositivo enviar y recibir mensajes a través del protocolo MQTT.

```
276   // MQTT details
277   const char* broker = "34.175.13.225";
278   const char* mqttUsername = "";  // MQTT username
279   const char* mqttPassword = "";  // MQTT password
280   const char *root_topic_subscribe = "input_ARTE";
281   const char *root_topic_publish = "telemetry/ARTE";
282
```

6. Inclusión de Librerías Específicas:

   Se incluyen las bibliotecas `TinyGsmClient.h` y `PubSubClient.h`, que son esenciales para la comunicación con el módem GSM y el servidor MQTT, respectivamente. Estas bibliotecas proporcionan funciones predefinidas para gestionar la comunicación en sus respectivos protocolos.

7. Inicialización del Módem GSM y Creación de Objetos:

   Se crea un objeto `modem` de la clase `TinyGsm`, que representa el módem GSM. Además, se crea un objeto `client` de la clase `TinyGsmClient`, que se utiliza para establecer la comunicación con el módem. También se crea un objeto `mqtt` de la clase `PubSubClient`, que se utilizará para la comunicación MQTT.

8. Configuración de Pines y Variables:

   Aquí se definen pines que se utilizarán posteriormente, como los pines para los LEDs. También se inicializan variables, como `count` para llevar un seguimiento y `lastReconnectAttempt` para gestionar los intentos de reconexión MQTT.

9. Definición de Callback de MQTT:

   La función `mqttCallback` se llama cuando se recibe un mensaje MQTT. Aquí se procesa y se imprime el tema y la carga útil del mensaje recibido. Esta es una utilidad interesante ya que permite realizar diversas funcionalidades que que mejoran el comportamiento del monoplaza en remoto.

```
311  void mqttCallback(char* topic, byte* payload, unsigned int len)
312  {
313    SerialMon.print("Message arrived [");
314    SerialMon.print(topic);
315    SerialMon.print("]: ");
316    SerialMon.write(payload, len);
317    SerialMon.println();
318
319    // Only proceed if incoming message's topic matches, is just an example
320    if (String(topic) == root_topic_subscribe)
321    {
322
323      DeserializationError error = deserializeJson(JSON_IN, payload,32);
324
325      if (error)
326      {
327      Serial.print("deserializeJson() failed: ");
328      Serial.println(error.c_str());
329      return;
330      }
331
332      enableFlag = JSON_IN["enableFlag"];
333      disableFlag = JSON_IN["disableFlag"];
334      tiempoRef = JSON_IN["tiempoRef"];
335
336    }
337  }
338
```

10. Función para Conectar a MQTT:

    La función `mqttConnect` intenta conectarse al servidor MQTT utilizando las credenciales definidas. Si la conexión falla, se reinicia el microcontrolador. Si la conexión tiene éxito, también se suscribe al tema especificado para recibir mensajes.

11. Configuración Inicial en la Función `setup()`:

    En esta sección se realizan diversas configuraciones iniciales. Se establece la comunicación serial con una velocidad de 115200 baudios. Se configuran los pines para los LEDs y se inicia el bus CAN. Luego, se configuran las comunicaciones GSM y se espera a que el dispositivo se conecte a la red celular.

12. Bucle Principal en la Función `loop()`:

    En este bucle, se verifica constantemente la conexión MQTT. Si no está conectada, se intenta reconectar cada 10 segundos. Además, se verifica el estado de un interruptor conectado al pin 17 y se lee la información del bus CAN. Dependiendo del identificador CAN, se procesan los datos y se almacenan en variables correspondientes.

13. Procesamiento y Publicación de Datos a MQTT:

    Dentro del bucle, se procesan los datos obtenidos y se almacenan en objetos JSON. Estos objetos se serializan en formato JSON y se publican a través de MQTT en intervalos regulares de tiempo. Esto permite el envío de datos en tiempo real al servidor MQTT.

4.2.1 MQTT BROKER

Un broker MQTT (Message Queuing Telemetry Transport) es un intermediario en la arquitectura de comunicación del protocolo MQTT. MQTT es un protocolo de mensajería ligero y eficiente diseñado para la comunicación entre dispositivos de Internet de las cosas (IoT) y aplicaciones que requieren una transferencia de datos rápida y confiable con baja sobrecarga.

El broker MQTT actúa como un servidor centralizado al que los dispositivos clientes se conectan para enviar y recibir mensajes. Su función principal es gestionar y enrutar los mensajes entre los dispositivos clientes, asegurando que los mensajes lleguen a los destinatarios correctos. [15]

Características del broker MQTT: [15]

1. Publicación y suscripción: El protocolo MQTT sigue un patrón de comunicación de publicación y suscripción. Los dispositivos pueden publicar mensajes en ciertos "temas" y otros dispositivos pueden suscribirse a esos temas para recibir los mensajes correspondientes.
2. Mensajes retenidos: El broker MQTT puede retener el último mensaje enviado en un tema, de modo que cuando un nuevo cliente se suscribe a ese tema, reciba inmediatamente el último mensaje publicado en él.
3. Calidad de servicio (QoS): MQTT ofrece diferentes niveles de calidad de servicio para garantizar la entrega de mensajes. Esto permite que los dispositivos elijan el nivel adecuado según la confiabilidad requerida.
4. Tópicos jerárquicos: Los tópicos en MQTT forman una estructura jerárquica, lo que facilita la organización y filtrado de mensajes.
5. Poca sobrecarga: MQTT es un protocolo de comunicación liviano y eficiente en términos de uso de ancho de banda y consumo de energía, lo que lo hace adecuado para dispositivos con recursos limitados.
6. Persistencia de sesiones: Los brokers MQTT pueden mantener sesiones para los clientes, lo que permite que los dispositivos se reconecten y continúen las comunicaciones incluso después de una desconexión temporal.

Los brokers MQTT son esenciales para el funcionamiento de sistemas IoT, ya que proporcionan la infraestructura necesaria para la comunicación entre dispositivos, la recopilación y distribución de datos y la coordinación de acciones en un entorno distribuido. Algunos ejemplos populares de brokers MQTT incluyen Mosquitto, HiveMQ, EMQ, entre otros.

4.2.2 JSON

JSON (JavaScript Object Notation) es un formato ligero de intercambio de datos que se utiliza para transmitir información estructurada entre sistemas. Es ampliamente utilizado en aplicaciones web y en el desarrollo de API debido a su sencillez y facilidad de lectura tanto para humanos como para máquinas.

Las características principales de JSON son: [16]

1. Sintaxis basada en texto: JSON utiliza una sintaxis legible por humanos, que consiste en pares clave-valor. Los datos se representan en un formato que se asemeja mucho a la notación de objetos en JavaScript.
2. Estructura de datos: Los datos en JSON están organizados en objetos (objetos) y matrices (arrays). Los objetos son colecciones de pares clave-valor encerrados entre llaves ` {} ` y las matrices son colecciones de elementos separados por comas y encerrados entre corchetes ` []`.
3. Tipos de datos compatibles: JSON admite varios tipos de datos, incluyendo cadenas de texto, números, booleanos (true o false), objetos, matrices y null (representación de la ausencia de valor).
4. Interoperabilidad: JSON es independiente del lenguaje, lo que significa que puede ser utilizado y comprendido por la mayoría de los lenguajes de programación. Esto lo convierte en un formato ideal para la comunicación entre diferentes sistemas y plataformas.
5. Facilidad de uso: La estructura simple y legible de JSON facilita su manipulación tanto para los desarrolladores como para las aplicaciones que lo procesan.

Ejemplo de un objeto JSON:

```
{
"timestamp": "2023-07-18T12:30:45.123Z",
"vehicle_id": "ABC123",
```

"speed": 85.2,
"engine_rpm": 2800,
"fuel_level": 60.5,
"engine_temperature": 98.6,
"tire_pressure": {
"front_left": 32.5,
"front_right": 32.4,
"rear_left": 32.2,
"rear_right": 32.3
},
"location": {
"latitude": 40.7128,
"longitude": -74.0060
},
"status": {
"engine_on": true,
"doors_locked": true,
"lights_on": false
}
} ```

5 INTERFACES GRÁFICAS DE USUARIO

5.1 Introducción

La competición Formula Student requiere una comprensión profunda de la telemetría para optimizar el rendimiento de los vehículos de carreras. En este documento, exploraremos cómo la incorporación de una Interfaz Gráfica de Usuario (GUI) para el análisis de los datos recolectados por la telemetría puede mejorar la evaluación del rendimiento del monoplaza y facilitar la toma de decisiones por parte del equipo.

La GUI aplicada a la telemetría en el contexto de Formula Student permite la visualización en tiempo real de datos esenciales mientras el vehículo está en la pista, ya se compitiendo o en la época de testeo. Datos como velocidad, temperatura del motor, niveles de combustible y otros parámetros se presentan de manera clara y accesible. Esto brinda a los miembros del equipo la capacidad de monitorear activamente el rendimiento y detectar posibles problemas durante las pruebas y competiciones.

La GUI facilita la representación gráfica de datos telemétricos, lo que permite a nuestro equipo analizar el rendimiento en función del tiempo. Gráficos interactivos permiten la comparación de variables como velocidad, aceleración y temperatura en diferentes puntos del neumático. Esto ayuda a identificar patrones, tendencias y correlaciones que podrían influir en la estrategia de conducción y ajustes del vehículo.

La GUI no solo proporciona información visual, sino que también puede ofrecer la posibilidad de realizar ajustes en tiempo real. Esto se debe a que los datos son publicados por MQTT a un servidor. Sin embargo, telemetría puede estar subscripta a diversos tópicos, siendo en este caso la GUI la que publica en el servidor la información que es interpretada por la PCB. Esto permite la optimización del vehículo sin requerir intervenciones físicas en el monoplaza.

Después de una sesión de pruebas o una competición, la GUI facilita el análisis exhaustivo de los datos almacenados. Esto brinda la oportunidad de revisar el rendimiento en detalle, identificar áreas de mejora y tomar decisiones informadas para futuras iteraciones del vehículo. Se pueden revisar gráficos, comparar resultados y evaluar la efectividad de los ajustes realizados.

