



DESARROLLO DE UN SISTEMA DE TELEMETRÍA PARA AUTOMOCIÓN

Grado en Ingeniería de la Automoción

UNIVERSIDAD DE VIC
UNIVERSIDAD CENTRAL DE CATALUÑA

Granollers, Junio de 2023

Trabajo Final de Grado

GERARDO ANIBAL PETROCHE CLAVIJO

Tutor: David Arcos Gutiérrez

Resumen

Título: *Desarrollo de un sistema de telemetría para automoción.*

Autor: Gerardo Anibal Petroche Clavijo

Cotutor: David Arcos Gutiérrez (UVic)

Fecha: junio de 2023

Palabras claves: Electrónica, programación, python, microcontrolador ESP-32, telemetría.

La motivación de realizar este proyecto surge del afán de investigar y adquirir conocimientos sobre la telemetría y la tecnología emergente de telecomunicación LoRa™. Una motivación adicional para emprender este proyecto es la oportunidad de poder aplicar los conceptos adquiridos a lo largo del grado en ingeniería de la automoción. En el desarrollo de este proyecto, se presentan explicaciones sobre el diseño de los sistemas electrónicos, tanto para la unidad de control receptora como para la unidad de control transmisora. También se abordan los cálculos necesarios para ajustar los valores de los componentes, se realiza el diseño de la PCB utilizando Allegro ORCAD, se lleva a cabo la programación del software para las unidades de control que conforman el sistema de telemetría, y se desarrolla la interfaz gráfica en Python para mostrar los datos en tiempo real.

Mediante el diseño del sistema de telemetría, se ha conseguido establecer una comunicación confiable en entorno urbano de hasta 19 milisegundos de latencia a una distancia máxima de 1173 metros. Lamentablemente, debido a limitaciones de tiempo, no se ha podido llegar a implementar la representación gráfica de los datos recibidos en tiempo real, ni desarrollar una carcasa en forma de carenado que integre la unidad de control transmisora con el vehículo de donde se adquieren los datos.

Este trabajo de final de grado se presenta como un desarrollo del sistema de telemetría, por lo tanto, se anima al lector a continuar con el trabajo con innovaciones y mejoras como las propuestas descritas en el apartado de anexos.

Summary

Title: *Development of a telemetry system for the automotive industry.*

Author: Gerardo Anibal Petroche Clavijo

Supervisor: David Arcos Gutiérrez (UVic)

Date: June 2023

Keywords: *Electronics, software coding, python, ESP-32 microcontroller, telemetry.*

The motivation behind this Project stems from the desire to investigate and acquire knowledge about telemetry and emerging LoRa™ telecommunications technology. An additional motivation to undertake this project is the opportunity to apply the acquaintance obtained during this automotive engineering degree. During the development of this project, explanations of the design of the electronic systems for both the receiving control unit and the transmitting control unit are presented. The calculations necessary to adjust the component values are also explained, the PCB design is carried out using Allegro ORCAD, the programming of the software for the control units that make up the telemetry system is carried out, and the graphical user interface is developed in Python to display the data in real time.

Through the design of the telemetry system, it has been possible to establish reliable communication in an urban environment with a latency of up to 19 milliseconds at a maximum distance of 1173 meters. Unfortunately, due to time constraints, it has not been possible to implement the graphical representation of the data received in real time, neither to develop a fairing-shaped housing that integrates the transmitter control unit with the vehicle from which the data is acquired.

This final degree project is presented as a development of the telemetry system; therefore, the reader is encouraged to continue the work with innovations and improvements such as the proposals described in the annexes section.

Índice de contenidos

1.	Introducción.....	1
1.1	Objetivo del proyecto.....	2
2.	Tecnología Lo-Ra para la telemetría en la automoción.....	3
2.1.	Funcionamiento de la tecnología Lo-Ra.....	3
2.2.	Integración de Lo-Ra para la telemetría en automoción.	4
2.3.	Análisis del vehículo y elección de datos a transmitir.....	5
3.	Diseño electrónico y del software para el sistema de telemetría.....	7
3.1.	Diseño del hardware	7
3.1.1.	Diagrama de bloques del sistema	8
3.1.2.	Circuitos tipo	9
3.1.3.	PCB.....	27
3.2.	Diseño del software.....	31
3.2.1.	Lógica de la máquina de estados y unidad de control transmisora	31
3.2.2.	Multitarea y acciones en paralelo	33
3.2.3.	Comunicación con el módulo RYLR998 y comandos AT.....	35
3.2.4.	Codificación de los datos.....	37
3.2.5.	Recepción de datos y unidad de control receptora	39
3.3.	Aplicación de escritorio Graphical User Interface.....	42
4.	Coste del proyecto	47
5.	Resultados y conclusiones	48
6.	Bibliografía	51
7.	Anexos.....	i
7.1.	Diseño mecánico	i
7.2.	Pruebas.....	v

7.2.1.	Distancia máxima de funcionamiento.....	v
7.2.2.	Latencia entre módulos.....	x
7.2.3.	Overflow de variables.....	xii
7.2.4.	Análisis del convertidor de potencia de la unidad de control transmisora.....	xv
7.2.5.	Consumo de la unidad de control transmisora	xx
7.3.	Limitaciones y mejoras que realizar en proyectos futuros	xxiv
7.3.1.	Secuencia de programación del microcontrolador ESP-32:	xxiv
7.3.2.	Pistas USB:	xxvii
7.3.3.	Conexionados transistores “Through Hole”	xxviii
7.3.4.	Footprint conector original Kawasaki “Through Hole”	xxxi
7.3.5.	Layout y dimensiones generales PCB ECU Transmisora.....	xxxi
7.3.6.	Encendido del microcontrolador ESP-32 en la PCB Transmisora.....	xxxii
7.3.7.	LDO ECU Receptora	xxxiv
7.4.	Códigos de programación.....	XXXV
7.5.	Esquemas de los circuitos electrónicos diseñados.....	xxxvi
7.5.1.	ECU Transmisora	xxxvi
7.5.2.	ECU Receptora.....	xl
7.6.	Lista de materiales.....	xli
7.6.1.	Unidad de control transmisora.....	xli
7.6.2.	Unidad de control receptora	xlii
7.7.	Comandos AT RYLR998.....	xliii
7.8.	Datasheets.....	
	Transistor BJT: BC33716TA	
	Transistor MOSFET de CANAL-P: IRF9530NPBF.....	.li
7.9.	Otros documentos.....	lviii

Índice de acrónimos:

GPS: *Global Positioning System*

LoRa: *Long Range.*

IoT: *Internet Of Things*

GUI: *Graphical User Interface*

V2X: *Vehicle to Everything*

Wi-Fi: *Wireless Fidelity*

BLE: *Bluetooth Low Energy*

ECU: *Electronic Control Unit*

CC: *Centímetros cúbicos (volumen)*

KDS: *Kawasaki Diagnostics System*

KBPS: *Kilo Bits Per Second*

MBPS: *Mega Bits Per Second*

ADC: *Analogic to Digital Converter*

MB: *Megabyte*

RTOS: *Real Time Operating System*

UART: *Universal Asynchronous Receiver-Transmitter*

PCB: *Printed Circuit Board*

PC: *Personal Computer*

USB: *Universal Serial Bus*

RPM: *Revolution Per Minute*

uC: *Microcontroller*

GND: *Ground*

GPIO: *General Purpose Input/Output*

KVL: *Kirchoff's Voltage Law*

mA: *mili-Amperios (corriente)*

VCC: *Voltaje de Corriente Continua*

VBAT: *Voltaje de Batería*

IC: *Integrated Circuit*

BJT: *Bipolar Junction Transistor*

NPN: *Negative-Positive-Negative Transistor*

MOSFET: *Metal-Oxide-Semiconductor Field-Effect Transistor*

PMOS: *Positive-Channel Metal Oxide Semiconductor*

LED: *Light Emitting Diode*

LDO: *Low-Dropout (regulator)*

ms: *milisegundos (tiempo)*

FIFO: *First In, First Out*

ASCII: *American Standard Code for Information Interchange*

TVS: *Transient Voltage Suppressor*

ESD: *Electrostatic Discharge*

Índice de figuras:

1. LORA TRABAJA DONDE EL Wi-Fi Y LA TECNOLOGÍA CELULAR NO	3
2. REPRESENTACIÓN DEL MÓDULO ELECTRÓNICO INTEGRADO EN LA MOTO	4
3. SEÑAL DE COMUNICACIÓN K-LINE.....	5
4. RECORTE DEL ESQUEMA ELÉCTRICO DE LA MOTOCICLETA.....	6
5. MICROCONTROLADOR ESP WROOM-32D	7
6. MÓDULO LoRa RYLR998	7
7. DIAGRAMA DE BLOQUES DEL SISTEMA DE TELEMETRÍA	8
8. CIRCUITO DE ACONDICIONAMIENTO PARA SEÑALES DIGITALES SIMPLIFICADO CON LAS MALLAS RESPECTIVAS PARA EL CÁLCULO DE KVL	10
9. CIRCUITO DE ACONDICIONAMIENTO PARA SEÑALES DIGITALES CON VALORES DE REFERENCIA.....	10
10. SIMULACIÓN SOBRE CIRCUITO DIGITAL CON CONDICIONES DE LA HIPÓTESIS 1 (VBAT=14 V).....	11
11. SIMULACIÓN SOBRE CIRCUITO DIGITAL CON CONDICIONES DE LA HIPÓTESIS 2 (VBAT=10 V).....	11
12. SIMULACIÓN SOBRE CIRCUITO DIGITAL CON CONDICIONES DE LA HIPÓTESIS 3 (VBAT=16 V).....	11
13. REPRESENTACIÓN DEL MONTAJE DEL CIRCUITO DIGITAL SOBRE PROTOBOARD	12
14 CIRCUITO DE ACONDICIONAMIENTO DEL VELOCÍMETRO CON VALORES DE REFERENCIA	13
15 CIRCUITO DE ACONDICIONAMIENTO DEL TACÓMETRO CON VALORES DE REFERENCIA.....	13
16. CIRCUITO DE ACONDICIONAMIENTO PARA LA SEÑAL DEL VELOCÍMETRO SIMPLIFICADO CON LAS MALLAS RESPECTIVAS PARA EL CÁLCULO DE KVL.....	14
17. CIRCUITO DE ACONDICIONAMIENTO PARA LA SEÑAL DEL TACÓMETRO SIMPLIFICADO CON LAS MALLAS RESPECTIVAS PARA EL CÁLCULO DE KVL.....	14
18. SIMULACIÓN SOBRE EL CIRCUITO DE ACONDICIONAMIENTO DEL VELOCÍMETRO	15
19. SIMULACIÓN SOBRE EL CIRCUITO DE ACONDICIONAMIENTO DEL TACÓMETRO	15
20. REPRESENTACIÓN DEL MONTAJE DEL CIRCUITO DE ACONDICIONAMIENTO DEL VELOCÍMETRO SOBRE PROTOBOARD	16
21. REPRESENTACIÓN DEL MONTAJE DEL CIRCUITO DE ACONDICIONAMIENTO DEL TACÓMETRO SOBRE PROTOBOARD .	16
22 CAPTURA DEL OSCILOSCOPIO DONDE SE OBSERVA LA DISTORSIÓN	17

23. CIRCUITO DE ACONDICIONAMIENTO DE LA SEÑAL DE COMBUSTIBLE CON VALORES DE REFERENCIA.....	18
24. CIRCUITO DE ACONDICIONAMIENTO DE LA SEÑAL DE COMBUSTIBLE SIMPLIFICADO CON LAS MALLAS RESPECTIVAS PARA EL CÁLCULO DE KVL.....	18
25. SIMULACIÓN SOBRE EL CIRCUITO DE ACONDICIONAMIENTO DE LA SEÑAL DE COMBUSTIBLE CON LA TENSIÓN MÁXIMA.....	19
26. REPRESENTACIÓN DEL MONTAJE DEL CIRCUITO DE ACONDICIONAMIENTO DE LA SEÑAL DE COMBUSTIBLE SOBRE PROTOBOARD	19
27. CIRCUITO PARA CONTROLAR CARGAS DE POTENCIA Y PROTECCIÓN DE LOS INTERRUPTORES DE POTENCIA.....	20
28. CIRCUITO DE ACONDICIONAMIENTO PARA MUESTRAR LA TENSIÓN DE BATERÍA CON VALORES DE REFERENCIA....	21
29. REPRESENTACIÓN DEL MONTAJE DEL CIRCUITO DE ACONDICIONAMIENTO PARA MUESTRAR LA TENSIÓN DE BATERÍA	23
30. CIRCUITO DE ACTIVACIÓN DE UN ACTUADOR LED CON VALORES DE REFERENCIA	25
31. REPRESENTACIÓN DEL MONTAJE DEL CIRCUITO DE ACTIVACIÓN DE UN ACTUADOR LED SOBRE PROTOBOARD	25
32. BREADBOARD	27
33. PCB.....	27
34. SÍMBOLO CREADO PARA EL COMPONENTE LM2673	28
35. FOOTPRINT CREADO PARA EL COMPONENTE LM2673.....	28
36. PCB TRANSMISORA: POTENCIA	29
37. PCB TRANSMISORA: CONECTOR USB, CP2102, ESP-32 Y RYLR998.....	29
38. PCB RECEPTORA: CONECTOR MÓDULO RYLR998, MICROCONTROLADOR ESP-32, BRIDGE UART A USB Y CONECTOR USB	30
39. DIAGRAMA LÓGICO DE LA MÁQUINA DE ESTADOS ECU SENDER	32
40. CÓDIGO DE UN ESTADO NO CONTEMPLADO	32
41. CÓDIGO FUENTE DE LA FUNCIÓN PARA EL MUESTREO DE DATOS “TASKDATASAMPLING”	33
42. CÓDIGO FUENTE DE LA FUNCIÓN PARA LA TRANSMISIÓN DE DATOS “TASKDATASENDIND”	34
43. DIAGRAMA DE BLOQUES RYLR998	35
44. CÓDIGO FUENTE DE LOS PARÁMETROS CONFIGURADOS EN EL MÓDULO RECEPTOR LORA.....	36
45. CÓDIGO FUENTE DE LOS PARÁMETROS CONFIGURADOS EN EL MÓDULO DE TRANSMISIÓN LORA.	36

46. CÓDIGO FUENTE DE LA CONFIGURACIÓN DEL MÓDULO LORA RYLR998	36
47. REPRESENTACIÓN BIT A BIT DE LA CODIFICACIÓN DE LOS DATOS EN UNA SOLA VARIABLE.....	38
48. REPRESENTACIÓN DE UN BUFFER CIRCULAR.....	39
49. TRAMA DE EJEMPLO RECIBIDA MEDIANTE LORA.....	39
50. DATALOG DE DATOS RECIBIDOS POR ECU RECEPTORA.....	40
51. CÓDIGO FUENTE USADO PARA LA DEPURACIÓN DEL VECTOR LINEAL.....	41
52. GUI TELEMETRIA – PANTALLA PRINCIPAL	42
53. GUI TELEMETRIA – PANTALLA DE MENÚ	43
54. GUI TELEMETRIA – PANTALLA MODO DEMO.....	43
55. GUI TELEMETRIA – PANTALLA DE TELEMETRIA	44
56. COMBOBOX DE PUERTOS SERIALES DISPONIBLES	44
57. GUI TELEMETRIA – VENTANA DE DEBUG	45
58. GUI TELEMETRÍA – TRATAMIENTO DE LOS DATOS ENTRANTES	46
59. ESQUEMA DE FASES DE UN PROCESO TECNOLÓGICO	48
60. RADIO DE COMUNICACIÓN ASEGUADA DE ESTE SISTEMA DE TELEMETRÍA.....	49
61. PCB ECU TRANSMISORA V1.0	50
62. PCB ECU RECEPTORA V1.0	50
63. ECU TRANSMISORA INTEGRADA EN LA MOTOCICLETA KAWASAKI Z800	I
64. ENSAMBLAJE DEL DISEÑO MECÁNICO DE LA CARCASA PARA LA UNIDAD DE CONTROL RECEPTORA	II
65. PRUEBA REALIZADA POR SEMTECH.....	V
66. FUNCIÓN USADA PARA CREAR VALORES DISTINTOS DE FORMA INCREMENTAL EN CADA VARIABLE.	VI
67. DISTANCIA MÁXIMA DE FUNCIONAMIENTO DEL SISTEMA DE TELEMETRÍA – PRUEBA 1: 53 METROS	VII
68. DISTANCIA MÁXIMA DE FUNCIONAMIENTO DEL SISTEMA DE TELEMETRÍA – PRUEBA 2: 201 METROS.....	VIII
69. DISTANCIA MÁXIMA DE FUNCIONAMIENTO DEL SISTEMA DE TELEMETRÍA – PRUEBA 3: 499 METROS.....	VIII
70. DISTANCIA MÁXIMA DE FUNCIONAMIENTO DEL SISTEMA DE TELEMETRÍA – PRUEBA 4: 1173 METROS.....	IX
71. PRUEBA DE LATENCIA ENTRE MÓDULOS – DATOS ENVIADOS POR ECU TRANSMISORA.....	X