La GUI puede ser programada para generar alertas visuales en caso de que ciertos valores excedan umbrales críticos. Esto permite reaccionar rápidamente a situaciones potencialmente peligrosas o a problemas inesperados, garantizando la seguridad y el rendimiento del monoplaza y el piloto.

5.2 GUI PC

A continuación, se detallan los parámetros mostrados en la interfaz:

1. Estado del inversor y sistema eléctrico (naranja):

   Comando par al inversor: controla la cantidad de par transmitido al motor eléctrico para ajustar la velocidad.

   Corriente DC de la batería de alta tensión: indica la corriente que fluye desde la batería hacia el inversor.

   Estado de los relés de alta tensión, precarga, BMS y R2D: informa sobre el estado de los relés críticos del sistema eléctrico y los sistemas de seguridad.

2. Sistema de frenado (púrpura):

   Presión de la línea de frenos traseras y delanteras: muestra la presión en los sistemas de frenos, permitiendo un control preciso de la desaceleración.

3. Batería de baja tensión (verde):

   Tensión de la batería de baja: supervisa el estado de la batería secundaria del monoplaza.

4. Inversor trifásico (amarillo):

Límite de potencia del inversor trifásico: controla y muestra el límite de potencia que el inversor puede suministrar al motor.

Feedback por parte del inversor para poder detectar errores de funcionamiento.

5. Dirección y conducción (rojo oscuro):

   Posición del volante (ángulo): indica el ángulo del volante, permitiendo el seguimiento preciso de la dirección.

   Modo de conducción del monoplaza: informa sobre el modo de conducción seleccionado (p. ej. normal, aceleración, ahorro de energía).

6. Temperaturas (verde claro):

   Temperaturas del motor, de las celdas, de los IGBT y del circuito de refrigeración: monitorea las temperaturas críticas del sistema, evitando el sobrecalentamiento y daños.

7. Batería de alta tensión (azul claro):

   Voltaje total de la batería: muestra la tensión total de la batería de alta tensión.

   Tensión media de cada celda: proporciona un promedio de la tensión de cada celda en la batería.

   Mínimo actual y mínimo histórico de la tensión de las celdas: alerta sobre celdas con baja tensión actual o histórica, indicando posibles problemas.

8. Neumáticos y velocidad de las ruedas (blanco):

   3 puntos de temperatura de cada neumático: proporciona la temperatura en diferentes puntos de cada neumático.

   Velocidad de cada rueda: muestra la velocidad individual de cada rueda, lo que ayuda a identificar desequilibrios o problemas de tracción.

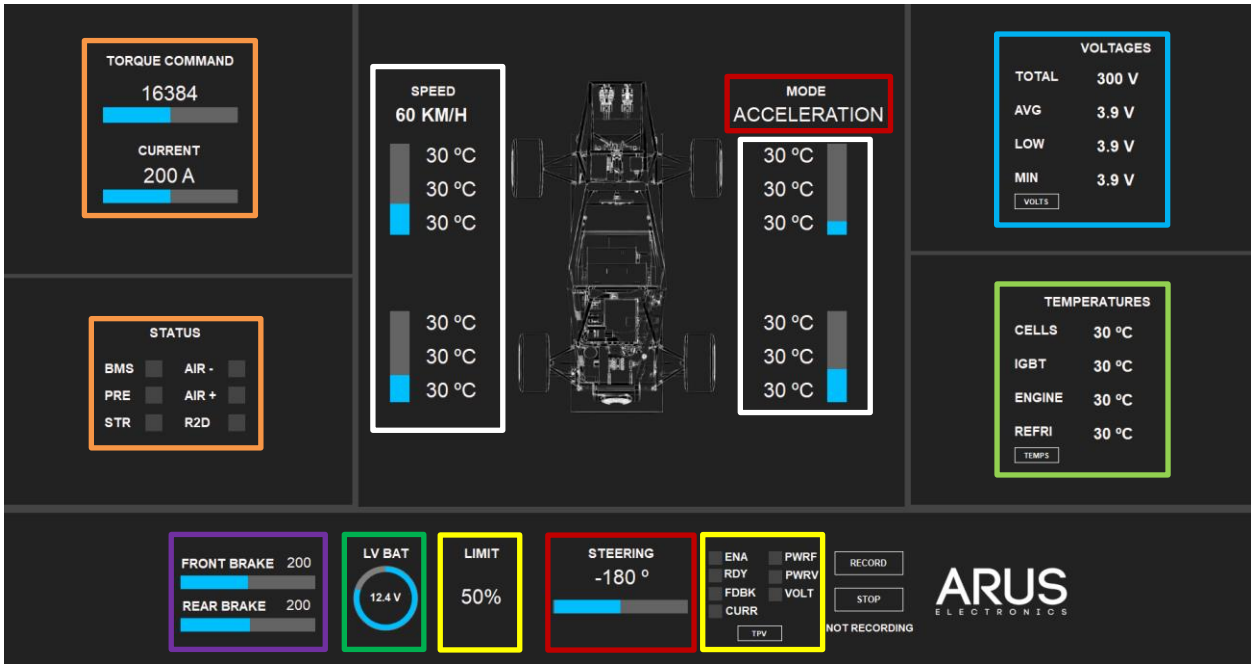


Ilustración 52: GUI PC

En el sistema de adquisición de datos, la monitorización y control de la telemetría se lleva a cabo mediante un servidor centralizado. En este servidor, se ejecuta un script que se suscribe al tópico en el cual se publica toda la información relevante del monoplaza. Este script funciona como un gestor de paquetes, recopilando los datos transmitidos y organizándolos en un archivo de formato .csv que almacena toda la información registrada.

La GUI, diseñada para interactuar con este sistema de adquisición de datos, presenta dos botones que permiten iniciar y detener la grabación de la telemetría. Estos botones ofrecen una interfaz intuitiva para que el usuario controle cuándo se inicia y se detiene la recopilación de datos en el servidor.

En resumen, el flujo de trabajo implica lo siguiente:

1. El servidor ejecuta un script que se suscribe al tópico de telemetría, donde el monoplaza publica información en tiempo real.
2. El script actúa como un gestor de paquetes, recopilando los datos transmitidos por el monoplaza.
3. Estos datos se organizan y almacenan en un archivo .CSV en el servidor, proporcionando una base de datos completa de la telemetría para su análisis posterior.
4. La GUI ofrece una forma visual de interactuar con el servidor. A través de dos botones específicos, se puede controlar cuándo comienza y termina la grabación de la telemetría:

Ilustración 53:Control de Grabación

5.3 GUI APP

La interfaz de la aplicación para Android desarrollada para monitorizar en tiempo real todos los parámetros de un monoplaza de Formula Student de combustión es una herramienta invaluable en el ámbito del automovilismo de competición. Esta aplicación brinda a los ingenieros, técnicos y pilotos una ventana digital directa hacia el rendimiento y la salud del vehículo mientras se encuentra en la pista.

Dentro de la interfaz de la aplicación, se presentan los diversos parámetros esenciales que influyen en el rendimiento del monoplaza, proporcionando una visión completa del estado y comportamiento del vehículo. Cada parámetro se representa de manera clara y organizada, lo que permite una fácil interpretación y análisis de la información en tiempo real.

Una de las características más destacadas de esta aplicación es su capacidad para ajustar la subida y bajada del embrague de forma precisa. Este control en tiempo real sobre el embrague es de gran utilidad, ya que permite afinar el rendimiento del monoplaza en diferentes situaciones de carrera. La capacidad de ajustar la respuesta del embrague puede tener un impacto significativo en la aceleración, la tracción y el rendimiento general del vehículo, permitiendo adaptarse a las cambiantes condiciones de la pista y maximizar la eficiencia en la transmisión de potencia.

Además del ajuste del embrague, la aplicación también ofrece la posibilidad de supervisar y controlar otros parámetros cruciales, como la temperatura del motor, la presión de los frenos, el consumo de combustible y más. Esta información en tiempo real es esencial para tomar decisiones informadas durante la competencia y realizar ajustes precisos que puedan marcar la diferencia en el rendimiento del monoplaza en la pista.

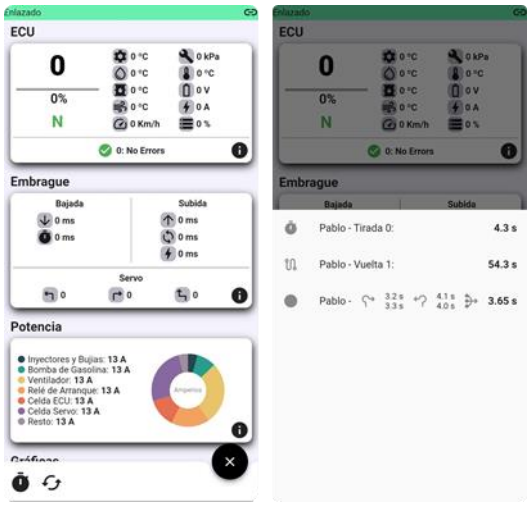


Ilustración 54: Captura aplicación Android

6 DATOS REGISTRADOS

La telemetría desempeña un papel fundamental en la optimización y desarrollo continuo de vehículos de carreras, especialmente en la categoría de coches eléctricos. La recopilación de datos en tiempo real permite a los ingenieros y estudiantes analizar el rendimiento del vehículo y la eficiencia de los sistemas clave. Además, esta información es vital para identificar áreas de mejora, realizar ajustes precisos y tomar decisiones informadas para obtener el mejor rendimiento posible en la pista

6.1 Tests de la prueba de aceleración

La prueba de aceleración en Formula Student implica que los vehículos compitan uno a uno en una pista recta de longitud fija (generalmente alrededor de 75-100 metros). Los vehículos comienzan desde una posición de detención y aceleran a fondo tan pronto como se da la señal de inicio. El objetivo es alcanzar la mayor velocidad posible en el tramo recto, lo que pone a prueba la eficiencia del sistema de propulsión, la tracción y la capacidad de aceleración del vehículo.

Esta prueba evalúa la capacidad de aceleración del vehículo y su sistema de transmisión, así como la eficiencia en la transferencia de potencia de su motor a las ruedas. Los equipos compiten por lograr los tiempos de aceleración más rápidos mientras mantienen un buen control y estabilidad del vehículo. La prueba también puede considerar la velocidad final alcanzada antes de cruzar la línea de meta

En la siguiente ilustración observamos diversas tiradas de la prueba de aceleración durante un test. Para el análisis vamos a analizar las siguientes variables:

1. Consigna: En este contexto, la "consigna" se refiere a la consigna par comandada al inversor del monoplaza.
2. Intensidad DC de la batería de alta tensión: Se refiere a la cantidad de corriente eléctrica continua (DC) que fluye a través de la batería de alta tensión del vehículo. Esta intensidad se mide en amperios (A) y es una medida importante para determinar el rendimiento y la capacidad de la batería.
3. Tensión total de la batería de alta tensión: Esta tensión es la suma de las tensiones individuales de las celdas de la batería y se mide en voltios (V). La tensión total afecta la potencia y la autonomía del vehículo.
4. Modo de conducción del vehículo: En este caso en nuestro monoplaza eléctrico tenemos 4 diferentes modos.
5. Error Bitmap: representación visual o codificada de los errores que están ocurriendo en el inversor del motor eléctrico.
6. Command Current: Es la corriente eléctrica establecida como objetivo por el sistema de control del vehículo, en este caso el inversor
7. Actual Current: Es la corriente eléctrica real que está fluyendo en ese momento a través del sistema, como la batería, el inversor y el motor eléctrico. Puede diferir de la corriente de comando debido a diversas condiciones y limitaciones del sistema.

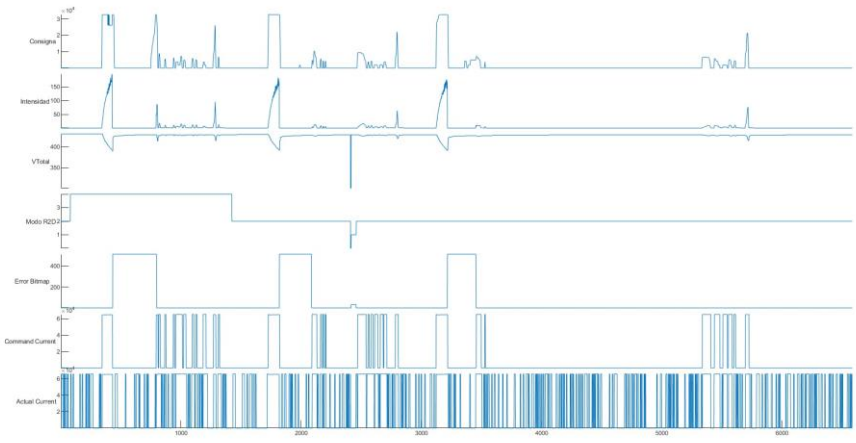


Ilustración 55: Test pruebas de aceleraación

Se pueden identificar principalmente tres series de pruebas de aceleración que evalúan los modos de aceleración y conducción estándar del monoplaza. Vamos a realizar un análisis exhaustivo de cada una de ellas de la siguiente manera:

1. Primera serie de pruebas: En esta serie, nos concentraremos en el análisis de las cuatro primeras variables representadas. En primer lugar, se destaca que la consigna alcanza un valor máximo de 32,768 (2^15). Observamos claramente que la intensidad de corriente continua (DC) que sale de la batería experimenta un aumento pronunciado, mientras que la tensión total de la batería disminuye de manera progresiva en función de la intensidad DC que sale de la batería.

2. Segunda serie de pruebas: En este caso, realizamos una serie de pruebas en el modo estándar. Observamos un comportamiento muy similar al de la serie anterior, ya que los valores de intensidad y tensión total de corriente continua (DC) son prácticamente idénticos.

3. Tercera serie de pruebas: Llevamos a cabo otra serie de pruebas en el modo estándar y se obtiene un comportamiento muy parecido al de la serie de pruebas anterior.

6.2 Energía consumida respecto a tensión total de la batería de alta tensión

En la gráfica que muestra el consumo de energía versus la tensión total de la batería, estás explorando cómo cambia el consumo de energía a medida que la tensión de la batería disminuye. Esta relación es fundamental para comprender cómo el monoplaza se comporta en términos de eficiencia y rendimiento en diferentes etapas de una carrera.

Al observar la gráfica, puedes identificar ciertos patrones y tendencias:

1. Inicio de la Curva: Donde la tensión de la batería es máxima, el consumo de energía es relativamente bajo. Esto se debe a que la batería está llena de energía y puede suministrarla eficientemente al motor eléctrico.
2. Disminución Gradual: A medida que avanzamos hacia la izquierda en la gráfica (a medida que la tensión disminuye), el consumo de energía puede comenzar a aumentar gradualmente.
3. Punto de Inflexión: En algunos puntos de la gráfica apreciamos cambios más pronunciado en la pendiente de la curva. Esto podría indicar un punto donde el motor comienza a operar en regiones menos eficientes, lo que provoca un aumento más rápido en el consumo de energía a medida que la tensión continúa disminuyendo.

4. Punto Crítico: Eventualmente, podrías llegar a un punto donde el consumo de energía aumenta significativamente en relación con la disminución de la tensión. Esto podría indicar que el motor o los sistemas eléctricos están luchando para mantener el rendimiento y están operando de manera ineficiente debido a la tensión extremadamente baja.

En tu análisis, deberías buscar estos puntos clave y tendencias en la gráfica. También puedes comparar esta información con los datos de la pista y las condiciones de la carrera para obtener una imagen completa de cómo la eficiencia y el rendimiento del monoplaza varían en función de la tensión de la batería. Esto te ayudará a tomar decisiones informadas sobre la estrategia de manejo de energía y optimizar el rendimiento en la pista.

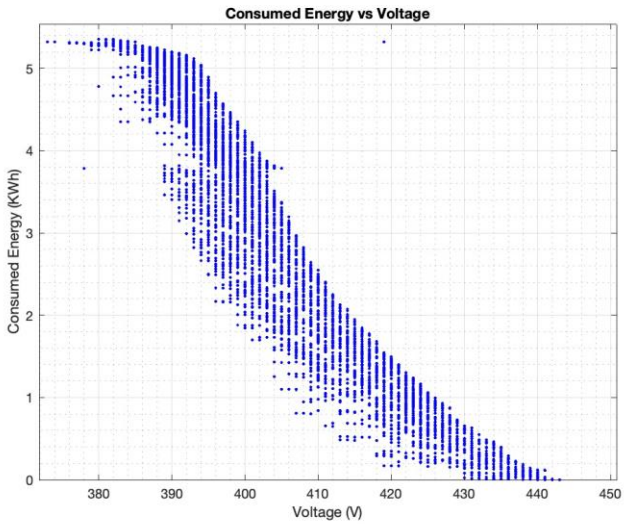


Ilustración 56: Energía consumida respecto a tensión total de la batería de alta tensión

6.3 Test de simulación de una endurance

La prueba de endurance en Formula Student es una competición en la que equipos de estudiantes diseñan, construyen y corren vehículos de carreras. En esta prueba, los vehículos deben completar una distancia de alrededor de 22 km. La carrera se realiza en relevos con al menos dos pilotos por equipo.

Los equipos deben planificar estrategias de conducción eficientes para administrar la energía de la batería o el combustible mientras maximizan la velocidad. Las detenciones en boxes para ajustes y cambios de piloto están permitidas, pero afectan la puntuación final. La prueba evalúa la confiabilidad, eficiencia y durabilidad de los vehículos, así como las habilidades de trabajo en equipo y toma de decisiones de los estudiantes.

Con este contexto, podemos observar en la ilustración XX la validación de la energía consumida de respecto al tiempo. Esta grafica es muy importante porque nos permite ajusta el tiempo por vuelta objetivo que tenemos que conseguir para asegurar acabar la prueba sin agotar la energía de la batería.

Se pueden observar principalmente tres tramos de interés:

1. Tramo Rojo: Se observa un consumo lineal hasta el cambio de piloto. Se observa que el consumo no es simétrico respecto al segundo piloto.
2. Tramo Azul: Podemos observar que el consumo permanece constante debido al cambio de piloto.
3. Tramo Verde: Observamos un consumo lineal hasta que finalmente se consume la energía completa de la batería.

En conclusión, podemos confirmar que el tiempo por vuelta ejecutado está muy por encima del tiempo objetivo.

Para solucionar esto deberíamos definir un tiempo por vuelta inferior y ajustar el consumo para que ambos pilotos consuman la mitad de la energía almacenada y así poder acabar la prueba de endurance de manera satisfactoria.