72. PRUEBA DE LATENCIA ENTRE MÓDULOS – DATOS RECIBIDOS POR ECU RECEPTORA	XI
73. CÓDIGO FUENTE DE LA FUNCIÓN UTILIZADA PARA VERIFICAR EL COMPORTAMIENTO DE LAS VARIABLES SEGÚN LOS VALORES DE LOS DATOS CODIFICADOS.....	XIII
74. LECTURA MONITOR SERIAL DURANTE LA PRUEBA DE DESBORDAMIENTO DE VARIABLES.	XIV
75. ESQUEMA BÁSICO DE UN CONVERTIDOR REDUCTOR BUCK.....	XV
76. ESQUEMA ELÉCTRICO ECU TRANSMISORA – CONVERTIDOR DE POTENCIA	XV
77. LAYOUT EJEMPLO PLACA DE DESARROLLO LM267X.	XVI
78. LAYOUT ACTUAL DEL CONVERSOR DE POTENCIA BUCK INTEGRADO EN LA ECU TRANSMISORA.	XVI
79. CONVERTIDOR BUCK - TENSIÓN DE SALIDA DURANTE EL ENCENDIDO (SOFT START).....	XVII
80. CONVERTIDOR BUCK - TENSIÓN DE SALIDA DURANTE EL APAGADO.	XVII
81. CONVERTIDOR BUCK – TENSIÓN DE SALIDA (CANAL AZUL) Y TENSIÓN DE ENTRADA DE LA BATERÍA (CANAL AMARILLO).....	XVIII
82. CONVERTIDOR BUCK – TENSIÓN PRE INDUCTOR (CANAL AMARILLO) Y TENSIÓN POST INDUCTOR (CANAL AZUL)..	XVIII
83. MEDICIÓN DE CONSUMO DE LA UNIDAD DE CONTROL TRANSMISORA EN STANDBY	XXI
84. MEDICIÓN DE CONSUMO DE LA UNIDAD DE CONTROL TRANSMISORA EN COMUNICACIÓN	XXI
85. MEDICIÓN DEL CONSUMO DE TODO EL SISTEMA ELÉCTRICO DE LA MOTOCICLETA SIN ECU TRANSMISORA.....	XXII
86 DISTRIBUCIÓN DE PINES DEL MICROCONTROLADOR ESP-32	XXV
87. DISEÑO DE PLACA DE DESARROLLO ESP-32 ESPRESSIF – AUTOMATIZACIÓN DEL PROCESO PARA REGRABAR EL MICROCONTROLADOR	XXVI
88. DISTRIBUCIÓN INCORRECTA DE LAS SEÑALES USB PRESENTES EN LOS DISEÑOS ACTUALES	XXVII
89. CORRECCIÓN DE LAS PISTAS USB MEDIANTE WIRE WRAP.	XXVII
90. CIRCUITO PARA LEER LA TENSIÓN DE LA BATERÍA.....	XXIX
91. RECORTE DE LA HOJA DE DATOS DEL TRANSISTOR BC33716TA DEL FABRICANTE FAIRCHILD.....	XXIX
92. FOTO PCB TRANSMISORA DONDE SE MUESTRA EN EL CENTRO EL FOOTPRINT DEL CONECTOR DE KAWASAKI Y EN LOS LATERALES EL FOOTPRINT DE LOS BLOQUES TERMINALES.	XXXI
93. COMPORTAMIENTO DEL GPIO 00 (CANAL AZUL) Y DEL PIN EN (CANAL AMARILLO) DURANTE EL ENCENDIDO DE LA ECU TRANSMISORA.....	XXXII
94. COMPORTAMIENTO DEL GPIO 00 (CANAL AZUL) Y DEL PIN EN (CANAL AMARILLO) DURANTE EL ENCENDIDO DE LA PLACA DE DESARROLLO DEL ESP-32.	XXXIII

95. CARPETA DE CÓDIGOS FUENTE.....	XXXV
96. ESQUEMA ECU TRANSMISORA – MICROCONTROLADOR Y PERIFÉRICOS.....	XXXVI
97. ESQUEMA ECU TRANSMISORA – CIRCUITOS DE ACONDICIONAMIENTO	XXXVII
98. ESQUEMA ECU TRANSMISORA – CONECTOR MOTO.....	XXXVIII
99. ESQUEMA ECU TRANSMISORA – POTENCIA	XXXIX
100. ESQUEMA ECU RECEPTORA.....	XL

1. Introducción

La telemetría aplicada a la automoción se refiere al proceso de medición y transmisión de datos en tiempo real desde un vehículo en movimiento a un centro de control o estación de monitoreo. El vehículo objeto de monitorización está equipado con una variedad de sensores y dispositivos electrónicos que capturan información en tiempo real, en su mayoría relacionada con el rendimiento del vehículo. Los datos recopilados incluyen velocidad, aceleración, temperatura del líquido refrigerante del motor, revoluciones del motor, posición del acelerador, entre otros.

Esta información es transmitida de manera inalámbrica mediante tecnologías de comunicación como la red celular o el GPS, lo que permite a los fabricantes, los ingenieros y los conductores recopilar datos precisos y actualizados sobre el rendimiento y el estado del vehículo. La telemetría en la automoción puede utilizarse para el diagnóstico y mantenimiento del vehículo, la mejora del rendimiento, el monitoreo de la seguridad y el análisis del comportamiento del conductor en tiempo real.

Este proyecto tiene como objetivo explorar el mundo de la telemetría para la automoción y evaluar la viabilidad de la tecnología inalámbrica por radiofrecuencia LoRa (Long Range) como una solución efectiva en este campo. En particular, se describe cómo se integra esta tecnología emergente, comúnmente utilizada en soluciones de IoT, y se evalúa su potencial para mejorar la seguridad y el rendimiento de los vehículos. Para ello, el presente trabajo se estructura en diferentes partes.

En primer lugar, se explica cómo funciona la tecnología LoRa y se estudia cómo integrarla para fines de telemetría en automoción. Así como el análisis del vehículo usado para adquirir sus datos y la elección de ellos.

En segundo lugar, se expone como se realiza el diseño electrónico y de software del sistema de telemetría (tanto el módulo transmisor como el módulo receptor).

En tercer lugar, se realizan diversas pruebas para comprobar la fiabilidad de los sistemas electrónicos diseñados, así como la transmisión de la comunicación por radiofrecuencia, la distancia máxima de la comunicación y su máxima latencia.

Y, por último, una conclusión de los resultados obtenidos del presente trabajo.

1.1 *Objetivo del proyecto*

El objetivo de este proyecto es explorar y evaluar la tecnología inalámbrica por radiofrecuencia LoRa como una solución efectiva en el campo de la telemetría para la automoción.

Para ello, se ha desarrollado un sistema completo que incluye el diseño electrónico con componentes discretos, la integración de módulos, el diseño completo del “*layout*” de PCB, la programación del firmware de los microcontroladores, la adquisición de datos y su conversión a formato digital, así como la transmisión, recepción y monitoreo de datos en tiempo real en una pantalla. Por último, se ha diseñado en lenguaje Python una aplicación de escritorio que hace de interfaz gráfica (Graphical User Interface o GUI) para poder mostrar los datos por pantalla en tiempo real.

En resumen, este proyecto tiene como objetivo demostrar que la tecnología LoRa puede ser una solución efectiva para la telemetría en la automoción, lo que puede mejorar la seguridad y el rendimiento de los vehículos y sentar las bases para su posible uso en futuros vehículos autónomos en redes V2X.

2. Tecnología Lo-Ra para la telemetría en la automoción

2.1. Funcionamiento de la tecnología Lo-Ra.

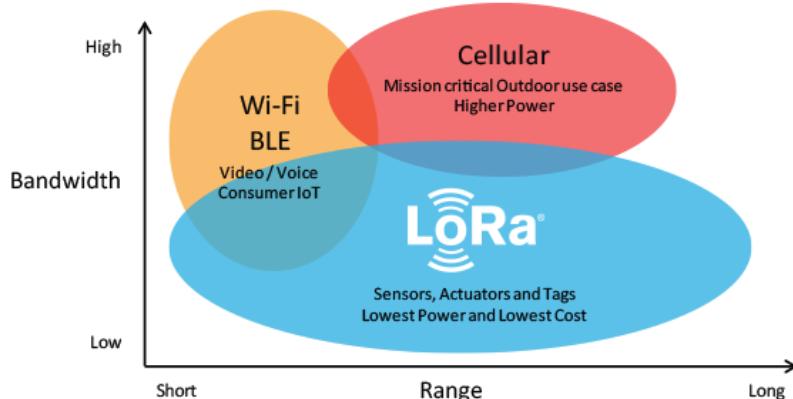
LoRa (acrónimo de Long Range) es una tecnología de comunicación inalámbrica de bajo consumo energético, baja transferencia de datos y con largo alcance (entre 10 y 20 km en zonas abiertas), que se basa en la modulación de espectro ensanchado. La técnica de modulación de espectro ensanchado usada en LoRa permite aumentar el alcance y la capacidad de penetración de la señal a través de obstáculos como edificios y árboles manteniendo un bajo consumo.

En contraste con la comunicación por radiofrecuencia convencional, que requiere una antena especializada para cada frecuencia utilizada, la tecnología LoRa utiliza una sola antena que cubre todas las frecuencias de su banda. Esto simplifica el diseño y reduce los costos de implementación de la tecnología LoRa en comparación con la comunicación por radiofrecuencia convencional.

Esta tecnología permite la comunicación bidireccional, por tanto, puede enviar como recibir datos.

LoRa utiliza técnicas avanzadas de procesamiento de señales, como el correlacionado de señales, para recuperar la información transmitida en un rango amplio de señales de frecuencia. Esto permite una alta sensibilidad y selectividad de la recepción de señales, lo que resulta en una transmisión de datos confiable incluso en entornos urbanos densamente poblados.

Además, LoRa es universal, opera en las bandas de frecuencia no licenciadas de 433 MHz, 868 MHz en Europa y 915 MHz en América del Norte, lo que permite una implementación fácil y económica en comparación con las tecnologías inalámbricas licenciadas.



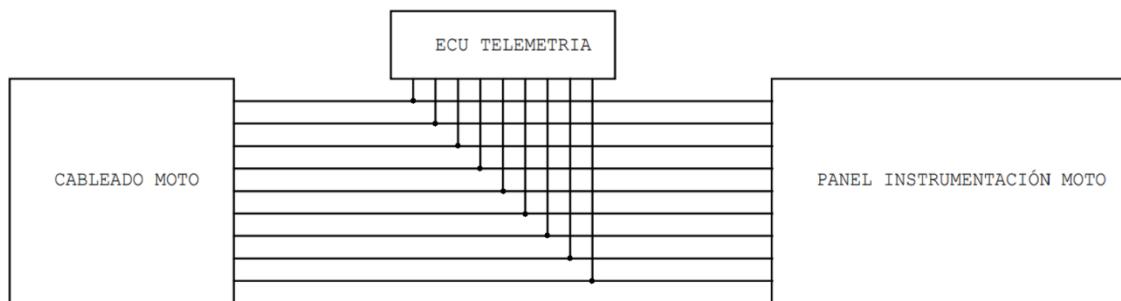
1. LoRa trabaja donde el Wi-Fi y la tecnología celular no. Fuente: Semtech Corporation [1].

2.2. Integración de Lo-Ra para la telemetría en automoción.

Debido a que este proyecto se realiza en colaboración con la *Universidad de Vic*, me han permitido usar un vehículo de sus instalaciones para poder llevar a cabo este trabajo de final de grado.

El sistema de telemetría consta de dos módulos: el primer módulo va integrado en la carrocería de la moto, y el segundo módulo es externo y debe ser conectado a un ordenador mediante USB para poder mostrar los datos por pantalla.

El primer módulo irá conectado al cableado original de la moto y hará de “bypass” hacia la siguiente unidad de control sin afectar a las señales, tal y como se muestra en la siguiente ilustración.

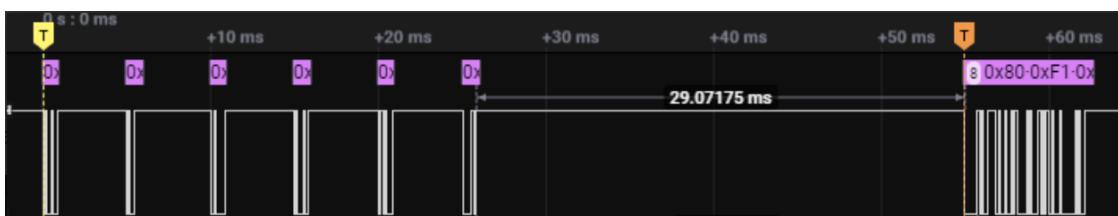


2. Representación del módulo electrónico integrado en la moto. Imagen de elaboración propia.

2.3. Análisis del vehículo y elección de datos a transmitir.

Este proyecto se ha desarrollado sobre la motocicleta Kawasaki Z800 del laboratorio de Can Muntanyola. Se trata de un vehículo con motor de 806 cc de 4 tiempos con 4 cilindros en línea e inyección electrónica. Pese a que el vehículo está equipado con sistema ABS, unidades de control electrónicas y diagnosis KDS, no cuenta con red de comunicación CAN BUS.

La diagnosis KDS, o “*Kawasaki Diagnostic System*”, es un sistema de diagnóstico de fallas electrónicas utilizado por Kawasaki en sus motocicletas. La herramienta de KDS se conecta a la motocicleta a través de un cable especial y utiliza software de diagnóstico específico de Kawasaki para analizar los datos de la motocicleta. La diagnosis utiliza el bus de comunicación K-Line, el cual tiene una velocidad de transmisión de datos mucho más baja en comparación con otros buses como el CAN BUS, alcanzando únicamente 10.4 kbps. Debido a esta limitación, no es posible obtener los datos en tiempo real utilizando el bus K-Line. Por lo tanto, resulta necesario utilizar otros buses de comunicación con mayores tasas de transmisión de datos, como el CAN BUS, el cual tiene una velocidad de transmisión de datos que puede variar entre 125 kbps y 1 Mbps.



3. Señal de comunicación K-Line. Imagen de elaboración propia.

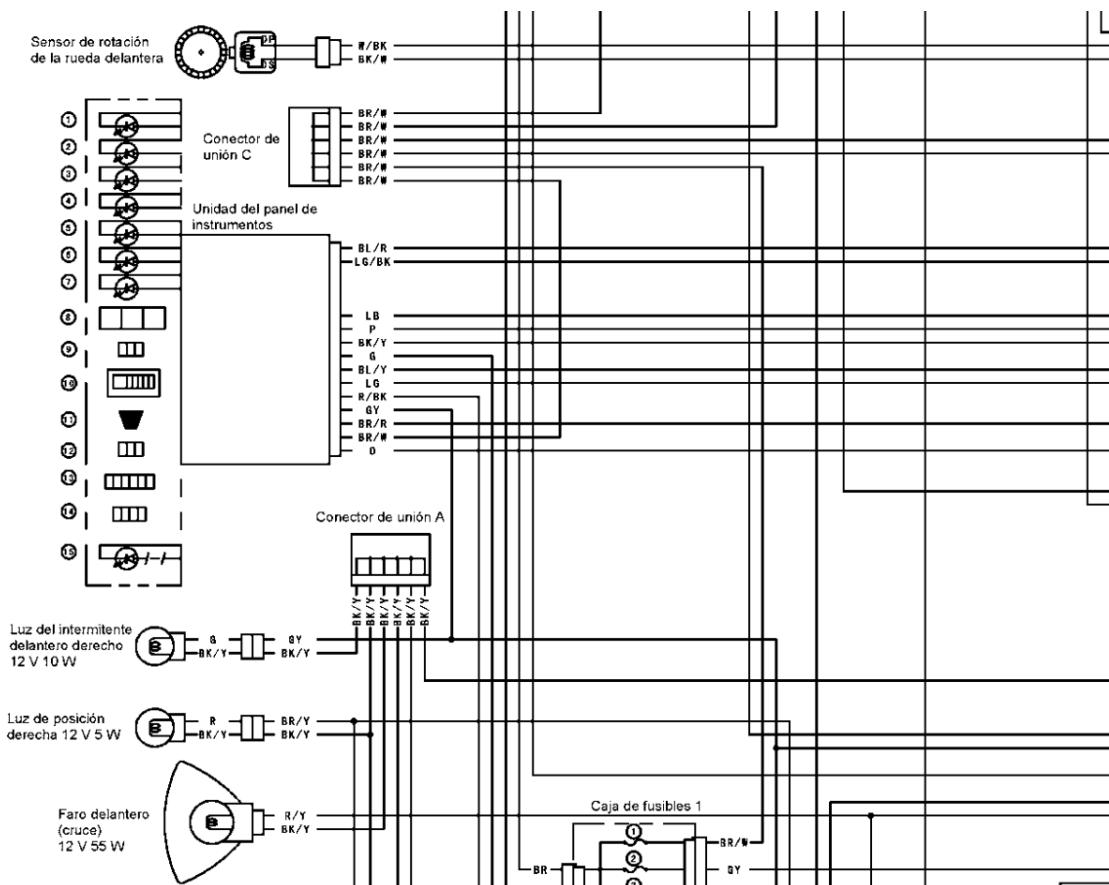
Debido a esta limitación de velocidad en K-Line y a la ausencia de un bus de comunicación rápido, se ha decidido realizar el tratamiento de las señales en crudo tal y como les llegan a las unidades de control desde los sensores. Obteniendo así, el máximo posible de latencia en las señales.

Dado que la Kawasaki Z800 es una motocicleta del tipo “*naked*” o sin carenado, para evitar la necesidad de desmontar una gran cantidad de componentes, se optará por recopilar los datos que se envían a la unidad de control del cuadro de instrumentos de la motocicleta.

Con la ayuda del manual de taller de la motocicleta y su esquema eléctrico, fue posible ver las señales que le llegaban al cuadro de instrumentos. Entre todas las señales consideré interesantes para ser transmitidas las siguientes señales:

- Tensión de batería.
- Nivel de combustible.
- Señal de llave.
- Señal de marcha neutra.
- Señal de presión de aceite.
- Señal de velocidad de rueda trasera.
- Señal de revoluciones del cigüeñal.

Después de esto, se procedió a utilizar un osciloscopio para identificar y analizar el tipo de señal presente en cada una de las líneas, con el fin de poder diseñar un circuito que permita el acondicionamiento adecuado de dichas señales.