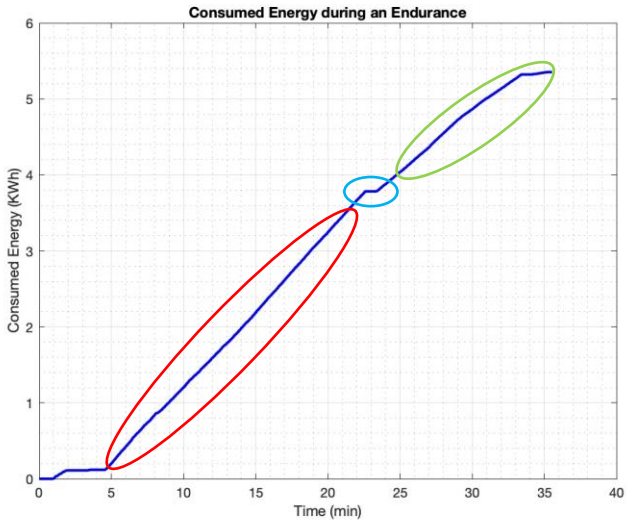


Ilustración 57: Test validación Endurance

7 CONCLUSIONES

En el marco de este Trabajo de Fin de Grado se llevó a cabo el desarrollo e implementación integral de una telemetría para dos vehículos pertenecientes al equipo de Formula Student: uno propulsado por un motor eléctrico y otro por un motor de combustión interna. A lo largo del desarrollo de este proyecto, se logró una comprensión profunda de los aspectos cruciales de la telemetría, desde el diseño del hardware hasta la implementación del software, pasando por la fabricación de la placa de circuito impreso (PCB) incluyendo la soldadura. Las conclusiones obtenidas son las siguientes:

1. Diseño Integral de Telemetría: La realización exitosa de la telemetría para ambos vehículos, incluyendo el diseño y fabricación de la PCB, representa un logro significativo. Se demostró la capacidad para abordar desafíos tanto en el ámbito del hardware como en el del software, lo que resultó en sistemas de telemetría funcionales y adaptados a las necesidades específicas de cada tipo de vehículo.
2. Desarrollo de Hardware Personalizado: La concepción y ejecución del diseño de hardware de la PCB fue un proceso altamente educativo y enriquecedor. Se exploraron aspectos tales como la selección de componentes adecuados, el enrutamiento eficiente de las trazas y la consideración de prácticas de diseño para minimizar ruidos y optimizar la integridad de la señal.
3. Fabricación de PCB: La experiencia de llevar a cabo la fabricación de las PCB en sí misma generó un valioso aprendizaje. La capacidad de transformar un diseño esquemático en una placa física funcional y lista para su implementación real en los vehículos fue una parte integral del proceso de desarrollo, brindando una comprensión más profunda de la fabricación electrónica.
4. Integración Software-Hardware: La interacción entre el software y el hardware fue un aspecto crítico en el éxito general del proyecto. La implementación de protocolos de comunicación eficientes, la adquisición precisa de datos y la visualización en tiempo real a través de interfaces de usuario fueron logros que destacaron la importancia de una colaboración fluida entre ambas disciplinas.
5. Optimización del Rendimiento: Los datos recopilados durante las pruebas en pista proporcionaron información valiosa sobre el rendimiento de ambos tipos de vehículos. Se identificaron oportunidades para mejorar la eficiencia energética, la gestión de la potencia y la respuesta del vehículo ante diferentes condiciones de pista.
6. Seguridad y Fiabilidad Reforzadas: La telemetría permitió monitorear parámetros críticos en tiempo real, lo que resultó en una detección temprana de posibles problemas y en la toma de decisiones informadas para garantizar la seguridad de los pilotos y la integridad de los vehículos.
7. Aprendizaje y Competencias Adquiridas: La realización de este proyecto me aportó una formación técnica y práctica altamente valiosa. La capacidad para afrontar desafíos complejos y para colaborar en un proyecto multidisciplinario dejó una marca perdurable en términos de habilidades y experiencia.

8 MEJORAS FUTURAS

Después de un período de desarrollo que abarca desde septiembre de 2021 hasta septiembre de 2022, hemos identificado varias áreas en las que podemos realizar mejoras. A continuación, presentamos estas mejoras de manera detallada:

En primer lugar, existe una oportunidad para simplificar la configuración eléctrica. Actualmente, tenemos dos fuentes de alimentación de 5V: una para el Modem y otra para la PCB. Al unificar estas fuentes, eliminaríamos la necesidad de emplear dos conectores mini USB para la depuración a través de comandos AT desde el software del fabricante.

Otra mejora considerable sería la capacidad de controlar el encendido y apagado del Modem utilizando un pin GPIO del microcontrolador. Aunque actualmente esto se controla mediante un jumper, permitir que el microcontrolador gestione esta señal establecería al Modem como un esclavo directo del microcontrolador principal, mejorando la coordinación entre ambos.

En relación con la depuración, hemos detectado que al conectar el Modem y establecer una conexión de puerto serie a través del FT232R, el microcontrolador entra en un ciclo de reinicio. Este comportamiento parece ser ocasionado por interferencias de señal que impactan el pin I0 del microcontrolador, generando reinicios intermitentes. Para abordar esta situación, sugerimos desconectar el jumper de 5V antes de abrir la conexión de puerto serie. Una vez establecida la conexión, es seguro reconectar el jumper sin experimentar problemas.

Otra perspectiva por considerar es que, al abrir el puerto serie, el modem emite señales RTS y DTR que podrían recibir valores altos o bajos, lo que resultaría en un reinicio continuo del microcontrolador. No obstante, este escenario precisa de una investigación más profunda para determinar la causa raíz y proponer una solución adecuada.

También exploramos la posibilidad de aprovechar el controlador CAN incorporado en el microcontrolador. Esta medida nos permitiría prescindir de un componente adicional, reduciendo costos y complejidad. Solo sería necesario contar con el transceptor para completar esta configuración.

Además, consideramos la implementación de un sistema operativo en tiempo real (RTOS) en el microcontrolador. Esta decisión nos permitiría dedicar un hilo de ejecución exclusivamente a la lectura de datos provenientes del CAN y de los sensores, mientras que otro hilo se enfocaría en la publicación de estos datos en el broker. Vale la pena destacar que hemos obtenido resultados exitosos con tasas de captura de datos de hasta 7ms.

En términos de optimización, sugerimos examinar y reducir las funcionalidades de las bibliotecas como TinyGsmclient y PubSubclient. Esto nos permitiría conservar solo las funciones esenciales para el proyecto, evitando sobrecargas innecesarias.

9 ANEXOS

9.1 Filtro anti-rebote

En la siguiente figura destacamos tres gráficas diferentes:

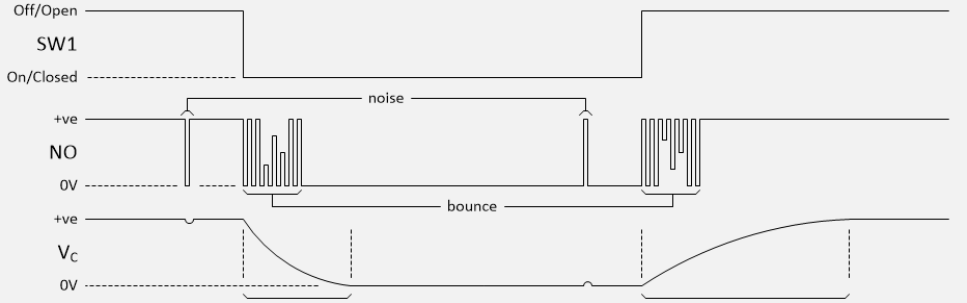


Ilustración 58: Análisis filtro anti-rebote

1. Primera gráfica: Muestra el momento en que el pulsador es pulsado. Esto podría ser un cambio de estado abrupto de "no pulsado" a "pulsado".
2. Segunda gráfica: Representa el comportamiento eléctrico de la señal del pulsador cuando se pulsa. Aquí, destacas dos aspectos: el ruido y el rebote. El ruido es cualquier interferencia no deseada en la señal, mientras que el rebote se refiere a las oscilaciones que pueden ocurrir en la señal cuando el pulsador es presionado debido a la naturaleza mecánica del interruptor.
3. Tercera gráfica: Muestra la señal eléctrica que el microcontrolador lee en su pin de entrada digital después de aplicar el filtro anti-rebote. Aquí, se ha aplicado un filtro para suavizar la señal y eliminar las fluctuaciones no deseadas causadas por el ruido y el rebote. Esto asegura que el microcontrolador reciba una señal más estable y coherente, evitando interpretaciones incorrectas de pulsaciones debido a estas interferencias.

En resumen, el filtro anti-rebote es un componente crucial para garantizar que los dispositivos electrónicos, como los microcontroladores, interpreten de manera precisa las entradas de los usuarios. Ayuda a eliminar las señales no deseadas que pudieran generar problemas en la operación del sistema.

Toda la información de este anexo ha sido extraída de [17].

9.2 Anti-resonancia y condensadores de desacople

La anti-resonancia es un fenómeno en el que la frecuencia autorresonante de dos condensadores difieren, produciendo una resonancia paralela en la región de frecuencias en la que un condensador se encuentra en la zona inducible y el otro en la zona capacitiva lo que produce un aumento de la impedancia total (idealmente, se convierte en infinitamente grande). El comportamiento de la impedancia de dos condensadores en paralelo queda resumido en la siguiente ilustración:

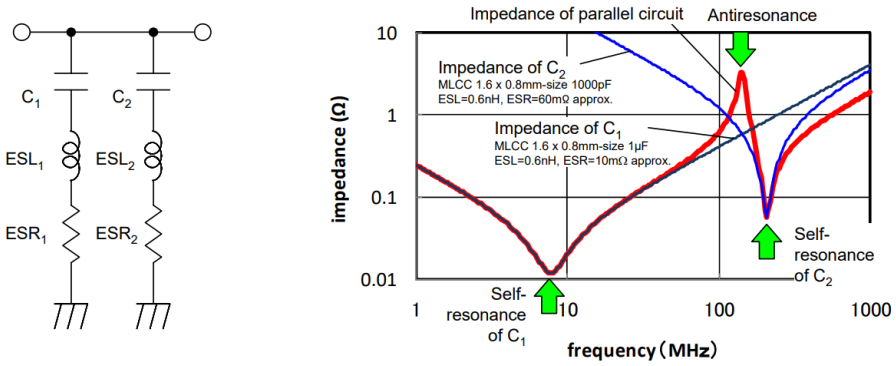


Ilustración 59: Impedancia de dos condensadores de 1uF y 1000pF en paralelo

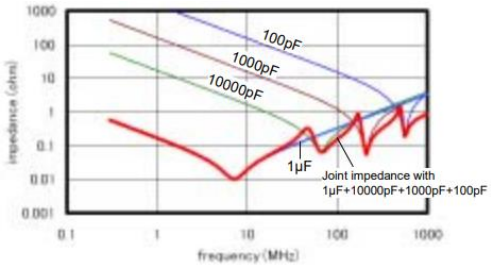


Ilustración 60: Combinación de condensadores con diferentes capacitancias

La ilustración 49 muestra una combinación de condensadores con diferentes capacitancias. Obtenemos una baja impedancia en un ancho rango de frecuencias mediante la combinación de condensadores con diferentes frecuencias resonantes (Aprovechando la característica de los condensadores en la que se obtiene una baja impedancia en la zona de auto resonancia).

Un ejemplo de impedancia conjunta cuando se utilizan cuatro condensadores a 1μF, 10000pF, 1000pF y 100pF en paralelo se muestra en la ilustración 49. La frecuencia característica de impedancia aparece en ondas, y en algunos casos su impedancia excede un condensador de 1μF a frecuencias de anti-resonancia.

La ilustración 50 muestra un caso de condensadores con la misma capacitancia en paralelo. En este caso, como indican los resultados del cálculo de la ilustración 50, los problemas de anti-resonancia no se producen con tanta frecuencia. También existe la ventaja de que es relativamente más fácil aumentar la capacitancia debido al mayor número de condensadores. Por otro lado, un mayor número de condensadores tiene las desventajas de un mayor espacio y coste.

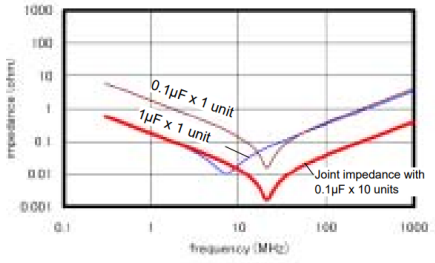


Ilustración 61: Combinación de 10 condensadores con la misma impedancia

Toda la información de este anexo se ha extraído del siguiente documento [18].

9.3 Código Completo

```
//DEFINE CAN BUS
#include <mcp_can.h>
#include <SPI.h>

long unsigned int rxId;
unsigned char len = 0;
unsigned char rxBuf[8];
unsigned char txBuf[8];

MCP_CAN CAN0(21);  // Set CS to pin 21

////////////////////////////////////////////////////

#define CAN_BMS_1                   0x220
#define CAN_BMS_2                   0x244
#define CAN_BAMOCAR_TX              0x181
#define CAN_BAMOCAR_RX              0x201
#define CAN_POTENCIA                0x251
#define CAN_R2D                     0x290
#define CAN_FRONT                   0x282
#define CAN_REAR                    0x283
#define CAN_FL_1                    0X310
#define CAN_FL_2                    0X311
#define CAN_FL_3                    0X312
#define CAN_FL_4                    0X313
#define CAN_FR_1                    0X330
#define CAN_FR_2                    0X331
#define CAN_FR_3                    0X332
#define CAN_FR_4                    0X333
#define CAN_RL_1                    0X350
#define CAN_RL_2                    0X351
#define CAN_RL_3                    0X352
#define CAN_RL_4                    0X353
#define CAN_RR_1                    0X370
#define CAN_RR_2                    0X371
#define CAN_RR_3                    0X372
#define CAN_RR_4                    0X373
#define CAN_VELOCITY                0X392

//IDs BMS 1
#define CAN_INTENSIDAD              0

//IDs BMS 2
#define CAN_STATE                   0
#define CAN_INTENSIDAD_DC           1
#define CAN_TOTAL_VDC               2
#define CAN_TEMP_MAX                3
#define CAN_MIN_VOLTAGE             4
#define CAN_MAX_VOLTAGE             5
#define CAN_STATUS_CELDAS           6

//IDs BAMOCAR RX
#define ID_CONSIGNA                 0x90
#define ID_STATUS                   0x51
#define ID_PORCENTAJE               0xC4
#define CAN_CONSIGNA_LSB            1
#define CAN CONSIGNA MSB            2

//IDs POTENCIA
#define CAN_STATUS_POTENCIA         0
#define CAN_BRAKE_LSB               1
#define CAN_BRAKE_MSB               2
#define CAN_TEMP_REFRI              3
#define CAN_BRAKE2_LSB              4
#define CAN BRAKE2 MSB              5

//IDs BAMOCAR TX
#define ID_TEMP_MOTOR               0x49
#define CAN_TEMP_MOTOR_LSB          1
#define CAN_TEMP_MOTOR_MSB          2