4. Recorte del esquema eléctrico de la motocicleta. Fuente: Manual de taller Kawasaki Z800 [2].

3. Diseño electrónico y del software para el sistema de telemetría

3.1. Diseño del hardware

Utilizando el osciloscopio, se procedió a analizar las señales seleccionadas para su transmisión. Durante este proceso, se observó que existen dos señales analógicas que requieren ser procesadas por el Convertidor Analógico-Digital (ADC) del microcontrolador, así como cinco señales digitales. Dentro de estas señales digitales, se identificó una señal de pulso cuadrado que oscila entre 0 y 5V, así como otra señal de pulso cuadrado que varía entre 0 y 12V.

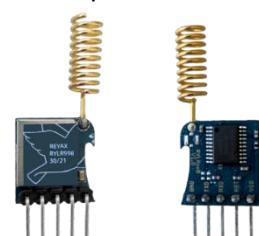
Se han elegido los microcontroladores de la familia ESP-32 para llevar a cabo este proyecto, en especial el modelo ESP WROOM-32D. Se trata de un módulo de bajo costo fabricado por la empresa china Espressif. Este modelo cuenta con un procesador de doble núcleo que funciona a una frecuencia máxima de 240 MHz y 4 MB de memoria flash. Esto significa que es posible ejecutar múltiples tareas al mismo tiempo, y asignar una tarea específica a cada núcleo. Por ejemplo, uno de los núcleos puede encargarse de la adquisición y tratamiento de los datos, mientras que el otro se dedica exclusivamente a la transmisión de datos.



5. Microcontrolador ESP WROOM-32D. Fuente: Espressif.

La ventaja de esto es que se pueden ejecutar procesos simultáneamente sin que uno afecte negativamente el desempeño del otro. Gracias al sistema operativo de tiempo real (RTOS) integrado [3], es posible asignar prioridades a las tareas y asegurar que las más críticas se ejecuten primero. Además, el sistema de interrupciones permite que se ejecuten tareas asíncronas en respuesta a eventos externos.

Para la comunicación por radiofrecuencia se han elegido los módulos de LoRa RYLR998. Este módulo en especial utiliza el chip transceptor de la serie SX127X de Semtech que es altamente eficiente y con una sensibilidad de recepción extremadamente alta. Esto permite una mejor recepción de señales incluso en ambientes de ruido y alta interferencia. El módulo RYLR998 cuenta con comunicación serial UART para comunicarse con un microcontrolador, además es altamente configurable permitiendo personalizar los parámetros de transmisión según las necesidades de la aplicación.



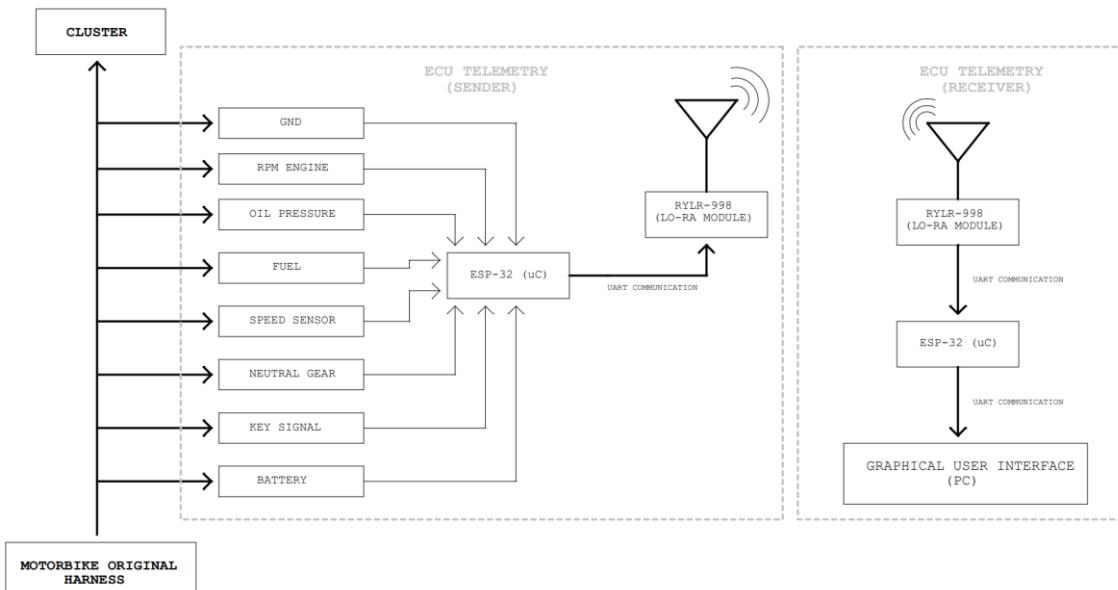
6. Módulo LoRa RYLR998. Fuente: Reyax [4].

3.1.1. Diagrama de bloques del sistema

El sistema de telemetría se compone de dos unidades de control. A la primera se referirá en adelante como la unidad de control electrónica transmisora, y a la segunda como unidad de control electrónica receptora.

La ECU (Electronic Control Unit) transmisora se intercala en el cableado original de la moto hacia el cuadro de instrumentos y, mediante pistas en la PCB, se bifurca la señal al microcontrolador (pasando por los circuitos de acondicionamiento de señal) y al cuadro de instrumentos original de la motocicleta. El microcontrolador ESP-32 se comunica con el módulo de LoRa RYLR998 mediante comunicación por puerto UART. Por último, el módulo RYLR998 envía mediante radiofrecuencia los datos ya adquiridos, acondicionados y codificados.

La ECU (Electronic Control Unit) receptora capta por radiofrecuencia los datos mediante el RYLR998, estos datos los envía al microcontrolador ESP-32 mediante UART, y posteriormente se envían mediante otro puerto UART al ordenador. Para alimentar la ECU y enviar los datos al puerto serial USB, es necesario conectar esta unidad de control al PC a través del conector USB. Por último, en el ordenador se debe iniciar la interfaz gráfica programada en Python, la cual recibe los datos por el puerto serial y los muestra por pantalla en tiempo real con la máxima latencia posible.



7. Diagrama de bloques del sistema de telemetría. Imagen de elaboración propia.

3.1.2. Circuitos tipo

Para garantizar el correcto funcionamiento de los circuitos de acondicionamiento de señal, se utilizan las siguientes tres hipótesis para ajustar los valores de los componentes electrónicos:

- Hipótesis 1: Tensión de entrada al circuito de +14 V. Esta hipótesis pretende simular un vehículo con una tensión nominal normal.
- Hipótesis 2: Tensión de entrada al circuito de +10 V. Esta hipótesis pretende simular un vehículo con una batería con celda defectuosa. En este caso, produce un bajo voltaje en su sistema eléctrico.
- Hipótesis 3: Tensión de entrada al circuito de +16 V. Esta hipótesis pretende simular un vehículo con un fallo en su sistema eléctrico. En este caso un malfuncionamiento del diodo regulador de tensión del alternador que ocasionaría una sobre tensión en el circuito.

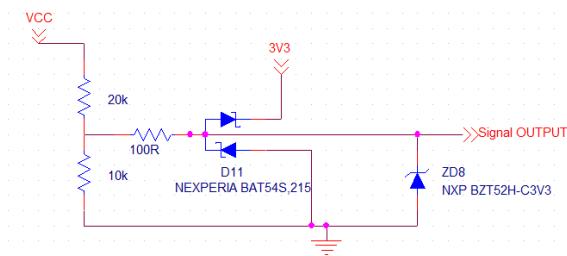
3.1.2.1. Circuito para Acondicionar Señales Digitales

El propósito de este tipo de circuito es reducir la señal de tensión de +12 V a un nivel que esté por debajo del límite máximo de tensión admitido por el microcontrolador en la entrada de sus GPIO. El resultado esperado es evitar dañar los componentes electrónicos del microcontrolador y garantizar una lectura precisa y confiable de la señal de entrada.

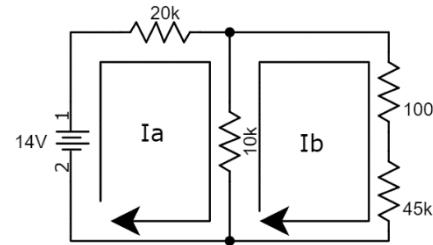
Para cumplir con este objetivo, se utilizará un divisor de tensión que adaptará la tensión de la señal de entrada al nivel adecuado, y una resistencia que limitará la corriente de entrada al GPIO del microcontrolador. Luego, se aplicará una terminación de red dual de Schottky, que actúa como un filtro para reducir el ruido de la señal y proteger el circuito [5]. Finalmente, se incluirá un diodo Zener con una tensión de ruptura de 3'3 V para proteger el microcontrolador de cualquier sobretensión que pueda dañarlo. De esta manera, se asegurará que el circuito funcione de manera efectiva y se eviten daños en los componentes electrónicos.

La terminación de Red Schottky entre otros beneficios actúa como protección contra sobretensiones, permitiendo que la señal fluya a través del circuito cuando está por debajo de los 3'3 V. Sin embargo, cuando ocurre una sobretensión o pico de voltaje, el diodo Schottky se activa y desvía el exceso de energía hacia la tierra protegiendo el GPIO [5].

Para ajustar los valores de las resistencias planteamos las tres hipótesis antes mencionadas con nuestro circuito suponiendo los valores mostrados en la figura 9. Tal y como se aprecia en la imagen 10, se simplifica el circuito para poder aplicar las leyes de voltaje de Kirchoff (KVL) y se añade la resistencia interna de pull-down del microcontrolador:



9. Circuito de acondicionamiento para señales digitales con valores de referencia. Imagen de elaboración propia.



8. Circuito de acondicionamiento para señales digitales simplificado con las mallas respectivas para el cálculo de KVL. Imagen de elaboración propia.

Cálculos analíticos del circuito:

Mediante el anterior circuito simplificado obtenemos las ecuaciones características del circuito y resolviendo obtenemos las corrientes de cada malla.

$$\text{Hipótesis 1: } V_{key} = 14 \text{ V}$$

$$\begin{cases} A: 14 = I_a \cdot 30k - I_b \cdot 10k \\ B: 0 = -I_a \cdot 10k + I_b \cdot 55k \end{cases} \rightarrow I_a = \frac{14 + I_b \cdot 10k}{30k} \rightarrow 0 = -\left(\frac{14 + I_b \cdot 10k}{30k}\right) \cdot 10k + I_b \cdot 55k \rightarrow$$

$$\begin{cases} I_b = 90'1481 \mu A \\ I_a = 496'7160 \mu A \end{cases}$$

$$\text{Hipótesis 2: } V_{key} = 10 \text{ V}$$

$$\begin{cases} A: 10 = I_a \cdot 30k - I_b \cdot 10k \\ B: 0 = -I_a \cdot 10k + I_b \cdot 55k \end{cases} \rightarrow I_a = \frac{10 + I_b \cdot 10k}{30k} \rightarrow 0 = -\left(\frac{10 + I_b \cdot 10k}{30k}\right) \cdot 10k + I_b \cdot 55k \rightarrow$$

$$\begin{cases} I_b = 64'3915 \mu A \\ I_a = 354'7971 \mu A \end{cases}$$

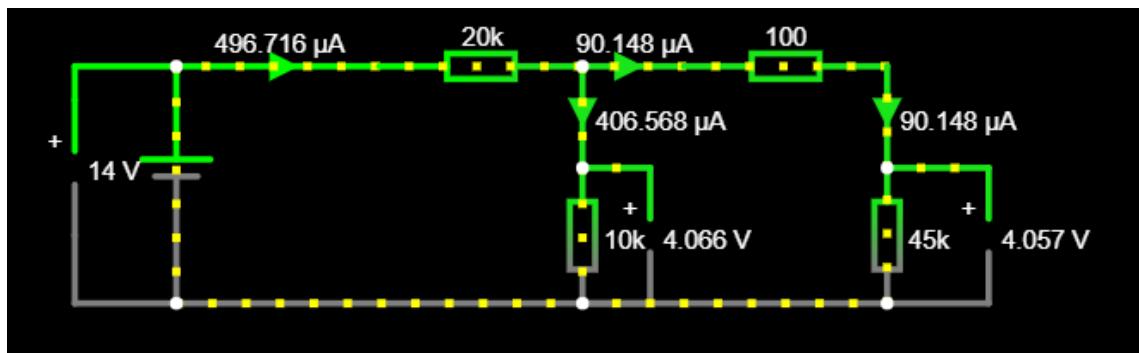
$$\text{Hipótesis 3: } V_{key} = 16 \text{ V}$$

$$\begin{cases} A: 16 = I_a \cdot 30k - I_b \cdot 10k \\ B: 0 = -I_a \cdot 10k + I_b \cdot 55k \end{cases} \rightarrow I_a = \frac{16 + I_b \cdot 10k}{30k} \rightarrow 0 = -\left(\frac{16 + I_b \cdot 10k}{30k}\right) \cdot 10k + I_b \cdot 55k \rightarrow$$

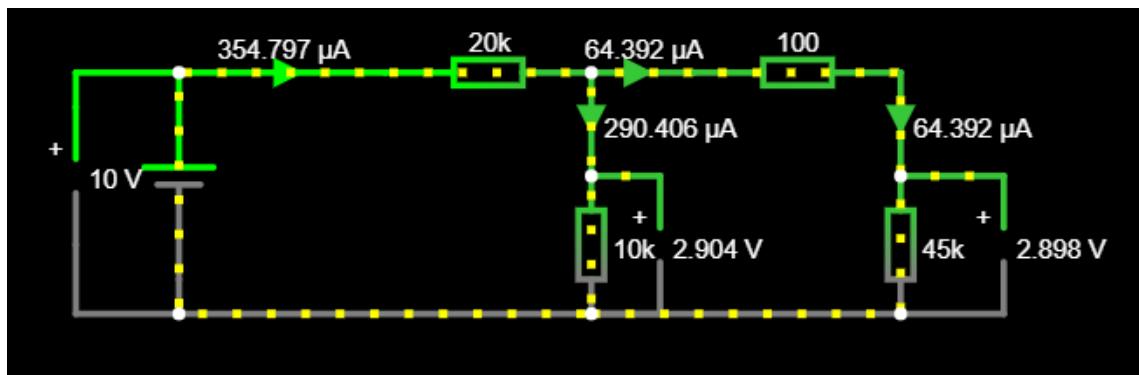
$$\begin{cases} I_b = 103'0264 \mu A \\ I_a = 567'6754 \mu A \end{cases}$$

Con los cálculos anteriormente mostrados se confirma que el circuito mostrará la señal de forma adecuada frente a sobre tensiones, bajas tensiones y cortocircuitos.

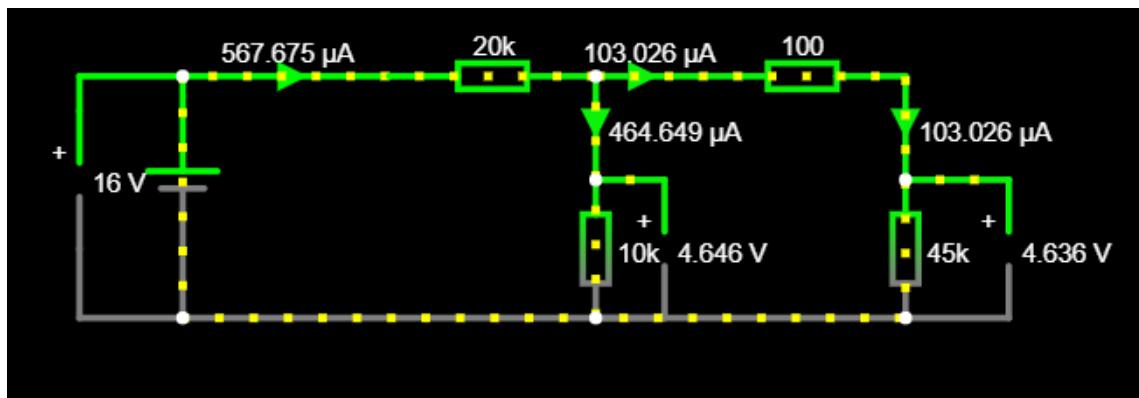
Circuito recreado en simulador:



10. Simulación sobre circuito digital con condiciones de la hipótesis 1 (VBAT=14 V). Imagen de elaboración propia.

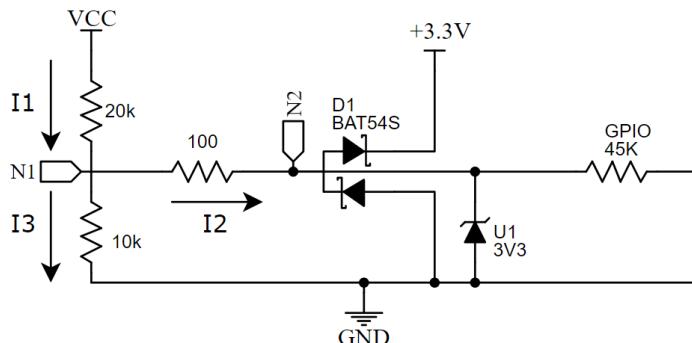


11. Simulación sobre circuito digital con condiciones de la hipótesis 2 (VBAT=10 V). Imagen de elaboración propia.



12. Simulación sobre circuito digital con condiciones de la hipótesis 3 (VBAT=16 V). Imagen de elaboración propia.

Mediciones sobre maqueta montada en Protoboard:



13. Representación del montaje del circuito digital sobre protoboard. Imagen de elaboración propia.

Medición hipótesis 1: $V_{cc} = 14 \text{ V}$

Con Zener: $N_1 = 2'319 \text{ V}$, $N_2 = 2'283 \text{ V}$

Sin zener: $N_1 = 4'670 \text{ V}$, $N_2 = 4'670 \text{ V}$

$I_1 = 500 \mu\text{A}$, $I_2 = 90 \mu\text{A}$, $I_3 = 409 \mu\text{A}$

Medición hipótesis 2: $V_{cc} = 10 \text{ V}$

Con Zener: $N_1 = 2'113 \text{ V}$, $N_2 = 2'094 \text{ V}$

Sin Zener: $N_1 = 3'340 \text{ V}$, $N_2 = 3'340 \text{ V}$

Medición hipótesis 3: $V_{cc} = 16 \text{ V}$

Con Zener: $N_1 = 2'396 \text{ V}$, $N_2 = 2'352 \text{ V}$

Sin Zener: $N_1 = 5'340 \text{ V}$, $N_2 = 5'340 \text{ V}$

Circuito con Zener y $V_{cc} = 32 \text{ V}$, al GPIO solo llegan $2'67 \text{ V}$.