```

```
72  #define ID_TEMP_IGBT              0x4A
73  #define CAN_TEMP_IGBT_LSB          1
74  #define CAN_TEMP_IGBT_MSB          2
75
76  #define ID_CURRENT_LIMIT           0xC4
77  #define CAN_CURRENT_LIMIT_LSB      1
78  #define CAN_CURRENT_LIMIT_MSB      2
79
80  #define ID_STATUS_BAMOCAR          0x40
81  #define CAN_STATUS_BAMOCAR_LSB     1
82  #define CAN_STATUS_BAMOCAR_MSB     2
83
84  #define ID_ERROR_WARNING_BITMAP    0x8F
85  #define CAN_ERROR_BITMAP_LSB       1
86  #define CAN_ERROR_BITMAP_MSB       2
87  #define CAN_WARNING_BITMAP_LSB     3
88  #define CAN_WARNING_BITMAP_MSB     4
89
90  #define ID_ACTUAL_CURRENT          0x20
91  #define CAN_ACTUAL_CURRENT_LSB     1
92  #define CAN_ACTUAL_CURRENT_MSB     2
93
94  #define ID_COMMAND_CURRENT         0x26
95  #define CAN_COMMAND_CURRENT_LSB    1
96  #define CAN_COMMAND_CURRENT_MSB    2
97
98  #define ID_SPEED_ACTUAL            0x30
99  #define CAN_SPEED_ACTUAL_LSB       1
100 #define CAN_SPEED_ACTUAL_MSB       2
101
102 #define ID_FILTERED_ACTL_CURRENT        0x5f
103 #define CAN_FILTERED_ACTL_CURRENT_LSB  1
104 #define CAN_FILTERED_ACTL_CURRENT_MSB  2
105
106 #define ID_VOLTAGE_DC_BUS          0xeb
107 #define CAN_VOLTAGE_DC_BUS_LSB     1
108 #define CAN_VOLTAGE_DC_BUS_MSB     2
109
110 //IDs R2D
111 #define CAN_MODO                   0
112 #define CAN_LVBAT_LSB              1
113 #define CAN_LVBAT_MSB              2
114
115 //IDs CAN_PCB
116 #define CAN_WSL_LSB                0
117 #define CAN_WSL_MSB                1
118 #define CAN_WSR_LSB                2
119 #define CAN_WSR_MSB                3
120 #define CAN_EXTL                   4
121 #define CAN_EXTR                   5
122 #define CAN_ANALOG_LSB             6
123 #define CAN_ANALOG_MSB             7
124
125 //IDs VELOCITY
126 #define CAN_VELOCITY_X_LSB         0
127 #define CAN_VELOCITY_X_MSB         1
128 #define CAN_VELOCITY_Y_LSB         2
129 #define CAN_VELOCITY_Y_MSB         3
130
131
132 ////////////////////////////////////////////////////
133
134 //DEFINE JSON PARAMETERS
135 #define ARDUINOJSON_USE_DOUBLE 0
136 #define ARDUINOJSON_USE_LONG_LONG 0
137 #include <ArduinoJson.h>
138
139 StaticJsonDocument<1000> JSON_OUT;
140 // String input;
141
142 StaticJsonDocument<32> JSON_IN;
```

```
143
144  char envio[2048];
145
146  //ALL VARIABLES
147
148  struct DATA1
149  {
150  char statusbms;
151  unsigned int intensidad_dc;
152  int  volttotal;
153  char tempceldas;
154  char voltbaja;
155  char voltminima;
156  char status_celdas;
157  bool precarga;
158  int intensidad = 200;
159  } bmsCAN;
160
161  struct DATA2
162  {
163  int lvbat;
164  unsigned char modo;
165  } r2dCAN;
166
167  struct DATA3
168  {
169  unsigned int statusbamocar;
170  unsigned int actualcurrent;
171  unsigned int commandcurrent;
172  unsigned int errorbitmap;
173  unsigned int warningbitmap;
174  int currentlimit;
175  int porcentaje;
176  int tempmotor;
177  int tempigbt;
178  int consigna;
179  int16_t speedactual;
180  uint16_t filtered_actual_current;
181  uint16_t voltage_dc_bus;
182
183  } bamocarCAN;
184
185  struct POTENCIA_CAN
186  {
187  char statuspotencia;
188  int freno;
189  int freno2;
190  int temprefri;
191  } potenciaCAN;
192
193  struct FRONT_CAN
194  {
195  int wsl;
196  int wsr;
197  int extl;
198  int extr;
199  int giro;
200  } frontCAN;
201
202  struct REAR_CAN
203  {
204  int wsl;
205  int wsr;
206  int extl;
207  int extr;
208  int lvbat;
209  } rearCAN;
210
211  struct TEMP_TYRE
212  {
213  int fl_1;
```

```
214   int fl_2;
215   int fl_3;
216   int fr_1;
217   int fr_2;
218   int fr_3;
219   int rl_1;
220   int rl_2;
221   int rl_3;
222   int rr_1;
223   int rr_2;
224   int rr_3;
225   } tempTyre;
226
227   struct IMU
228   {
229   int velocity;
230   }imu;
231
232   bool CAN_status=0;
233   unsigned long ref_tiempoEnvio=0;
234   unsigned long currenttime=millis();
235   bool disableFlag = 0;
236   bool enableFlag = 0;
237   int tiempoRef = 10;
238
239   // Select your modem:
240   #define TINY_GSM_MODEM_SIM7600 //
241
242
243   // Set serial for debug console (to the Serial Monitor, default speed 115200)
244   #define SerialMon Serial
245
246   // Set serial for AT commands (to the module)
247   // In this case we use ESP32 so we have to use Serial1
248   #define SerialAT Serial1
249
250   /* We use ESP32 in this case
251   // or Software Serial on Uno, Nano
252   #else
253   #include <SoftwareSerial.h>
254   SoftwareSerial SerialAT(2, 3);  // RX, TX
255   #endif
256   */
257
258   // See all AT commands, if wanted
259    //#define DUMP_AT_COMMANDS
260
261   // Define the serial console for debug prints, if needed
262   #define TINY_GSM_DEBUG SerialMon
263
264   //Use this define to choose between GPRS or WIFI:
265   #define TINY_GSM_USE_GPRS true
266   #define TINY_GSM_USE_WIFI false
267
268   // set GSM PIN, if any
269   #define GSM_PIN ""
270
271   // Your GPRS credentials, if any
272   const char apn[] = "live.vodafone.com"; // APN use https://wiki.apnchanger.org
273   const char gprsUser[] = "wap@wap";
274   const char gprsPass[] = "wap125";
275
276   // MQTT details
277   const char* broker = "34.175.13.225";
278   const char* mqttUsername = "";  // MQTT username
279   const char* mqttPassword = "";  // MQTT password
280   const char *root_topic_subscribe = "input_ARTE";
281   const char *root_topic_publish = "telemetry/ARTE";
282
283
284   #include <TinyGsmClient.h>
```

```
285  #include <PubSubClient.h>
286
287  //This mode is just to debugg with the modem
288  #ifdef DUMP_AT_COMMANDS
289  #include <StreamDebugger.h>
290  StreamDebugger debugger(SerialAT, SerialMon);
291  TinyGsm        modem(debugger);
292  #else
293  TinyGsm        modem(SerialAT);
294  #endif
295
296  TinyGsmClient client(modem);
297  PubSubClient  mqtt(client);
298
299  #define MODEM_TX             14
300  #define MODEM_RX             12
301
302  char msg[25];
303
304  long count=0;// MQTT details
305
306  uint32_t lastReconnectAttempt = 0;
307
308  const int ledPin = 13;
309  const int ledPin1 = 5;
310
311  void mqttCallback(char* topic, byte* payload, unsigned int len)
312  {
313    SerialMon.print("Message arrived [");
314    SerialMon.print(topic);
315    SerialMon.print("]: ");
316    SerialMon.write(payload, len);
317    SerialMon.println();
318
319    // Only proceed if incoming message's topic matches, is just an example
320    if (String(topic) == root_topic_subscribe)
321    {
322
323      DeserializationError error = deserializeJson(JSON_IN, payload,32);
324
325      if (error)
326      {
327      Serial.print("deserializeJson() failed: ");
328      Serial.println(error.c_str());
329      return;
330      }
331
332      enableFlag = JSON_IN["enableFlag"];
333      disableFlag = JSON_IN["disableFlag"];
334      tiempoRef = JSON_IN["tiempoRef"];
335
336    }
337  }
338
339  boolean mqttConnect()
340  {
341    SerialMon.print("Connecting to ");
342    SerialMon.print(broker);
343
344    // Connect to MQTT Broker
345    //boolean status = mqtt.connect("GsmClientTest");
346
347    // Or, if you want to authenticate MQTT:
348    boolean status = mqtt.connect("GsmClientName", "mqtt_user", "mqtt_pass");
349
350    if (status == false)
351    {
352      SerialMon.println(" fail");
353      ESP.restart();
354      return false;
355    }
```

```
356
357     SerialMon.println(" success");
358     mqtt.subscribe(root_topic_subscribe);
359
360     return mqtt.connected();
361
362   }
363
364   #include <Arduino.h>
365
366   void setup()
367   {
368     // Set console baud rate
369     SerialMon.begin(115200);
370
371     delay(1000);
372
373     pinMode (ledPin, OUTPUT);
374     pinMode (ledPin1, OUTPUT);
375
376     //SETUP CAN BUS//////////////////////////
377     if(CAN0.begin(MCP_STDEXT, CAN_500KBPS, MCP_16MHZ) == CAN_OK) Serial.print("MCP2515
        Init Okay!!\r\n");
378     else Serial.print("MCP2515 Init Failed!!\r\n");
379     pinMode(17, INPUT);                          // Setting pin 22 for INT input
380
381     Serial.println("MCP2515 Library Mask & Filter Example...");
382
383     CAN0.setMode(MCP_NORMAL);
384     ////////////////////////////////////////
385     delay(100);
386
387     // !!!!!!!!!!!
388     // Set your reset, enable, power pins here
389     // !!!!!!!!!!!
390
391     SerialMon.println("Wait...");
392
393     // Set GSM module baud rate, IMPORTANT in the case of usign ESP32 we have to use
        this function with these specific parameters
394     SerialAT.begin(3686400, SERIAL_8N1, MODEM_TX, MODEM_RX);
395
396     delay(6000);
397
398     // Restart takes quite some time
399     // To skip it, call init() instead of restart()
400     SerialMon.println("Initializing modem...");
401     modem.restart();
402     // modem.init();
403
404     delay(1000);
405
406     String modemInfo = modem.getModemInfo();
407     SerialMon.print("Modem Info: ");
408     SerialMon.println(modemInfo);
409
410     uint8_t LTE = 2; // Lo indica el la guia de usuario del SerialAT
411
412     //AÑADIDO POR MI
413     modem.setNetworkMode(LTE);
414
415     delay(6000);
416
417     SerialMon.print("Network Mode: ");
418
419     SerialMon.println(modem.getNetworkMode());
420
421   ////////////////////////////////////////////////
422
423     // Unlock your SIM card with a PIN if needed
424     if (GSM_PIN && modem.getSimStatus() != 3) { modem.simUnlock(GSM_PIN); }
```

```
425
426     SerialMon.print("Waiting for network...");
427     if (!modem.waitForNetwork())
428     {
429       SerialMon.println(" fail");
430       delay(10000);
431       return;
432     }
433     SerialMon.println(" success");
434
435     if (modem.isNetworkConnected()) { SerialMon.println("Network connected"); }
436
437     // GPRS connection parameters are usually set after network registration
438     SerialMon.print(F("Connecting to "));
439     SerialMon.print(apn);
440     if (!modem.gprsConnect(apn, gprsUser, gprsPass))
441     {
442       SerialMon.println(" fail");
443       delay(10000);
444       return;
445     }
446     else
447     {
448
449     SerialMon.println(" success");
450
451     }
452
453     if (modem.isGprsConnected()) { SerialMon.println("GPRS connected"); }
454
455     // MQTT Broker setup
456     mqtt.setServer(broker, 1883);
457     mqtt.setCallback(mqttCallback);
458     mqtt.setKeepAlive(600);
459     mqtt.setSocketTimeout(600);
460   }
461
462   void loop()
463   {
464
465     if (!mqtt.connected())
466     {
467       SerialMon.println("=== MQTT NOT CONNECTED ===");
468       // Reconnect every 10 seconds
469       uint32_t t = millis();
470       if (t - lastReconnectAttempt > 10000L)
471       {
472         lastReconnectAttempt = t;
473         if (mqttConnect()) { lastReconnectAttempt = 0; }
474       }
475       delay(100);
476       return;
477     }
478
479
480
481     if(digitalRead(17)==0)
482     {
483       CAN status=1;
484     }
485     else
486     {
487       CAN status=0;
488     }
489
490       digitalWrite (ledPin, HIGH);    // turn on the LED
491
492       CAN0.readMsgBuf(&rxId, &len, rxBuf); // Read data: len = data length, buf = data
          byte(s)
493
494       if((unsigned long)rxId==(unsigned long)CAN_BMS_1)
```

```
495         {
496           bmsCAN.intensidad=(rxBuf[CAN_INTENSIDAD]);
497         }
498 
499         else if((unsigned long)rxId==(unsigned long)CAN_BMS_2)
500         {
501             bmsCAN.statusbms=(rxBuf[CAN_STATE]);
502             bmsCAN.intensidad_dc=rxBuf[CAN_INTENSIDAD_DC];
503             bmsCAN.volttotal=rxBuf[CAN_TOTAL_VDC];
504             bmsCAN.tempceldas=rxBuf[CAN_TEMP_MAX];
505             bmsCAN.voltminima=rxBuf[CAN_MIN_VOLTAGE];
506             bmsCAN.voltbaja=rxBuf[CAN_MAX_VOLTAGE];
507             bmsCAN.status_celdas=rxBuf[CAN_STATUS_CELDAS];
508         }
509 
510         else if((unsigned long)rxId==(unsigned long)CAN_BAMOCAR_RX)
511         {
512             if(rxBuf[0]==ID_CONSIGNA)
513             {
514             bamocarCAN.consigna=(int)((rxBuf[CAN_CONSIGNA_MSB] << 8 | rxBuf[
                CAN_CONSIGNA_LSB]));
515             }
516 
517             else if(rxBuf[0]==ID_PORCENTAJE)
518             {
519               bamocarCAN.porcentaje=(int)((rxBuf[CAN_CONSIGNA_MSB] << 8 | rxBuf[
                  CAN_CONSIGNA_LSB]));
520             } else {
521               bamocarCAN.porcentaje = 0;
522             }
523 
524         }
525 
526         else if((unsigned long)rxId==(unsigned long)CAN_BAMOCAR_TX)
527         {
528 
529             if(rxBuf[0]==ID_TEMP_MOTOR)
530             {
531             bamocarCAN.tempmotor=(int)((rxBuf[CAN_TEMP_MOTOR_MSB] << 8 | rxBuf[
                CAN_TEMP_MOTOR_LSB]));