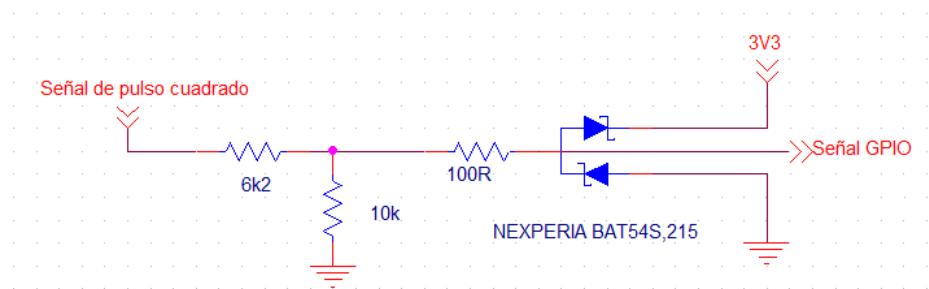
Conclusión:

Se aprecia como obtenemos el mismo resultado en los cálculos realizados mediante KVL y el simulador electrónico. También, los valores medidos en la protoboard no difieren tanto de los cálculos realizados de forma analítica. Mediante pruebas sobre el circuito montado en protoboard se ha observado que cuando el diodo Zener está montado, con una tensión de entrada al circuito de 32 V, tan sólo le llegan 2'67 V al GPIO.

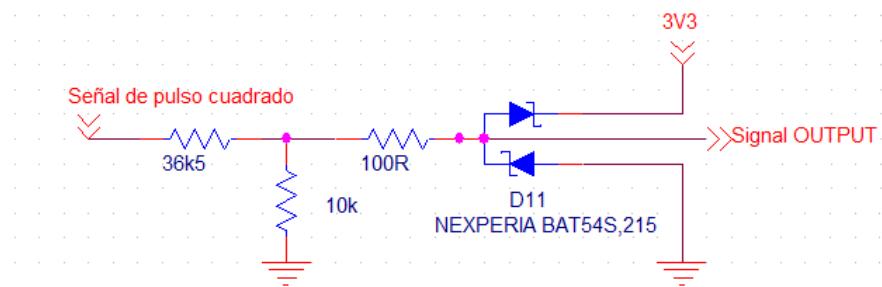
1.1.1.1. Circuito para Acondicionar Señales Digitales de Pulso Cuadrado

Este circuito tipo, se usará para dos señales distintas de pulso cuadrado, una correspondiente a la velocidad y otra a las revoluciones del motor. Cada señal tiene una particularidad en cuanto a su nivel lógico: en la señal de velocidad, un valor lógico de 1 equivale a 5 V, mientras que en la señal de RPMs del motor, un valor lógico de 1 equivale a 12 V. Es importante tener en cuenta estas diferencias para evitar dañar los componentes electrónicos del circuito y garantizar una correcta lectura de las señales de entrada.

La información de velocidad del vehículo y de revoluciones del motor se obtienen a partir de la frecuencia de los pulsos cuadrados.



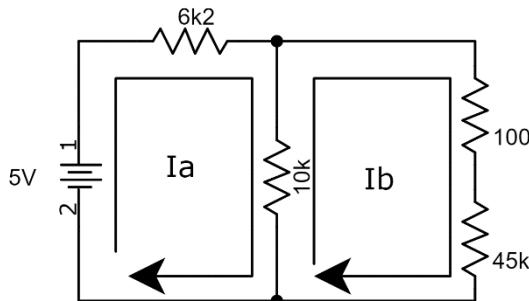
14 Circuito de acondicionamiento del velocímetro con valores de referencia.
Imagen de elaboración propia.



15 Circuito de acondicionamiento del tacómetro con valores de referencia.
Imagen de elaboración propia.

Cálculos analíticos del circuito:

- Velocímetro:



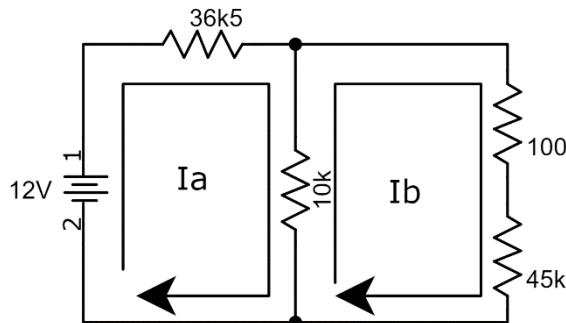
16. Circuito de acondicionamiento para la señal del velocímetro simplificado con las mallas respectivas para el cálculo de KVL. Imagen de elaboración propia.

$$\begin{cases} A: 5 V = I_a \cdot (6k2 + 10k) - (I_b \cdot 10k) \\ B: 0 = I_b \cdot (10k + 100 + 45k) - (I_a \cdot 10k) \end{cases} \rightarrow I_b = \frac{I_a \cdot 10k}{10k + 100 + 45k} \rightarrow$$

$$5 V = I_a \cdot (6k2 + 10k) - \left(\frac{I_a \cdot 10k}{10k + 100 + 45k} \right) \cdot 10k$$

$$\begin{cases} I_a = 347'5814 \mu A \\ I_b = 63'0818 \mu A \end{cases}$$

- Tacómetro:



17. Circuito de acondicionamiento para la señal del tacómetro simplificado con las mallas respectivas para el cálculo de KVL. Imagen de elaboración propia.

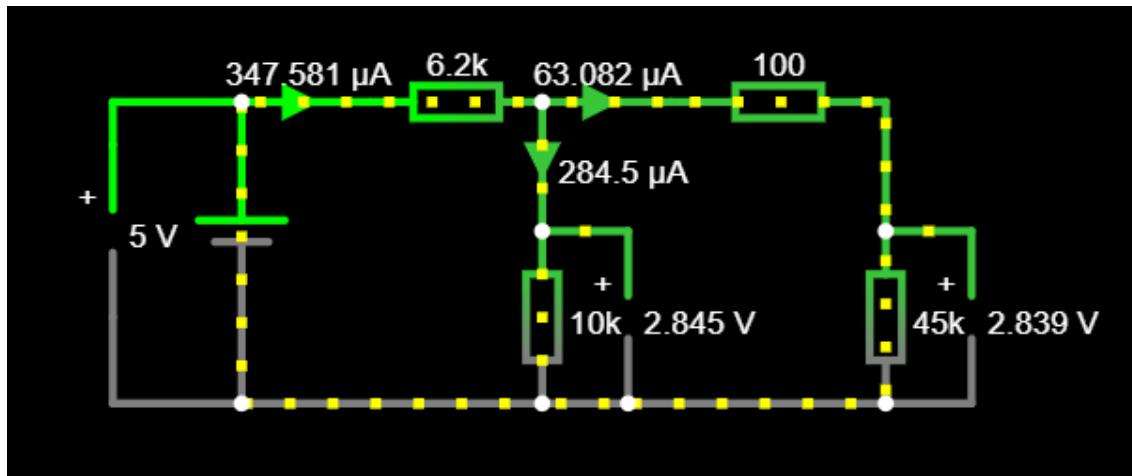
$$\begin{cases} A: 12 V = I_a \cdot (36k5 + 10k) - (I_b \cdot 10k) \\ B: 0 = I_b \cdot (10k + 45k + 100) - I_a \cdot 10k \end{cases} \rightarrow I_b = \frac{I_a \cdot 10k}{10k + 100 + 45k} \rightarrow$$

$$12 V = I_a \cdot (36k5 + 100) - \left(\frac{I_a \cdot 10k}{10k + 100 + 45k} \right) \cdot 10k \rightarrow$$

$$\begin{cases} I_a = 268'5457 \mu A \\ I_b = 48'7375 \mu A \end{cases}$$

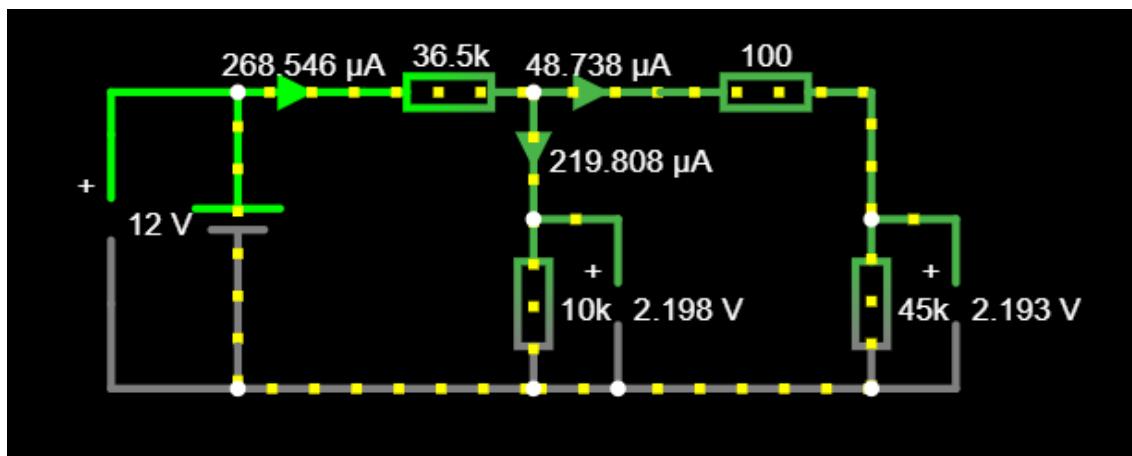
Círcuito recreado en simulador:

- Velocímetro:



18. Simulación sobre el circuito de acondicionamiento del velocímetro. Imagen de elaboración propia.

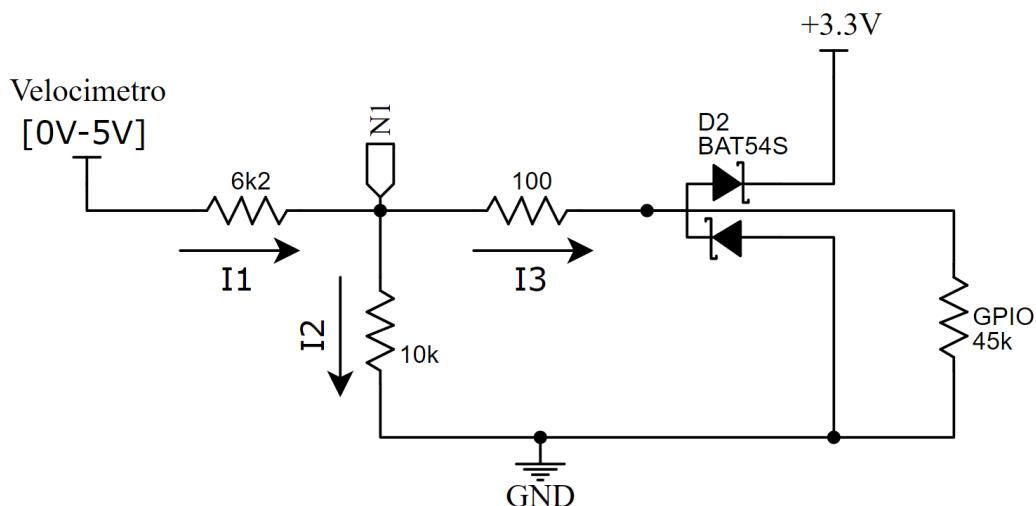
- Tacómetro:



19. Simulación sobre el circuito de acondicionamiento del tacómetro. Imagen de elaboración propia.

Mediciones sobre maqueta montada en Protoboard:

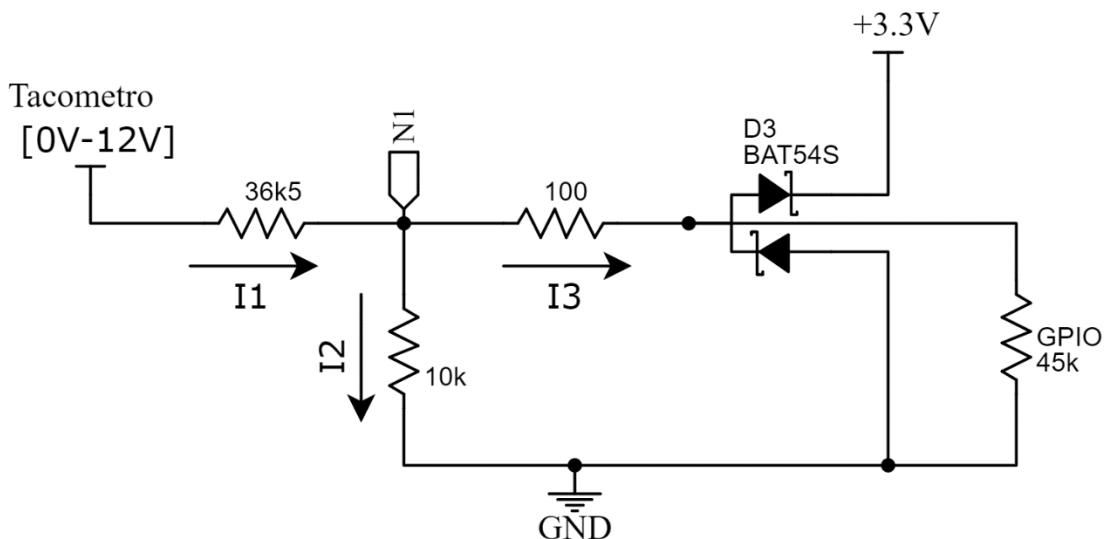
- Velocímetro:



20. Representación del montaje del circuito de acondicionamiento del velocímetro sobre protoboard.
Imagen de elaboración propia.

$$N_1 = 2'92 \text{ V}, \quad V_{GPIO} = 2'92 \text{ V}, \quad I_1 = 0'34 \text{ mA}, \quad I_2 = 0'28 \text{ mA}, \quad I_3 = 0'06 \text{ mA}$$

- Tacómetro:



21. Representación del montaje del circuito de acondicionamiento del tacómetro sobre protoboard. Imagen de elaboración propia.

$$N_1 = 2'24 \text{ V}, \quad V_{GPIO} = 2'24 \text{ V}, \quad I_1 = 0'27 \text{ mA}, \quad I_2 = 0'22 \text{ mA}, \quad I_3 = 0'04 \text{ mA}$$

Conclusión:

Los cálculos analíticos, los valores obtenidos en el simulador y los resultados medidos en la maqueta en protoboard coinciden, lo que demuestra una alta precisión en el diseño del circuito. Es crucial que se obtenga una reproducción precisa de la señal de entrada a la señal de salida, por lo que no se puede añadir un diodo Zener al circuito, ya que esto generaría un retraso y distorsión en la señal de salida.

Además, se llevó a cabo un experimento en el que se agregó una señal cuadrada de 50 kHz de frecuencia en la entrada del circuito acondicionador y se observó el fenómeno del Slew Rate en la señal de salida. A pesar de esto, la señal sigue siendo válida para que el microcontrolador cuente los pulsos de las señales.

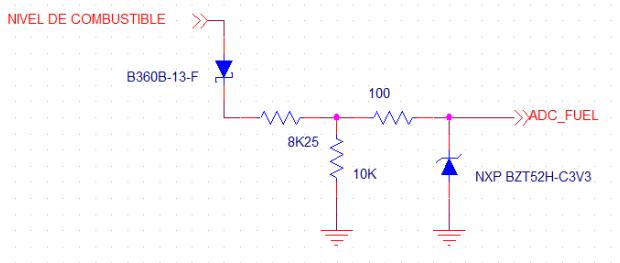


22 Captura del osciloscopio donde se observa la distorsión de la señal en la salida respecto a la entrada del circuito debido al efecto del Slew Rate.
Imagen de elaboración propia.

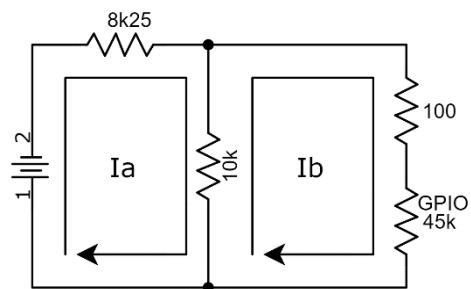
1.1.1.1. Circuito para Acondicionar Señal Analógica de la Sonda del Combustible

Para acondicionar esta señal debemos tener en cuenta que el cuadro de instrumentos de la moto ya aplica una tensión al circuito (oscila entre 0 a 6 V, siendo 6 V depósito de combustible lleno). Debido a que nosotros obtenemos el terminal de señal de la sonda de combustible, para evitar distorsionar la señal es de gran importancia añadir un diodo de baja caída de tensión como un Schottky con el fin de separar los circuitos, tal y como se muestra en la figura 23.

Se puede observar que, en la imagen 24, se ha simplificado el circuito para poder aplicar el método de cálculo de tensiones de Kirchoff al circuito.



23. Circuito de acondicionamiento de la señal de combustible con valores de referencia. Imagen de elaboración propia.



24. Circuito de acondicionamiento de la señal de combustible simplificado con las mallas respectivas para el cálculo de KVL. Imagen de elaboración propia.

Cálculos analíticos del circuito:

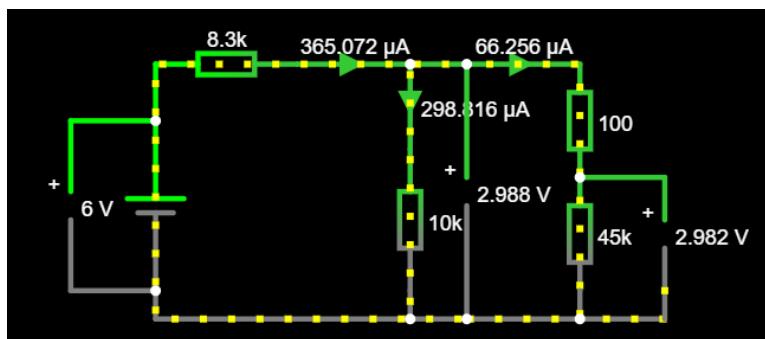
$$V_{fuel} = 6 \text{ V:}$$

$$\begin{cases} a: 6 \text{ V} = I_a \cdot (8k25 + 10k) - I_b \cdot 10k \\ b: 0 \text{ V} = I_b \cdot (45k + 10k + R100) - I_a \cdot 10k \end{cases}$$

$$I_b = \frac{I_a \cdot 10k}{45k + 10k + R100} \rightarrow 6 \text{ V} = I_a \cdot (8k25 + 10k) - \left(\frac{I_a \cdot 10k}{45k + 10k + R100} \right) \cdot 10k \rightarrow$$

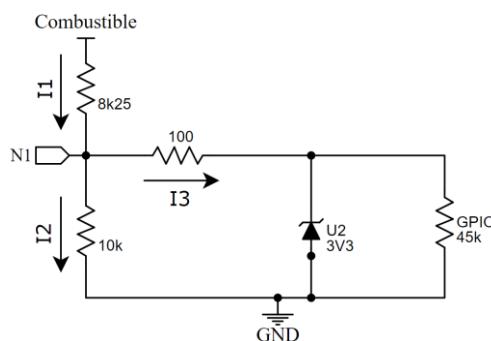
$$\begin{cases} I_a = 365'0719 \text{ uA} \\ I_b = 66'2562 \text{ uA} \end{cases}$$

Circuito recreado en simulador:



25. Simulación sobre el circuito de acondicionamiento de la señal de combustible con la tensión máxima. Imagen de elaboración propia.

Mediciones sobre maqueta montada en Protoboard:



26. Representación del montaje del circuito de acondicionamiento de la señal de combustible sobre protoboard. Imagen de elaboración propia.

Sin Zener:

$$V_{cc} = 6 V, \quad N_2 = 3'10 V, \quad V_{GPIO} = 3'10 V \\ I_1 = 0'36 mA, \quad I_2 = 0'30 mA, \quad I_3 = 0'06 mA$$

Para que lleguen 3'6 V al GPIO, la tensión de alimentación V_{cc} debe ser de 7 V.

Para que lleguen 3'3 V al GPIO, la tensión de alimentación V_{cc} debe ser de 6'50 V.

Con Zener:

$$V_{cc} = 6V, \quad N_2 = 2'38V, \quad V_{GPIO} = 2'32V$$

Para que lleguen 3'3 V al GPIO, la tensión de alimentación V_{cc} debe ser de 31 V.

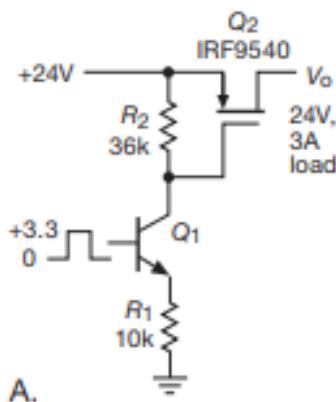
Para que lleguen 3'6 V al GPIO, la tensión de alimentación V_{cc} debe de ser mayor a 32 V, por lo que no se ha medido por seguridad.

Conclusión:

Para este circuito tipo se aprecia como coinciden las corrientes en los cálculos analíticos, las mediciones del prototipo sobre la protoboard y la simulación del circuito electrónico.

1.1.1.1. Circuito para Acondicionar Señal Analógica de la Tensión de Batería

Circuito basado en la propuesta de Paul Horowitz [6], para controlar cargas de potencia y protección de los interruptores de potencia (Imagen 27).

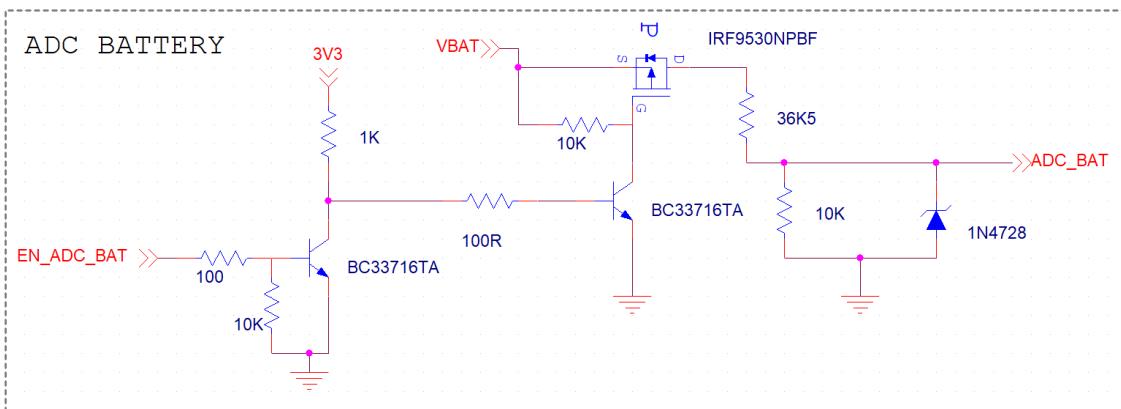


27. Circuito para controlar cargas de potencia y protección de los interruptores de potencia. Fuente: The art of electronics [6].

El propósito principal de este circuito es garantizar la protección contra la inversión de polaridad. Normalmente, esta protección se logra mediante la inclusión de un diodo en la entrada del circuito. No obstante, en este caso en particular, la medición de la tensión de la batería es crucial y la inclusión del diodo no es la mejor opción debido a la caída de tensión que se produce a través de este componente, la cual varía en función de la temperatura y la corriente que circula por el mismo.

Este circuito utiliza el MOSFET como un interruptor para permitir que la tensión de VBAT sea comunicada a un canal específico del conversor analógico-digital (ADC) de 12 bits del microcontrolador. Es importante destacar que el MOSFET solo se activará cuando se reciba la señal EN_ADC_BAT del microcontrolador.

Para el diseño de este circuito (figura 28) se usa un transistor MOSFET de canal P junto con un transistor BJT NPN para usar el MOSFET como un interruptor. También, se añade otro transistor NPN para invertir la lógica del circuito en reposo. De esta manera, sólo llegará la tensión desde el cableado de la moto hasta el pin ADC del microcontrolador cuando se active la señal EN_ADC_BAT mediante software.



28. Circuito de acondicionamiento para muestrear la tensión de batería con valores de referencia. Imagen de elaboración propia.

Se ha usado el modelo de BJT BC33716TA que, entre muchas otras, tiene las siguientes características (para más información ver apartado de Anexo 7.8 Datasheets):

- $V_{CES} = 50\text{ V}$
- $V_{CEO} = 45\text{V}$
- $I_C = 800\text{ mA}$
- $h_{fe} = [100, 250]$

Para el MOSFET de canal P se ha usado el modelo del fabricante Infineon IRF9530NPBF, el cual tiene las siguientes características (para más información ver apartado de Anexo 7.8 Datasheets):

- $V_{DS} = 100\text{ V}$
- $I_D = 14\text{ A}$
- $V_{GS} = [-20V, +20V]$
- $V_{GS(th)} = [-2V, -4V]$

Por tanto, según el datasheet del fabricante, si $I_{C(sat\ máx)} < 100\ mA \rightarrow h_{fe1} = [100, 250]$.

La corriente de base mínima necesaria para la saturación se puede calcular a partir de la corriente de colector máxima $I_{C(sat)}$ y la ganancia del transistor (h_{fe}), utilizando la siguiente fórmula:

$$I_{b(min)} = \frac{I_{C(sat)}}{(h_{fe})}$$

Por lo tanto, si se tiene una ganancia de 100, la corriente de base mínima necesaria sería mayor que si se tiene una ganancia de 250, lo que significa que, para asegurar la saturación con una ganancia de 100, se necesitaría proporcionar una corriente de base aún mayor que para una ganancia de 250.

Para asegurar la saturación del transistor, suponiendo el caso más crítico, de ahora en adelante la ganancia será $h_{fe} = 100$.

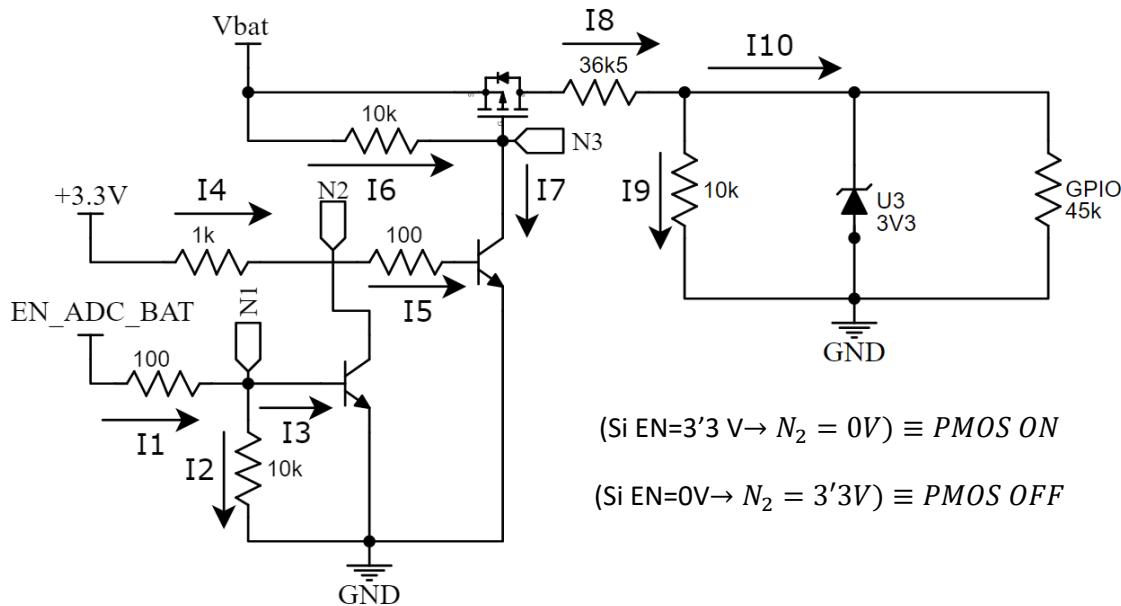
$$I_{C(sat)} = h_{fe} \cdot I_b \rightarrow I_{b(min)} = \gamma \cdot \left(\frac{I_{C(sat)}}{h_{fe}} \right) = 2 \cdot \left(\frac{3'32 \times 10^{-3}}{100} \right) = 66'4 \times 10^{-6}\ A$$

$$I_{b(min)} = 66'4 \times 10^{-6}\ A$$

Donde el factor γ es una constante de seguridad que se añade a la ecuación para asegurar la saturación del transistor.

Por tanto, para que el transistor sature correctamente para reducir pérdidas de energía y aumentar la velocidad de conmutación la corriente de base de los transistores “NPN BC33716TA” ha de ser igual o mayor a $66'4 \times 10^{-6}\ A$ o $66'4\ \mu\text{A}$.

Mediciones sobre maqueta montada en protoboard:



29. Representación del montaje del circuito de acondicionamiento para muestrear la tensión de batería. Imagen de elaboración propia.

Medición hipótesis 1: $V_{cc} = 14V$

Con Zener: $N_1 = 960\text{ mV}, N_2 = 160\text{ mV}, N_3 = 11'6\text{ V}, N_4 = 1'94\text{ V}$

Sin Zener: $N_1 = 960\text{ mV}, N_2 = 140\text{ mV}, N_3 = 11'6\text{ V}, N_4 = 2'20\text{ V}$

$I_1 = 24'28\text{ mA}, I_2 = 0'0836\text{ mA}, I_3 = 24'43\text{ mA}, I_4 = 3'32\text{ mA}, I_5 = 0\text{ mA}$

$I_6 = 0'259\text{ mA}, I_7 = 0\text{ mA}, I_8 = 0'2670\text{ mA}, I_9 = 0'2111\text{ mA}, I_{10} = 0'0463\text{ mA}$

Medición hipótesis 2: $V_{cc} = 10V$

Con Zener: $N_1 = 940\text{ mV}, N_2 = 120\text{ mV}, N_3 = 8'32\text{ V}, N_4 = 1'44\text{ V}$

Sin Zener: $N_1 = 960\text{ mV}, N_2 = 160\text{ mV}, N_3 = 8'32\text{ V}, N_4 = 1'60\text{ V}$

Medición hipótesis 3: $V_{cc} = 16V$

Con Zener: $N_1 = 940\text{ mV}, N_2 = 140\text{ mV}, N_3 = 13'2\text{ V}, N_4 = 2'00\text{ V}$

Sin Zener: $N_1 = 960\text{ mV}, N_2 = 160\text{ mV}, N_3 = 13'2\text{ V}, N_4 = 2'48\text{ V}$

Para que al GPIO le lleguen 3'6 V, la tensión V_{cc} debe ser de 23 V.

Para que al GPIO le lleguen 3'3 V, la tensión V_{cc} debe ser de 21 V.

Desde que el microcontrolador por software activa la señal “EN_ADC_Bat” hasta que el PMOS cierra el circuito completamente (N_4) pasan 40 mS.

Cuando señal EN_ADC_BAT tiene un valor lógico de 1, el transistor PMOS tiene una tensión $V_{GS} = -2'63$ V. Y cuando la señal lógica de EN_ADC_BAT es de 0, la tensión $V_{GS} = -14$ V. Según Datasheet, para que el transistor actúe como un interruptor, su tensión V_{GS} debe estar comprendida entre -2 V y -4 V. Por tanto, nuestro circuito está optimizado y funcionará correctamente.

Conclusión:

Se observa como la intensidad mínima de base en los transistores BJT NPN cuando actúan como interruptor son de 24'43 mA (I_3) y de 3'32 mA ($I_4 \cong I_5$ cuando EN_ADC_BAT equivale a 0V), por tanto:

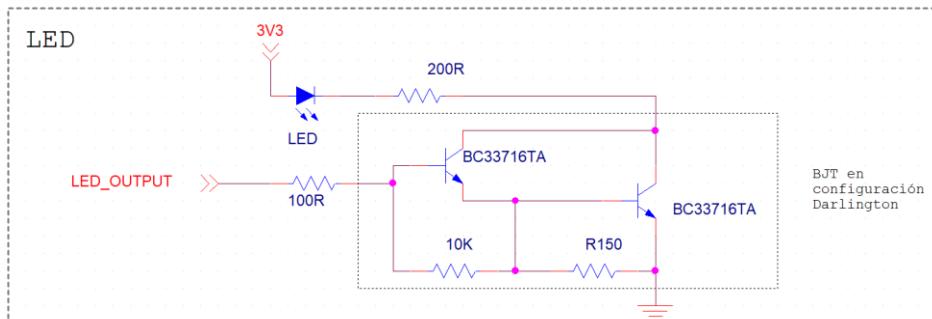
$$I_{b(\min \text{ medida})} = 3'32 \times 10^{-3} A \gg I_{b(\min)} = 66'4 \times 10^{-6} A$$

Asegurando así la saturación de los transistores NPN.

Debido a la complejidad en la modelización de los componentes activos, tales como BJT y PMOS, y la falta de opciones de personalización en el simulador, he decidido no simular este circuito.

1.1.1.1. Circuito de control para LED

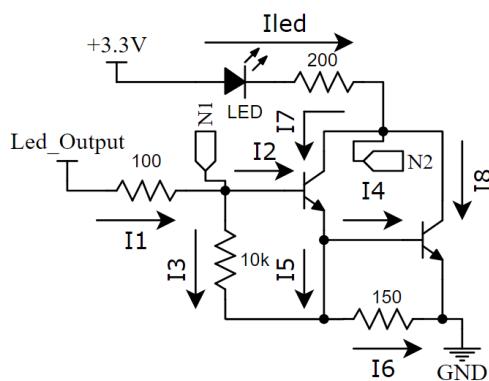
Se ha decidido implementar un circuito que utilice un transistor Darlington para el control de grandes cantidades de corriente con una muy pequeña intensidad de base [7]. Aunque solo se requiere controlar un LED que consume un máximo de 20 mA, se ha sobredimensionado el circuito para estar preparado en caso de querer implementar más LEDs o activadores adicionales, como un zumbador.



30. Circuito de activación de un actuador LED con valores de referencia. Imagen de elaboración propia.

Además, se ha optado por utilizar una configuración de transistores en configuración Darlington [8] con el objetivo de experimentar y aprender más sobre este tipo de configuración.

Mediciones sobre maqueta montada en Protoboard:



31. Representación del montaje del circuito de activación de un actuador LED sobre protoboard. Imagen de elaboración propia.

$$V_{LED_OUTPUT} = 3'3V, N_1 = 1'58 V, N_2 = 700 mV, N_3 = 780 mV$$

$$I_1 = 18'45 mA, I_2 = 18'42 mA, I_3 = 0'15 mA, I_4 = 0'04 mA$$

$$I_5 = 4'33 mA, I_6 = 4'34 mA, I_7 = 14'15 mA, I_8 = 15'64 mA, I_{led} = 1'48 mA$$

Conclusión:

Se ha observado en el circuito sobre protoboard que la corriente mínima de base en los transistores BJT NPN, cuando actúan como interruptores, es de $18'42\text{ mA}$ (I_2) y de $0'04\text{ mA}$ (I_4).

Esto significa que la corriente de base medida ($I_{b(\min medida)} = 0'04 \times 10^{-3} A$) es mucho mayor que la corriente de base mínima requerida para garantizar la saturación del transistor $I_{b(min)} = 66'4 \times 10^{-6} A$.