532 
533             //bamocarCAN.tempmotor=((bamocarCAN.tempmotor*(0.01901))-179);
534             }
535 
536             else if(rxBuf[0]==ID_TEMP_IGBT)
537             {
538             bamocarCAN.tempigbt=(int)((rxBuf[CAN_TEMP_IGBT_MSB] << 8 | rxBuf[
                CAN_TEMP_IGBT_LSB]));
539 
540             //bamocarCAN.tempigbt=((bamocarCAN.tempigbt*(9.393/1000))-153);
541             }
542 
543             else if(rxBuf[0]==ID_ERROR_WARNING_BITMAP)
544             {
545             bamocarCAN.errorbitmap=(int)((rxBuf[CAN_ERROR_BITMAP_MSB] << 8 | rxBuf[
                CAN_ERROR_BITMAP_LSB]));
546 
547             bamocarCAN.warningbitmap=(int)((rxBuf[CAN_WARNING_BITMAP_MSB] << 8 | rxBuf[
                CAN_WARNING_BITMAP_LSB]));
548             }
549 
550             else if(rxBuf[0]==ID_STATUS_BAMOCAR)
551             {
552             bamocarCAN.statusbamocar=(int)((rxBuf[CAN_STATUS_BAMOCAR_MSB] << 8 | rxBuf[
                CAN_STATUS_BAMOCAR_LSB]));
553             }
554 
555             else if(rxBuf[0]==ID_CURRENT_LIMIT)
556             {
557             bamocarCAN.currentlimit=(int)((rxBuf[CAN_CURRENT_LIMIT_MSB] << 8 | rxBuf[
                CAN_CURRENT_LIMIT_LSB]));
```

```
558             }
559
560             else if(rxBuf[0]==ID_ACTUAL_CURRENT)
561             {
562             bamocarCAN.actualcurrent=(int)((rxBuf[CAN_ACTUAL_CURRENT_MSB] << 8 | rxBuf[
                CAN_ACTUAL_CURRENT_LSB]));
563             }
564
565             else if(rxBuf[0]==ID_COMMAND_CURRENT)
566             {
567             bamocarCAN.commandcurrent=(int)((rxBuf[CAN_COMMAND_CURRENT_MSB] << 8 | rxBuf[
                CAN_COMMAND_CURRENT_LSB]));
568             }
569
570             else if(rxBuf[0]==ID_SPEED_ACTUAL)
571             {
572             bamocarCAN.speedactual=(int16_t)((rxBuf[CAN_SPEED_ACTUAL_MSB] << 8 | rxBuf[
                CAN_SPEED_ACTUAL_LSB]));
573             }
574
575             else if(rxBuf[0]==ID_FILTERED_ACTL_CURRENT)
576             {
577             bamocarCAN.filtered_actual_current=(uint16_t)((rxBuf[
                CAN_FILTERED_ACTL_CURRENT_MSB] << 8 | rxBuf[CAN_FILTERED_ACTL_CURRENT_LSB]));
578             }
579
580             else if(rxBuf[0]==ID_VOLTAGE_DC_BUS)
581             {
582             bamocarCAN.voltage_dc_bus=(uint16_t)((rxBuf[CAN_VOLTAGE_DC_BUS_MSB] << 8 |
                rxBuf[CAN_VOLTAGE_DC_BUS_LSB]));
583             }
584
585         }
586
587         else if((unsigned long)rxId==(unsigned long)CAN_POTENCIA)
588         {
589
590             potenciaCAN.statuspotencia=rxBuf[CAN_STATUS_POTENCIA];
591             potenciaCAN.freno=(int)((rxBuf[CAN_BRAKE_MSB] << 8 | rxBuf[CAN_BRAKE_LSB]));
592             potenciaCAN.freno2=(int)((rxBuf[CAN_BRAKE2_MSB] << 8 | rxBuf[CAN_BRAKE2_LSB
                ]));
593             potenciaCAN.temprefri=rxBuf[CAN_TEMP_REFRI];
594
595         }
596
597         else if((unsigned long)rxId==(unsigned long)CAN_FRONT)
598         {
599
600             frontCAN.wsl=(int)(rxBuf[CAN_WSL_MSB] << 8 | rxBuf[CAN_WSL_LSB]);
601             frontCAN.wsr=(int)(rxBuf[CAN_WSR_MSB] << 8 | rxBuf[CAN_WSR_LSB]);
602             frontCAN.extl=rxBuf[CAN_EXTL];
603             frontCAN.extr=rxBuf[CAN_EXTR];
604             frontCAN.giro=(int)(rxBuf[CAN_ANALOG_MSB] << 8 | rxBuf[CAN_ANALOG_LSB]);
605
606         }
607
608         else if((unsigned long)rxId==(unsigned long)CAN_REAR)
609         {
610
611             rearCAN.wsl=(int)(rxBuf[CAN_WSL_MSB] << 8 | rxBuf[CAN_WSL_LSB]);
612             rearCAN.wsr=(int)(rxBuf[CAN_WSR_MSB] << 8 | rxBuf[CAN_WSR_LSB]);
613             rearCAN.extl=rxBuf[CAN_EXTL];
614             rearCAN.extr=rxBuf[CAN_EXTR];
615             rearCAN.lvbat=(int)(rxBuf[CAN_ANALOG_MSB] << 8 | rxBuf[CAN_ANALOG_LSB]);
616         }
617
618         else if((unsigned long)rxId==(unsigned long)CAN_R2D)
619         {
620             r2dCAN.modo=rxBuf[CAN_MODO];
621             r2dCAN.lvbat=(int)((rxBuf[CAN_LVBAT_MSB] << 8 | rxBuf[CAN_LVBAT_LSB]));
622             //r2dCAN.lvbat=r2dCAN.lvbat*0.0596562;
```

```
623         }
624
625         else if((unsigned long)rxId==(unsigned long)CAN_FL_1)
626         {
627             tempTyre.fl_1=((((rxBuf[0]<<8 | rxBuf[1]) + (rxBuf[2]<<8 | rxBuf[3]) + (rxBuf[
                4]<<8 | rxBuf[5]) + (rxBuf[6]<<8 | rxBuf[7])) / 4) / 10 - 200);
628         }
629
630         else if((unsigned long)rxId==(unsigned long)CAN_FL_2)
631         {
632             tempTyre.fl_2=((((rxBuf[0]<<8 | rxBuf[1]) + (rxBuf[2]<<8 | rxBuf[3]) + (rxBuf[
                4]<<8 | rxBuf[5]) + (rxBuf[6]<<8 | rxBuf[7])) / 4) / 10 - 200) / 2;
633         }
634
635         else if((unsigned long)rxId==(unsigned long)CAN_FL_3)
636         {
637             tempTyre.fl_2+=((((rxBuf[0]<<8 | rxBuf[1]) + (rxBuf[2]<<8 | rxBuf[3]) + (rxBuf
                [4]<<8 | rxBuf[5]) + (rxBuf[6]<<8 | rxBuf[7])) / 4) / 10 - 200) / 2;
638         }
639
640         else if((unsigned long)rxId==(unsigned long)CAN_FL_4)
641         {
642             tempTyre.fl_3=((((rxBuf[0]<<8 | rxBuf[1]) + (rxBuf[2]<<8 | rxBuf[3]) + (rxBuf[
                4]<<8 | rxBuf[5]) + (rxBuf[6]<<8 | rxBuf[7])) / 4) / 10 - 200);
643         }
644
645         else if((unsigned long)rxId==(unsigned long)CAN_FR_1)
646         {
647             tempTyre.fr_1=((((rxBuf[0]<<8 | rxBuf[1]) + (rxBuf[2]<<8 | rxBuf[3]) + (rxBuf[
                4]<<8 | rxBuf[5]) + (rxBuf[6]<<8 | rxBuf[7])) / 4) / 10 - 200);
648         }
649
650         else if((unsigned long)rxId==(unsigned long)CAN_FR_2)
651         {
652             tempTyre.fr_2=((((rxBuf[0]<<8 | rxBuf[1]) + (rxBuf[2]<<8 | rxBuf[3]) + (rxBuf[
                4]<<8 | rxBuf[5]) + (rxBuf[6]<<8 | rxBuf[7])) / 4) / 10 - 200) / 2;
653         }
654
655         else if((unsigned long)rxId==(unsigned long)CAN_FR_3)
656         {
657             tempTyre.fr_2+=((((rxBuf[0]<<8 | rxBuf[1]) + (rxBuf[2]<<8 | rxBuf[3]) + (rxBuf
                [4]<<8 | rxBuf[5]) + (rxBuf[6]<<8 | rxBuf[7])) / 4) / 10 - 200) / 2;
658         }
659
660         else if((unsigned long)rxId==(unsigned long)CAN_FR_4)
661         {
662             tempTyre.fr_3=((((rxBuf[0]<<8 | rxBuf[1]) + (rxBuf[2]<<8 | rxBuf[3]) + (rxBuf[
                4]<<8 | rxBuf[5]) + (rxBuf[6]<<8 | rxBuf[7])) / 4) / 10 - 200);
663         }
664
665         else if((unsigned long)rxId==(unsigned long)CAN_RL_1)
666         {
667             tempTyre.rl_1=((((rxBuf[0]<<8 | rxBuf[1]) + (rxBuf[2]<<8 | rxBuf[3]) + (rxBuf[
                4]<<8 | rxBuf[5]) + (rxBuf[6]<<8 | rxBuf[7])) / 4) / 10 - 200);
668         }
669
670         else if((unsigned long)rxId==(unsigned long)CAN_RL_2)
671         {
672             tempTyre.rl_2=((((rxBuf[0]<<8 | rxBuf[1]) + (rxBuf[2]<<8 | rxBuf[3]) + (rxBuf[
                4]<<8 | rxBuf[5]) + (rxBuf[6]<<8 | rxBuf[7])) / 4) / 10 - 200) / 2;
673         }
674
675         else if((unsigned long)rxId==(unsigned long)CAN_RL_3)
676         {
677             tempTyre.rl_2+=((((rxBuf[0]<<8 | rxBuf[1]) + (rxBuf[2]<<8 | rxBuf[3]) + (rxBuf
                [4]<<8 | rxBuf[5]) + (rxBuf[6]<<8 | rxBuf[7])) / 4) / 10 - 200) / 2;
678         }
679
680         else if((unsigned long)rxId==(unsigned long)CAN_RL_4)
681         {
682             tempTyre.rl_3=((((rxBuf[0]<<8 | rxBuf[1]) + (rxBuf[2]<<8 | rxBuf[3]) + (rxBuf[
```

```
            4]<<8 | rxBuf[5]) + (rxBuf[6]<<8 | rxBuf[7])) / 4) / 10 - 200);
683     }
684
685     else if((unsigned long)rxId==(unsigned long)CAN_RR_1)
686     {
687         tempTyre.rr_1=((((rxBuf[0]<<8 | rxBuf[1]) + (rxBuf[2]<<8 | rxBuf[3]) + (rxBuf[
            4]<<8 | rxBuf[5]) + (rxBuf[6]<<8 | rxBuf[7])) / 4) / 10 - 200);
688     }
689
690     else if((unsigned long)rxId==(unsigned long)CAN_RR_2)
691     {
692         tempTyre.rr_2=((((rxBuf[0]<<8 | rxBuf[1]) + (rxBuf[2]<<8 | rxBuf[3]) + (rxBuf[
            4]<<8 | rxBuf[5]) + (rxBuf[6]<<8 | rxBuf[7])) / 4) / 10 - 200) / 2;
693     }
694
695     else if((unsigned long)rxId==(unsigned long)CAN_RR_3)
696     {
697         tempTyre.rr_2+=((((rxBuf[0]<<8 | rxBuf[1]) + (rxBuf[2]<<8 | rxBuf[3]) + (rxBuf
            [4]<<8 | rxBuf[5]) + (rxBuf[6]<<8 | rxBuf[7])) / 4) / 10 - 200) / 2;
698     }
699
700     else if((unsigned long)rxId==(unsigned long)CAN_RR_4)
701     {
702         tempTyre.rr_3=((((rxBuf[0]<<8 | rxBuf[1]) + (rxBuf[2]<<8 | rxBuf[3]) + (rxBuf[
            4]<<8 | rxBuf[5]) + (rxBuf[6]<<8 | rxBuf[7])) / 4) / 10 - 200);
703     }
704
705     else if((unsigned long)rxId==(unsigned long)CAN_VELOCITY)
706     {
707         imu.velocity=(int)sqrt((((rxBuf[CAN_VELOCITY_X_MSB] << 8 | rxBuf[
            CAN_VELOCITY_X_LSB]) / 100) ^ 2) + (((rxBuf[CAN_VELOCITY_Y_MSB] << 8 | rxBuf[
            CAN_VELOCITY_Y_LSB]) / 100) ^ 2) * 3.6);
708     }
709
710
711     if(enableFlag)
712     {
713       txBuf[0] = 0x51;
714       txBuf[1] = 0x00;
715       txBuf[2] = 0x00;
716
717       CAN0.sendMsgBuf(CAN_BAMOCAR_RX, 0, 8, txBuf);
718       enableFlag = 0;
719     }
720
721     if(disableFlag)
722     {
723       txBuf[0] = 0x51;
724       txBuf[1] = 0x04;
725       txBuf[2] = 0x00;
726
727       CAN0.sendMsgBuf(CAN_BAMOCAR_RX, 0, 8, txBuf);
728
729       disableFlag = 0;
730     }
731
732     currenttime=millis();
733
734     if (((unsigned long)(currenttime - ref_tiempoEnvio)) >= tiempoRef)
735     {
736
737     //SERIALIZE JSON
738
739               JSON_OUT["tiempo"] = millis();
740
741               JSON_OUT["current"] = bmsCAN.intensidad;
742               JSON_OUT["volt_total"] = bmsCAN.volttotal;
743               JSON_OUT["volt_baja"] = bmsCAN.voltbaja;
744               JSON_OUT["volt_minima"] = bmsCAN.voltminima;
745               JSON_OUT["temp_celdas"] = bmsCAN.tempceldas;
746               JSON_OUT["statusbms"] = bmsCAN.statusbms;
```

```
747
748                 JSON_OUT["mode"] = r2dCAN.modo;
749                 //JSON_OUT["volt_lvbat"] = r2dCAN.lvbat;
750
751                 //JSON_OUT["statuspotencia"] = potenciaCAN.statuspotencia;
752                 JSON_OUT["front_brake"] = potenciaCAN.freno;
753                 JSON_OUT["temp_refri"] = potenciaCAN.temprefri;
754
755                 JSON_OUT["velocity"] = imu.velocity;
756
757                 JSON_OUT["temp_motor"] = bamocarCAN.tempmotor;
758                 JSON_OUT["command_torque"] = bamocarCAN.consigna;
759                 JSON_OUT["temp_igbt"] = bamocarCAN.tempigbt;
760                 JSON_OUT["statusbamocar"] = bamocarCAN.statusbamocar;
761                 JSON_OUT["errorbitmap"] = bamocarCAN.errorbitmap;
762                 JSON_OUT["warningbitmap"] = bamocarCAN.warningbitmap;
763                 JSON_OUT["actual_current"] = bamocarCAN.actualcurrent;
764                 JSON_OUT["command_current"] = bamocarCAN.commandcurrent;
765                 JSON_OUT["current_limit"] = bamocarCAN.currentlimit;
766
767                 JSON_OUT["wheelspeed_fl"] = frontCAN.wsl;
768                 JSON_OUT["wheelspeed_fr"] = frontCAN.wsr;
769                 JSON_OUT["ext_fl"] = frontCAN.extl;
770                 JSON_OUT["ext_fr"] = frontCAN.extr;
771                 JSON_OUT["steeringwheel"] = frontCAN.giro;
772
773                 JSON_OUT["wheelspeed_rl"] = rearCAN.wsl;
774                 JSON_OUT["wheelspeed_rr"] = rearCAN.wsr;
775                 JSON_OUT["ext_rl"] = rearCAN.extl;
776                 JSON_OUT["ext_rr"] = rearCAN.extr;
777                 JSON_OUT["volt_lvbat"] = rearCAN.lvbat;
778
779
780                 //JSON_OUT["velocidad_motor"] = bamocarCAN.speedactual;
781                 //JSON_OUT["intensidad_cc"] = bamocarCAN.filtered_actual_current;
782                 //JSON_OUT["voltaje_dc_bus"] = bamocarCAN.voltage_dc_bus;
783
784                 JSON_OUT["porcentaje"] = bamocarCAN.porcentaje;
785
786                 JSON_OUT["temp_fl_1"] = tempTyre.fl_1;
787                 JSON_OUT["temp_fl_2"] = tempTyre.fl_2;
788                 JSON_OUT["temp_fl_3"] = tempTyre.fl_3;
789
790                 JSON_OUT["temp_fr_1"] = tempTyre.fr_1;
791                 JSON_OUT["temp_fr_2"] = tempTyre.fr_2;
792                 JSON_OUT["temp_fr_3"] = tempTyre.fr_3;
793
794                 JSON_OUT["temp_rl_1"] = tempTyre.rl_1;
795                 JSON_OUT["temp_rl_2"] = tempTyre.rl_2;
796                 JSON_OUT["temp_rl_3"] = tempTyre.rl_3;
797
798                 JSON_OUT["temp_rr_1"] = tempTyre.rr_1;
799                 JSON_OUT["temp_rr_2"] = tempTyre.rr_2;
800                 JSON_OUT["temp_rr_3"] = tempTyre.rr_3;
801
802
803       serializeJson(JSON_OUT, envio);
804
805       mqtt.publish(root_topic_publish,envio);
806
807       mqtt.loop();
808
809       ref_tiempoEnvio = currenttime;
810       }
811
812
813   }
```