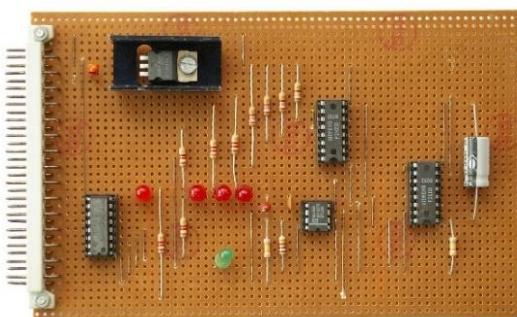
Por lo tanto, se puede asegurar la saturación de los transistores NPN en este circuito.

Al igual que en el anterior circuito (acondicionamiento de señal tensión batería), debido a la complejidad en la modelización de los componentes activos y la falta de opciones de personalización en el simulador, se ha decidido no simular este circuito.

3.1.3. PCB

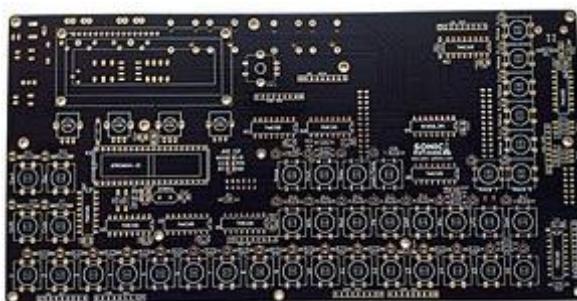
Una PCB (Printed Circuit Board) es una placa de circuito impreso, que es una herramienta importante en la creación de dispositivos electrónicos. Está diseñada para conectar componentes electrónicos mediante la creación de pistas conductoras y orificios de montaje para que los componentes estén conectados eléctricamente y en su lugar.

Debido a las restricciones de tiempo para el desarrollo de este proyecto, se optó por diseñar directamente la placa de circuito impreso (PCB) sin utilizar una “breadboard” como paso previo. Para asegurar el correcto funcionamiento de los circuitos, se realizaron simulaciones previas en protoboard.



32. Breadboard. Fuente: Wikipedia [9].

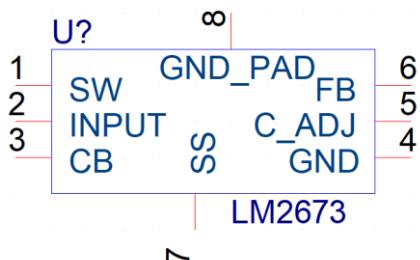
La utilización de una PCB ofrece numerosas ventajas en comparación con una “breadboard”. En primer lugar, las conexiones eléctricas se encuentran en el interior de la placa, lo que proporciona mayor fiabilidad y estabilidad al reducir la interferencia y el ruido eléctrico. En segundo lugar, una PCB puede soportar cargas más pesadas y, por lo tanto, más corriente eléctrica que una “breadboard”. En tercer lugar, una PCB es más compacta y estética, ya que los componentes se disponen de manera más ordenada y no hay cables sueltos. Por último, una PCB es más fácil de replicar en masa, ya que se puede crear el diseño una vez y luego producir múltiples copias, mientras que, con una “breadboard”, cada dispositivo debe construirse individualmente.



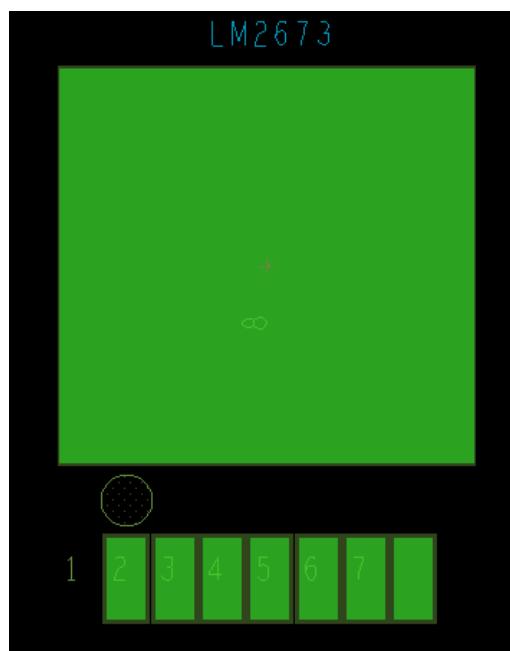
33. PCB. Fuente: Wikipedia [10].

El diseño de ambas unidades de control (transmisora y receptora) se han realizado con el programa *Orcad Allegro* y se han desarrollado respetando las recomendaciones del fabricante de cada circuito integrado.

Para obtener el diseño de la PCB final, se ha seguido un proceso riguroso que incluyó la creación del esquemático de cada circuito utilizando ORCAD Capture CIS. Luego, se procedió a crear los símbolos de los componentes, sus correspondientes “*footprint*”, y a relacionarlos adecuadamente entre ellos. Una vez finalizada esta etapa, se importó el diseño del esquemático al programa de PCB Orcad Editor, donde se realizó el diseño de la PCB con sus respectivas restricciones y peculiaridades. Este proceso permitió obtener un diseño preciso y detallado que asegura el correcto funcionamiento del circuito, así como una correcta integración de los componentes en la PCB.



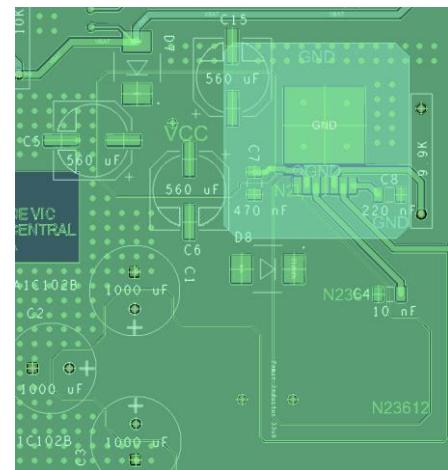
34. Símbolo creado para el componente LM2673.
Imagen de elaboración propia.



35. Footprint creado para el componente LM2673.
Imagen de elaboración propia.

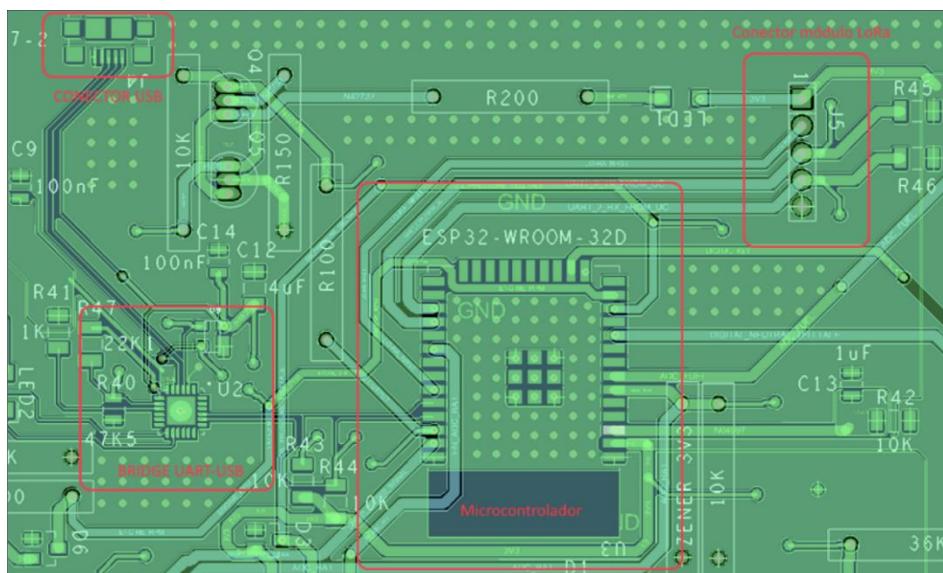
3.1.3.1. PCB Transmisora

La ECU se alimenta mediante la tensión suministrada por la batería, la cual es protegida mediante un diodo Schottky (con baja caída de tensión) para evitar dañar el cableado de la moto y sus unidades de control en caso de cortocircuito en la PCB. El microcontrolador y el módulo LoRa, que necesitan ser alimentados a 3'3 V, obtienen esta tensión a partir de la batería y gracias al convertidor Buck LM2673 de Texas Instruments que reduce la tensión. De esta manera, se garantiza una alimentación segura y estable para los componentes electrónicos (ver 36. PCB transmisora: Potencia. Imagen de elaboración propia. Anexo "7.2.4. Análisis del convertidor de potencia de la unidad de control transmisora").



En el microcontrolador ESP-32 se disponen de tres puertos de comunicación UART. El puerto nativo por defecto es el UART0, mientras que para usar el puerto 1 (UART1) y el puerto 2 (UART2) se deben configurar los GPIO para tal fin. En este proyecto se ha decidido utilizar el puerto UART0 para programar el microcontrolador mediante conexión USB y el puerto UART2 para comunicarse con el módulo LoRa.

Para programar el microcontrolador mediante conexión USB es necesario un conversor de UART a USB, que es llevado a cabo por el circuito integrado CP2102. Finalmente, se ha incluido un LED para indicar el estado del sistema: transmitiendo datos o en espera.

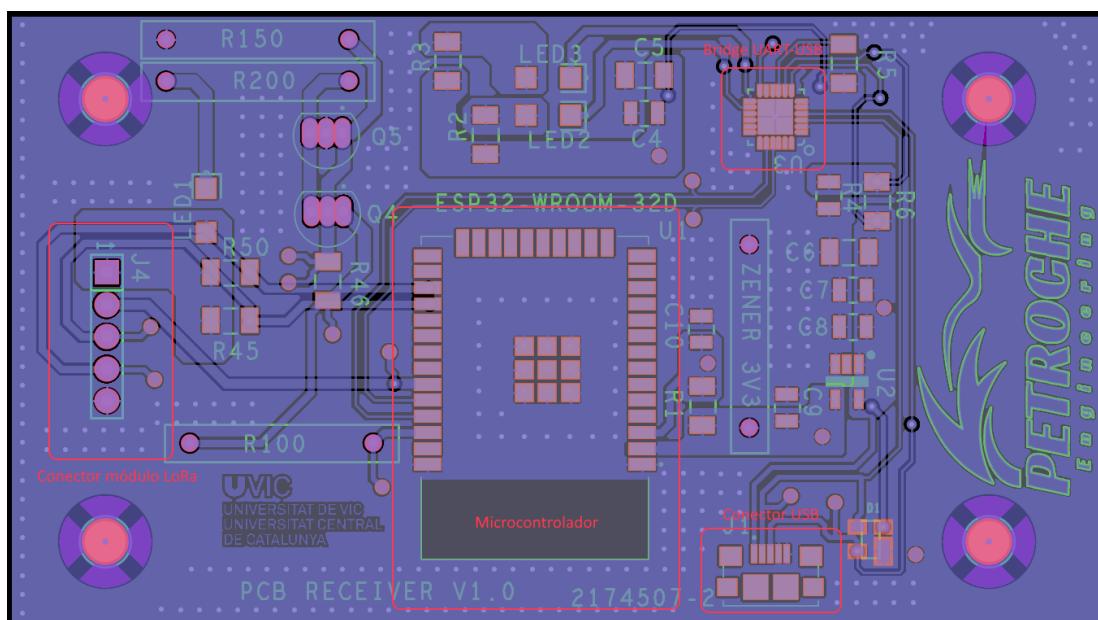


37. PCB transmisora: Conector USB, CP2102, ESP-32 y RYLR998. Imagen de elaboración propia.

3.1.3.2. PCB Receptora

Para la ECU receptora, se ha decidido utilizar un regulador lineal de baja caída de tensión (LDO) para obtener los 3'3 V de suministro en lugar de un conversor Buck. La elección del LDO se debe a que el microcontrolador tiene un bajo consumo de energía y no hay otros componentes que necesiten ser alimentados. Lo que hace que, según las mediciones de consumo realizadas, el rendimiento del LDO sea suficiente para proporcionar la energía necesaria para los pocos componentes que hay que alimentar. Además, el uso del LDO simplifica el diseño y reduce la cantidad de componentes necesarios para la alimentación de la ECU receptora.

La unidad de control receptora es significativamente más simple en comparación con la unidad transmisora. Consta únicamente del microcontrolador, un conector USB que proporciona la alimentación de 5 V al módulo, un LDO que reduce la tensión de 5 V a 3'3 V para alimentar el microcontrolador, un conversor UART a USB, un conector para el módulo LoRa, y un LED con los componentes necesarios para indicar si se encuentra en espera o funcionando.



38. PCB Receptora: Conector módulo RYRL998, microcontrolador ESP-32, Bridge UART a USB y Conector USB. Imagen de elaboración propia.

3.2. Diseño del software

Este proyecto requiere de una interacción adecuada entre el hardware y el software para su correcto funcionamiento. El software es esencial, ya que permite el control y la gestión de los componentes electrónicos, así como el procesamiento de las señales obtenidas. Además, un buen diseño de software puede optimizar el rendimiento y la eficiencia de todo el sistema, lo que resulta fundamental para lograr los objetivos del proyecto. Por lo tanto, es fundamental dar la debida importancia al software y desarrollarlo de manera óptima.

3.2.1. Lógica de la máquina de estados y unidad de control transmisora

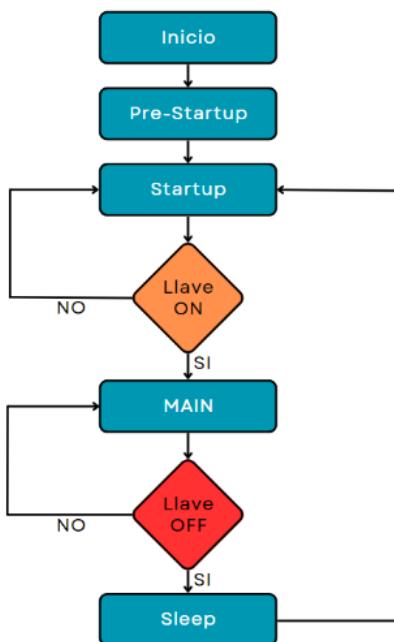
Con el fin de obtener un control efectivo y eficiente del comportamiento del sistema en función de los eventos que se producen, se ha decidido integrar una máquina de estados en la unidad de control transmisora.

La máquina de estados es una técnica de programación comúnmente utilizada en sistemas embebidos. Permite controlar el comportamiento del sistema en función de los eventos que se producen, lo que garantiza un comportamiento coherente y predecible del sistema.

La máquina de estados de la ECU Transmisora consta de 4 estados diferentes:

- Estado “Pre-Startup”: Es el primer estado por el que el programa pasa al encenderse el microcontrolador. Sólo se accede a este estado cuando el microcontrolador recibe batería por primera vez. A lo largo de este estado se configuran los parámetros del módulo LoRa, asignación de pines usados, inicialización de temporizadores, ajustes del ADC y los parámetros de los puertos seriales UART.
- Estado “Startup”: Es el segundo estado por el que pasa el programa durante una ejecución normal, aunque también se accede a este estado si previamente el programa pasa por el estado “Sleep” y se activa la llave del vehículo. Durante este estado se monitoriza el valor de la señal de llave, para que, en caso de activarse se permita pasar al siguiente estado “Main”.
- Estado “Main”: Este estado es el principal de todo el programa. Cuando el programa entra en él, primero de todo se envía una señal lógica de 1 al pin EN_BAT_ADC que consecuentemente cierra el transistor MOSFET que permite leer la tensión de batería mediante un canal ADC microcontrolador ESP-32. Durante este estado también se monitoriza el estado de la llave, de forma que, si se detecta que ésta se desactiva se pasará al siguiente estado “Sleep”.

- Estado “Sleep”: El estado “Sleep” está pensado para reducir consumos y mantener el estado en Standby hasta que el sistema detecte que se vuelve a accionar la llave y encender el vehículo. Cuando el programa entra en este estado primero de todo envía una señal lógica de 0 al pin EN_BAT_ADC para abrir el interruptor MOSFET. Esto se hace para ahorrar consumo y como precaución por cualquier posible cruce ya que sería crítico para los sistemas del vehículo.



39. Diagrama lógico de la máquina de estados ECU Sender. Imagen de elaboración propia.

En caso de que el programa obtenga un estado diferente de cualquiera de los estados anteriormente mencionados, se informa mediante comunicación serial del error y se reconduce al estado Pre-Startup.

```

default:
    Serial.println("Invalid State Machine. Rebooting the system...");
    STATE_MACHINE = STM_PRE_STARTUP;
    break;
}
  
```

40. Código de un estado no contemplado. Imagen de elaboración propia.

Durante cada iteración del microcontrolador al programa, se accede a la máquina de estados y al final se ejecutan una serie de acciones o funciones que pueden o no depender del estado actual del programa. Entre ellas están la función “LedManagerProcess(STATE_MACHINE)” que activa un patrón de parpadeo según el estado en el que se encuentre el programa, y se realizan acciones para llevar el control correcto del tiempo durante la ejecución del programa.

3.2.2. Multitarea y acciones en paralelo

Los más observadores se habrán percatado que durante la explicación de la lógica de la máquina de estados de la ECU Transmisora no se ha mencionado la adquisición o envío de datos por radiofrecuencia. Debido a que el ESP-32 es un microcontrolador que cuenta con dos núcleos, lo que lo convierte en una excelente opción para aplicar técnicas de programación multihilo [3].

Es posible utilizar uno de los núcleos para realizar la adquisición de datos y otro para el envío, lo que permite que ambas tareas se ejecuten simultáneamente en paralelo, sin interrupciones entre ellas. Esto mejora significativamente el rendimiento y la eficiencia del sistema, especialmente en aplicaciones como esta donde la velocidad de adquisición y envío de datos es crítica.

De forma que, el núcleo 1 del microcontrolador se encarga del muestreo de datos de la moto. Y el núcleo 0 se destina a enviar los datos adquiridos, controlar la lógica de la máquina de estados, contadores y patrón de parpadeo del LED.