Referencias

[1] Wikipedia, «Bus CAN,» [En línea]. Available: https://es.wikipedia.org/wiki/Bus_CAN.

[2] Rohde & Schwarz, «Qué es UART,» R&S®Essentials | Principios básicos del osciloscopio digital y las sondas, [En línea]. Available: ¿Qué es UART?.

[3] Z. Peterson, «Resource Altium,» 4 March 2021. [En línea]. Available: Z. Peterson, «The mysterious 50 ohm impedance: Where it came from and why we use it», Altium, jul. 2023, [En línea]. Disponible en: https://resources.altium.com/p/mysterious-50-ohm-impedance-where-it-came-and-why-we-use-it.

[4] T. Power, «TSR2 Datasheet,» November 2022. [En línea]. Available: https://www.tracopower.com/int/tsr2-datasheet.

[5] T. Instruments, «LM1117MP-3.3 Datasheet,» February 2020. [En línea]. Available: https://www.ti.com/lit/gpn/lm1117.

[6] T. Instruments, «LP3856-ADJ Datasheet,» September 2003. [En línea]. Available: https://www.ti.com/lit/gpn/lp3856-adj.

[7] FTDI, «FT232R USB UART IC Datasheet - FTDI,» [En línea]. Available: https://ftdichip.com/wp-content/uploads/2020/08/DS_FT232R.pdf.

[8] Toshiba, «TLP293-4,» May 2019. [En línea]. Available: https://toshiba.semicon-storage.com/info/TLP293-4_datasheet_en_20190520.pdf?did=15287&prodName=TLP293-4.

[9] OnSemi, «FDN304P-D Datasheet,» March 2022. [En línea]. Available: https://www.onsemi.com/download/data-sheet/pdf/fdn304p-d.pdf.

[10] Espressif, «esp32wroom32d & esp32wroom32u - Espressif Systems,» [En línea]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32d_esp32-wroom-32u_datasheet_en.pdf.

[11] SimCOM, «SIM7600E-H Datasheet,» 2022. [En línea]. Available: https://www.simcom.com/product/SIM7600X.html.

[12] R. Hu, PCB Design and Layout Fundamentals for EMC, 978-1082079252, 2019.

[13] D. P. R. Wilson, The Circuit Designer's Companion 4th Edition, Newnes, 2017.

[14] Espressif Systems, «ESP32 Hardware Design Guidelines - Espressif Systems,» 2022. [En línea]. Available:

https://www.espressif.com/sites/default/files/documentation/esp32_hardware_design_guidelines_en.pdf.

[15] «MQTT Essentials,» HiveMQ, [En línea]. Available: https://www.hivemq.com/mqtt-essentials/.

[16] JSON.org, «Introducing JSON,» [En línea]. Available: https://www.json.org/json-en.html.

[17] Digi-Key, «Cómo implementar el rebote de hardware para interruptores y relés cuando el rebote de software no es apropiado,» 2021. [En línea]. Available: https://www.digikey.es/es/articles/how-to-implement-hardware-debounce-for-switches-and-relays.

[18] muRATA, «Decoupling for digital ICs,» 2010. [En línea]. Available: https://www.murata.com/~/media/webrenewal/support/library/catalog/products/emc/emifil/c39e.ashx?la=en-us.