Tal y como se aprecia en la siguiente captura, la función “*TaskDataSampling*” es la encargada de muestrear todos los datos ya acondicionados de la moto. El núcleo que está asignado a esta tarea (núcleo 1) adquiere los datos de forma cíclica siempre y cuando la llave esté en estado ON.

```
void TaskDataSampling(void *pvParameters)
{
    while (1)
    {
        if (getKeyStatus() == 1)
        {
            if (DEBUG_MODE == 1) ...
            else
            {
                adc_battery_value = getBatteryVoltage();
                adc_fuel_value = getFuel();
                NeutralGear_value = getNeutralGearStatus();
                OilPressure_value = getOilPressureStatus();
                Key_value = getKeyStatus();
                Speedometer_value = getSpeed();
                Tachometer_value = getRPM();

            }
            LoraBuffTxAll = (adc_battery_value & 0xFF) +
                ((OilPressure_value & 0b0001) << 8) +
                ((NeutralGear_value & 0b0001) << 9) +
                ((Tachometer_value & 0xFF) << 10) + //Se limita la variable hasta 0xFF valores
                ((Speedometer_value & 0xFF) << 18) +
                ((adc_fuel_value & 0x7F) << 26);

            Serial.print(LoraBuffTxAll,BIN); Serial.print("   "); Serial.println(LoraBuffTxAll);
            //Serial.print(LoraBuffTx1,BIN); Serial.print("   "); Serial.print(LoraBuffTx1); Serial.print("   ");
            //vTaskDelay(100);
        }
        else
        {
            //Do nothing
        }
    }
}
```

41. Código fuente de la función para el muestreo de datos “*TaskDataSampling*”. Imagen de elaboración propia.

Estos datos se refrescan en las variables globales, así, la función designada para enviar los datos pueda tener acceso a ellos siempre que lo necesite.

Para enviar los datos en el sistema, se utiliza la función "TaskDataSending", la cual está asignada al núcleo 0. En primer lugar, esta función revisa el estado de la llave. Si la llave está activada, la función procede a declarar un vector de tipo CHAR con una longitud máxima de 64 elementos, lo que ayuda a prevenir la posibilidad de un "overflow" y la pérdida de datos o la obtención de valores erróneos en caso de que se llene la variable.

A continuación, se utiliza una función nativa de C para convertir la variable numérica donde están los datos codificados a una variable de tipo CHAR. Esto se hace para que los datos puedan ser enviados a través del puerto serial UART 2 utilizando comandos "AT" (en los siguientes puntos se explica la codificación de los datos y comandos AT).

Una vez que se han convertido los datos a tipo CHAR, se envían al módulo de LoRa para su transmisión al módulo receptor. Es importante tener en cuenta que la transmisión y recepción de datos a través de LoRa es un proceso complejo y puede requerir una cuidadosa optimización para garantizar que los datos se transmitan y reciban de manera confiable y eficiente.

```
void TaskDataSending(void *pvParameter)
{
    while (1)
    {
        if (getKeyStatus() == 1)
        {
            char CHAR_LoraBuffTxAll[64];

            itoa(LoraBuffTxAll, CHAR_LoraBuffTxAll, 10); //Se usa la función nativa en C itoa para convertir string a char
            //Serial.print(LoraBuffTxAll, BIN); Serial.print(" "); Serial.print(CHAR_LoraBuffTxAll); Serial.print(" "); Serial.println(strlen(CHAR_LoraBuffTxAll));

            Serial2.println("AT+SEND=" + lora_RX_address + "," + strlen(CHAR_LoraBuffTxAll) + "," + CHAR_LoraBuffTxAll);
            vTaskDelay(1); //ms
        }
        else
        {
            //do nothing
        }
        vTaskDelay(1); //delay añadido a propósito para evitar que salte el watchdog a causa de un while "vacío" cuando no hay llave
    }
}
```

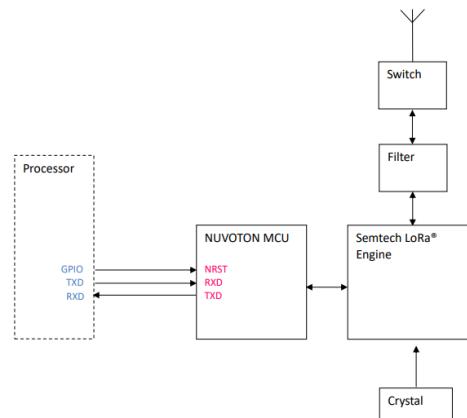
42. Código fuente de la función para la transmisión de datos "TaskDataSendind". Imagen de elaboración propia.

Es importante mencionar que se ha añadido un delay de 1ms dentro de la función "TaskDataSending". Esto se debe a que, dado que la función se ejecuta en un bucle infinito ("while (1)"), si la llave se encuentra desconectada, el programa se queda en bucle en esta función. Como resultado, el watchdog del microcontrolador se activa y reinicia el sistema.

Para evitar que esto suceda, se ha agregado el delay de 1ms para que la función "TaskDataSending" tenga un pequeño tiempo de espera antes de volver a ejecutarse en cada ciclo del bucle. Esto garantiza que la función no consuma todos los recursos del microcontrolador y permite que el programa siga ejecutándose sin problemas en caso de que la llave esté desconectada.

3.2.3. Comunicación con el módulo RYLR998 y comandos AT

El módulo RYLR998 contiene embebido diversos componentes, los más relevantes son el microcontrolador que hace de transceiver UART, el módulo en si de comunicación LoRa y la antena transmisora.



43. Diagrama de bloques RYLR998. Fuente: Reyax [11].

Para poder comunicarse con el módulo LoRa se debe de hacer a través de un puerto UART y utilizando comandos AT. Los comandos AT se utilizan para configurar y controlar las funciones del dispositivo o módulo conectado. Tienen una estructura estándar que incluye el prefijo "AT" seguido de una serie de caracteres que representan el comando y sus parámetros. El comando y los parámetros están separados por un signo de igual ("="). Seguido de una separación por comas (",") y terminan con un carácter de retorno de carro ("\r") y un salto de línea ("\n").

Estos comandos no están estandarizados, por lo que, cada fabricante puede tener su propia versión de comandos AT. El fabricante del módulo RYLR998 (Reyax) tiene implementado diversos comandos [12] (se pueden consultar todos en el apartado de Anexos “7.7. Comandos AT RYLR998”), aunque lo más importantes son:

- “AT+ADDRESS”: Se utiliza para configurar la dirección del módulo que transmite.
- “AT+BAND”: Se utiliza para establecer la banda de radiofrecuencia a usar. En Europa se establece la banda permitida de 865 MHz para comunicaciones LoRa sin necesidad de obtener una licencia de radioaficionado.
- “AT+PARAMETER”: Esta función cuenta con diversos parámetros configurables para la transmisión como el Spreading Factor, Bandwidth, Coding Rate y Programmed Preamble. (Explicación más en detalle de estos parámetros en anexos).
- “AT+SEND”: Para enviar cualquier dato mediante LoRa se debe enviar “AT+SEND=” seguido de la dirección del módulo destinatario, la cantidad de bytes que se enviarán, los bytes de datos a enviar y todo esto separado por comas.
- “+RCV”: Cuando se recibe cualquier dato se recibe con el siguiente formato: “+RCV=” seguido de la dirección del módulo transmisor, la cantidad de bytes de información, los

bytes de datos, un parámetro RSSI que indica la fuerza de la señal recibida, un parámetro SNR que mide la relación señal a ruido y todo esto separado también por comas.

De forma que, utilizando los comandos implementados por Reyax en el RYLR998 [12], la configuración de los módulos transmisor y receptor queda de la siguiente manera.

```
String lora_band = "865000000";           //Banda de frecuencia (Hz)
String lora_networkid = "18";             //Identificación de la red Lora
String lora_address = "1";                //Dirección del módulo
String lora_RX_address = "2";             //Dirección del módulo receptor
String lora_mode = "1";                  //Transceiver mode
String lora_baudrate = "115200";          //UART Baudrate
String lora_parameter_spreading_factor = "7"; //Cuanto mayor sea el valor de spreading factor, mayor será la distancia
                                                //de cobertura pero menor será la velocidad de transmisión.
String lora_parameter_bandwidth = "9";     // Un ancho de banda más amplio permite una mayor tasa de bits, pero
                                                //a costa de una mayor potencia de transmisión.
String lora_parameter_coding_rate = "1";    //Una tasa de codificación más alta proporciona una mayor resistencia
                                                //a la interferencia, pero a costa de una mayor latencia de transmisión.
String lora_parameter_programmed_preamble = "4"; //Un preámbulo más largo proporciona una mayor robustez
                                                //en la sincronización, pero a costa de una mayor latencia de transmisión.
```

45. Código fuente de los parámetros configurados en el módulo de transmisión LoRa. Imagen de elaboración propia.

```
String lora_band = "865000000";           //Banda de frecuencia (Hz)
String lora_networkid = "18";             //Identificación de la red LORA
String lora_address = "2";                //Dirección del módulo sender
//String lora_RX_address = "1";           //Dirección del módulo receptor receiver NO USADA
String lora_mode = "1";                  //Transceiver mode
String lora_baudrate = "115200";          //UART Baudrate
String lora_parameter_spreading_factor = "7"; //Cuanto mayor sea el valor de spreading factor, mayor será la
                                                //distancia de cobertura pero menor será la velocidad de transmisión.
String lora_parameter_bandwidth = "9";     // Un ancho de banda más amplio permite una mayor tasa de bits, pero
                                                //a costa de una mayor potencia de transmisión.
String lora_parameter_coding_rate = "1";    //Una tasa de codificación más alta proporciona una mayor resistencia
                                                //a la interferencia, pero a costa de una mayor latencia de transmisión.
String lora_parameter_programmed_preamble = "4"; //Un preámbulo más largo proporciona una mayor robustez en la
                                                // sincronización, pero a costa de una mayor latencia de transmisión.
String lora_address_receive_data_from = "1";
```

44. Código fuente de los parámetros configurados en el módulo receptor LoRa. Imagen de elaboración propia.

Nótese que el módulo receptor no usa la variable “*lora_RX_address*” debido a que es una comunicación unidireccional donde el transmisor sólo transmite y el receptor sólo recibe, por lo tanto, no requiere de dirección a enviar.

Es importante destacar que, para que los módulos puedan establecer una comunicación punto a punto, es necesario que los parámetros “*band*”, “*networkid*”, “*spreading factor*”, “*bandwidth*”, “*coding rate*” y “*programmed preamble*” sean idénticos en ambos nodos LoRa.

La configuración de los parámetros mencionados se realiza de la siguiente manera una vez al iniciarse el microcontrolador.

```
delay(1500);
Serial2.println("AT+BAND=" + lora_band);
delay(500);
Serial2.println("AT+NETWORKID=" + lora_networkid);
delay(500);
Serial2.println("AT+ADDRESS=" + lora_address);
delay(500);
Serial2.println("AT+MODE=" + lora_mode);
delay(500);
Serial2.println("AT+IPR=" + lora_baudrate);
delay(500);
Serial2.println("AT+PARAMETER=" + lora_parameter_spreading_factor + "," + lora_parameter_bandwidth + "," + lora_parameter_coding_rate + "," + lora_parameter_programmed_preamble);
delay(1500);
```

46. Código fuente de la configuración del módulo LoRa RYLR998. Imagen de elaboración propia.

3.2.4. Codificación de los datos

Para transmitir los datos entre los módulos, una posible opción es enviar cada dato de forma individual y que el receptor los vaya ordenando en un buffer por orden de entrada FIFO y tratando de forma secuencial. Por ejemplo:

“AT+SEND=2, Nº data bytes, adc_battery_value”

“AT+SEND=2, Nº data bytes, adc_fuel_value”

“AT+SEND=2, Nº data bytes, Neutral_gear_value”

“AT+SEND=2, Nº data bytes, OilPressura_value”

“AT+SEND=2, Nº data bytes, Tachometer_value”

Este método presenta ciertos riesgos, dado que la pérdida de una trama resulta en un desfase que puede generar errores difíciles de identificar. Además, la velocidad actual del bus serial UART exige un intervalo mínimo de 1 ms entre trama y trama para garantizar su correcta transmisión. En consecuencia, se trata de una opción poco eficiente, ya que usando este método el envío de todos los datos puede tardar al menos 5 ms.

Otra opción, que es la elegida en este proyecto, es la codificación de todos los datos en un número entero expresado en formato decimal de hasta 4 bytes de tamaño.

Sabiendo el valor máximo y mínimo de las variables que adquiere el sistema, se proceden a ordenarse de la siguiente manera:

- Bits 26-32: valor limitado de “adc_fuel_value” [0, 127].
- Bits 18-25: valor limitado de “Speedometer_value” [0, 255].
- Bits 10-17: valor limitado de “Tachometer_value” [0, 255].
- Bit 9: valor de “NeutralGear_value” [0 o 1].
- Bit 8: valor de “OilPressure_value” [0 o 1].
- Bits 0-7: valor limitado de “adc_battery_value” [0, 255].

De forma que, si la unidad de control receptora obtiene el siguiente número decimal inventado “1528503749” cuando lo decodifique desplazando los bits obtendrá los datos transmitidos y los podrá asignar a su correspondiente variable

Siguiendo con el ejemplo, el número recibido en decimal es el “1528503749”, lo que en formato binario se expresa como “01011011000110110001110111000101”. Al separar los bits y asignarlos a sus correspondientes variables, se obtiene la siguiente información:

- ADC_BATTERY_VALUE = 1100 0101 (binario) = 197 (decimal)
- OILPRESSURE_VALUE = 1 (binario) = 1 (decimal)
- NEUTRAL_GEAR_VALUE = 0 (binario) = 0 (decimal)
- TACHOMETER_VALUE = 1000 1110 (binario) = 142 (decimal)
- SPEEDOMETER_VALUE = 1000 1101 (binario)= 141 (decimal)
- ADC_FUEL_VALUE = 010 1101 (binario) = 45 (decimal)

Esto indica que el vehículo en ese momento tiene una tensión de batería de 19'7 voltios, una correcta presión de aceite, no tiene engranada la marcha neutral, el motor está girando a 14200 revoluciones por minuto, se desplaza a una velocidad de 141 km/h y el nivel de combustible es del 45%.

	ADC_FUEL_VALUE								SPEEDOMETER_VALUE								TACHOMETER_VALUE								N E U T R A L	O I L	ADC_BATTERY_VALUE							
Número de BIT	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
BIT de dato	0	1	0	1	1	0	1	1	0	0	0	0	1	1	0	1	1	0	0	0	1	1	1	0	0	1	1	1	0	0	0	1	0	1

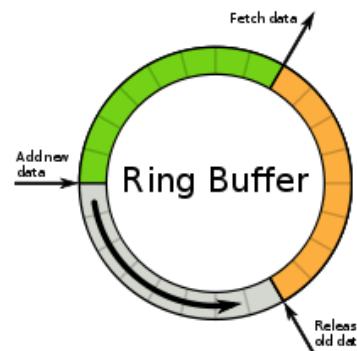
47. Representación bit a bit de la codificación de los datos en una sola variable. Imagen de elaboración propia.

El método de enviar todos los datos codificados en una sola variable ofrece una menor latencia y disminuye la probabilidad de errores en comparación con la opción de enviar cada dato en un mensaje independiente. Con este nuevo método, se requiere un tiempo mínimo de 1 ms para enviar todos los datos, mientras que con el método anterior se necesitaban al menos 5 ms. Por lo tanto, se ha logrado mejorar la latencia en hasta cinco veces utilizando este nuevo método.

3.2.5. Recepción de datos y unidad de control receptora

La unidad de control receptora no cuenta con una máquina de estados como la unidad de control transmisora. Esto es debido a que, para la funcionalidad que tiene la unidad de control se requiere de un código secuencial e implementar un código multihilo que realice diversas operaciones en paralelo podría inducir a errores.

El programa en esta unidad de control lee los datos que recibe por el monitor serial UART2 provenientes del módulo RYLR998 y los almacena en un buffer circular de 60 elementos. Depura los bytes entrantes eliminando los datos innecesarios y devuelve al ordenador sólo los 4 bytes que contienen la información para posteriormente ser graficadas por pantalla.



48. Representación de un buffer circular.

Fuente: Wikipedia [13].

Depurar los mensajes entrantes y obtener la información relevante ha sido un proceso bastante complicado debido a que el sistema recibe un nuevo mensaje para ser procesado, depurado y enviado cada mínimo 1 ms y 2 ms como máximo.

El método que he encontrado más eficiente ha sido el siguiente, supongamos que recibimos por el puerto serial UART los siguientes 30 bytes de información:

Orden llegada	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Representación [ASCII]	LF	+	R	C	V	=	1	,	1	1	,	-	1	9	0	6	6	8	0	9	4	3	,	-	1	8	,	1	3	CR
Datos UART [Decimal]	10	43	82	67	86	61	49	44	49	49	44	45	49	57	48	54	54	56	48	57	52	51	44	45	49	56	44	49	51	13

49. Trama de ejemplo recibida mediante LoRa. Imagen de elaboración propia.

Como se ha mencionado previamente, cada byte se almacena en orden de llegada en una posición diferente del buffer circular. Se le llama buffer circular a este tipo de arreglo porque, cuando se llena de valores, el valor más antiguo se sobrescribe. En este buffer circular, se busca un elemento que contenga el valor 61, que en ASCII representa el signo "=", y a partir de este elemento se almacenan los próximos 16 elementos en un vector con un offset aplicado de 4 elementos.

En esta etapa, se realiza una primera depuración ignorando todos los valores hasta recibir un carácter ASCII "=", lo que implica ignorar los bytes "LF", "+", "R", "C", "V", "1", ",", y "1". Después, se almacenan solamente los 16 bytes siguientes al valor decimal 61, lo que implica ignorar los bytes "8", ",", "1", "3", y "CR".

Con estas acciones, se pueden localizar los datos que contienen información en un vector de 16 bytes ASCII. Cabe destacar que se ha realizado una depuración poco conservadora, lo que significa que no se han descartado dos bytes ASCII antes de los bytes de información y tres bytes al final. Sin embargo, esto no afecta el proceso, ya que posteriormente se filtrarán de los bytes que contienen información.

Luego, se debe distinguir entre un vector correcto y uno con errores. Para ello, se recorre el vector que almacena los siguientes bytes ASCII "1,-1906680943,-1". Según el protocolo que usa LoRa, los bytes de datos siempre se devuelven entre dos comas. Por lo tanto, se busca en este vector estos caracteres y se cuenta cuántas veces aparecen. Si este contador es igual a cero o mayor que dos, se sale de la función "for" que recorre el vector y se establece una variable booleana como falsa. Si el contador es igual a 1, no se hace nada porque todavía no se ha encontrado la segunda coma. Si el contador es igual a 2, se establece la variable booleana "data_ok" como verdadera.

```
23:08:04:419 -> 10  
23:08:04:430 -> 43  
23:08:04:431 -> 82  
23:08:04:431 -> 67  
23:08:04:431 -> 86  
23:08:04:432 -> 61  
23:08:04:432 -> 49  
23:08:04:433 -> 44  
23:08:04:433 -> 49  
23:08:04:433 -> 49  
23:08:04:434 -> 44  
23:08:04:434 -> 45  
23:08:04:434 -> 49  
23:08:04:435 -> 57  
23:08:04:435 -> 48  
23:08:04:435 -> 54  
23:08:04:436 -> 54  
23:08:04:436 -> 56  
23:08:04:436 -> 48  
23:08:04:437 -> 57  
23:08:04:437 -> 52  
23:08:04:437 -> 51  
23:08:04:438 -> 44  
23:08:04:438 -> 45  
23:08:04:438 -> 49  
23:08:04:439 -> 56  
23:08:04:439 -> 44  
23:08:04:439 -> 49  
23:08:04:440 -> 51  
23:08:04:440 -> 13
```

50. Datalog de datos recibidos por ECU Receptora. Imagen de elaboración propia.

Posteriormente, se revisa esta variable booleana, y si es verdadera, se envía el vector completo a través del puerto serial al ordenador, exceptuando el primer byte, que siempre será una coma.

```
int counter_delimiter = 0;
int delimiter = 44;
int start_index = 0;
int finish_index = 0;
int k = 0;

for (int j=0; j<15 ; j++)
{
    if(VectorDataRaw[j] == delimiter)
    {
        counter_delimiter = counter_delimiter + 1;
        if (counter_delimiter > 2)
        {
            data_ok = false;
            break;
        }
        else
        {
            //do nothing
        }

        for (k = j ; k<15 ; k++)
        {
            VectorData[k] = VectorDataRaw[k];
            if(counter_delimiter == 2)
            {
                data_ok = true;
            }
            else if (counter_delimiter > 2)
            {
                data_ok = false;
                break;
            }
            else if (counter_delimiter == 1) //posiblemente encontrará otra delimiter o no
            {
                data_ok = false;
                //do nothing
            }
            else
            {
                data_ok = false;
                break;
            }
        }
    }
    else
    {
        //do nothing.
    }
}
```

51. Código fuente usado para la depuración del vector lineal. Imagen de elaboración propia.

3.3. Aplicación de escritorio *Graphical User Interface*

Una GUI es una interfaz de usuario con elementos gráficos como botones, iconos, menús y ventanas, para permitir que los usuarios interactúen con el software o sistema de manera más intuitiva y visual.

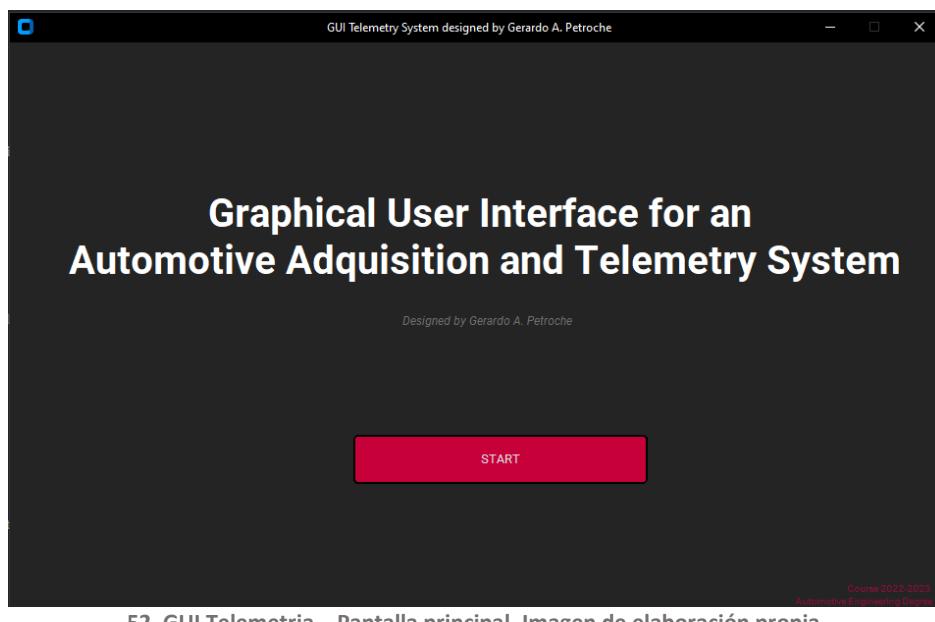
Para este trabajo de fin de grado se ha desarrollado una aplicación de escritorio propia para poder mostrar los datos adquiridos en tiempo real.

La aplicación ha sido desarrollada en Python, utilizando principalmente las librerías "CustomTkinter" [14], "PIL", "Time", "CSV", "Matplotlib" [15], "Numpy" [16] y "Serial" [17].

La librería "CustomTkinter" ha sido fundamental para crear la interfaz gráfica de la aplicación, ya que se han utilizado sus elementos visuales. Asimismo, la librería "Serial" se ha utilizado para establecer la conexión con los puertos seriales disponibles en el PC y obtener los datos de ellos.

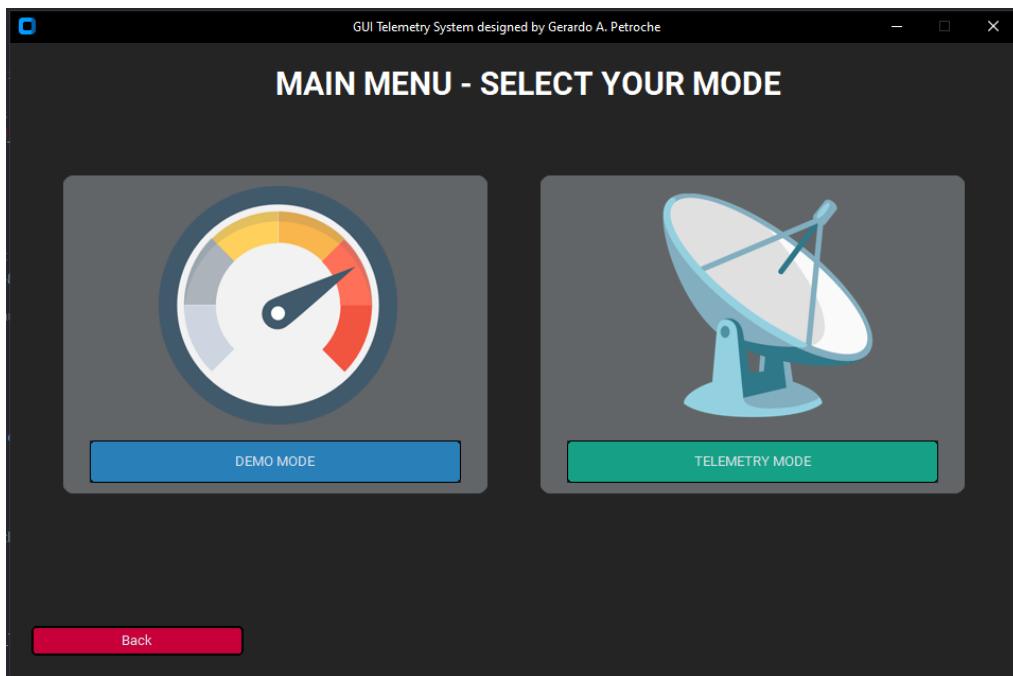
Por otro lado, las restantes librerías han sido utilizadas para el procesamiento y visualización de los datos. En resumen, estas herramientas han permitido desarrollar una aplicación con una interfaz gráfica intuitiva y funcional que permite el procesamiento de datos y su representación visual de manera eficiente.

Al iniciar la aplicación, aparece la pantalla principal:



52. GUI Telemetria – Pantalla principal. Imagen de elaboración propia.

Cuando el usuario pulse el botón central START pasará a la pantalla de menú, donde podrá escoger entre dos modos: DEMO MODE y TELEMETRY MODE.



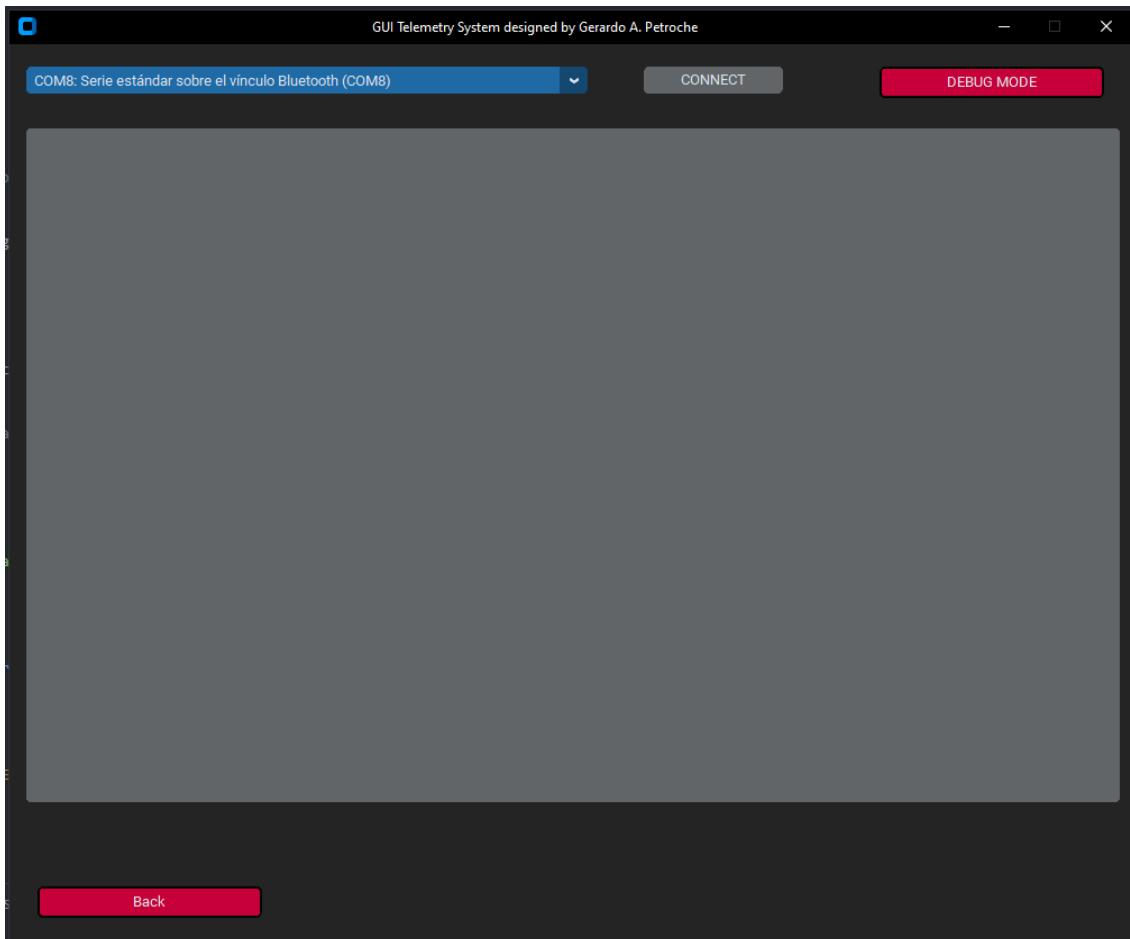
53. GUI Telemetria – Pantalla de menú. Imagen de elaboración propia.

En caso de que el usuario presione el botón azul “DEMO MODE”, como su nombre indica, aparecerá en una pantalla con gráficas de ejemplo actualizándose cada segundo con valores aleatorios. Cabe destacar que este modo se ha desarrollado con la intención de que el usuario pueda hacerse una idea de cómo se mostrarán los datos recibidos por el sistema de telemetría.



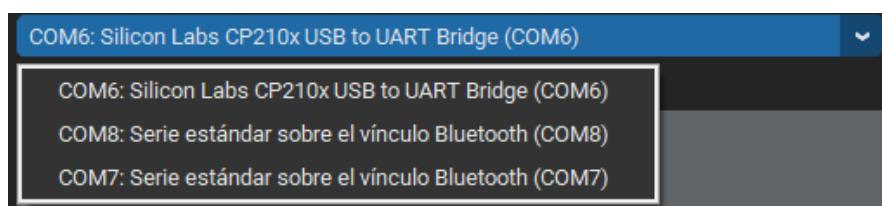
54. GUI Telemetria – Pantalla modo demo. Imagen de elaboración propia.

En caso de que el usuario, en vez de pulsar el botón “DEMO MODE”, pulse el botón “TELEMETRY MODE” entrará en el modo telemetría de la aplicación. En un principio se mostrará una pantalla un tanto vacía, esto es debido a que no se mostrará ninguna gráfica hasta que no se empiecen a recibir datos por el puerto serial.



55. GUI Telemetria – Pantalla de telemetría. Imagen de elaboración propia.

Como bien se puede observar, en la parte superior izquierda, hay un combobox desplegable de color azul que muestra las conexiones seriales detectadas en el momento de haber seleccionado el modo telemetría. Para realizar correctamente la vinculación con la ECU Receptora, se debe buscar la conexión con el bridge UART-USB CP210x del fabricante Silicon Labs.



56. Combobox de puertos seriales disponibles. Imagen de elaboración propia.

En caso de no aparecer la conexión con el integrado de Silicon Labs, reiniciar la aplicación de escritorio y cerciorarse que el cable USB está correctamente conectado en un puerto válido del PC.

Es posible que no aparezca la conexión en el mismo puerto COM6, esto se puede deber al hecho de tener más o menos dispositivos seriales conectados en el ordenador en ese preciso momento. En caso de aparecer con otro número de puerto (ejemplo: COM12) la comunicación y conexión se podrá establecer correctamente.

Posteriormente de seleccionar el puerto COM correspondiente a nuestra unidad de control receptora, se debe pulsar el botón “CONNECT” para iniciar la comunicación entre PC-ECU.

En caso de establecerse la comunicación correctamente, el combobox desaparecerá y se mostrará en su lugar el mensaje “SUCCESSFULLY CONNECTED”.

Desde el primer instante en que la ECU Receptora transmita los datos al PC, se empezarán a graficar todos los canales.

Debido a que graficar los datos en tiempo real requiere un coste computacional alto y, que Python al ser un lenguaje interpretado es demasiado lento para tal propósito, no se pueden actualizar con la misma latencia en que se reciben, por lo que se tendría que optar por almacenar estos datos en un buffer y graficar el buffer entero. La representación gráfica de los datos recibidos no se ha podido acabar de implementar por falta de tiempo.

Para poder mostrar los mensajes ya decodificados con la máxima latencia posible se ha implementado el DEBUG MODE. Este modo se activa al presionar el botón rojo de la parte superior derecha en la pantalla de “TELEMETRY MODE”. Este botón activa una ventana en la zona inferior de la aplicación, que muestra en tiempo real los datos recibidos y decodificados.



57. GUI Telemetria – Ventana de debug. Imagen de elaboración propia.

En el apartado “3.2.5. Recepción de datos y unidad receptora” se estableció que los datos serían enviados al PC mediante vectores de datos que contenían dos comas “,”. El programa de Python, tanto en el modo de telemetría normal como en el modo debug, lee los datos entrantes del puerto serial para recibir estos vectores de datos válidos. Los datos entrantes se leen, decodifican en UTF-8 y se eliminan los espacios en blanco para su procesamiento.

Luego, el string se divide en una lista de tres elementos separados por las comas, y los datos relevantes se extraen del segundo elemento de la lista. Estos datos se decodifican desplazando los bits y se almacenan en sus variables respectivas, como "battery_voltage_value", "oil_pressure_value", "neutral_gear_value", "tachometer_value", "speedometer_value" y "fuel_value".

Posteriormente, se almacenan los datos en un archivo CSV llamado "Data_recorded.csv" utilizando la función "csv.writer()" y también se agregan a la matriz "data_received" para su posterior procesamiento. Finalmente, se agrega la matriz "data_received" a otra matriz llamada "data_matrix", que contiene todos los datos recibidos por la unidad receptora de la ECU. En resumen, este código se utiliza para recibir, procesar y almacenar los datos enviados por la ECU receptora y hacerlos disponibles para su posterior análisis y uso. Para la parte de almacenamiento de datos, se ha desarrollado en un archivo Python independiente de la GUI. Sin embargo, para futuras iteraciones de este trabajo, se recomienda integrar esta funcionalidad junto con el código de la interfaz gráfica de usuario (GUI).

```
if(serial_communication.is_open == True):
    RawString = serial_communication.readline()
    rawDataIn = RawString.decode("utf-8").strip()
    rawDataIn = rawDataIn.split(",")
    DataIn = int(rawDataIn[1])
    bin_num = bin(DataIn & int("1"*32, 2))[2:]

    battery_voltage_value = int(bin_num) & 0xFF
    oil_pressure_value = int(bin_num,2) >> 8 & 0b1
    neutral_gear_value = int(bin_num,2) >> 9 & 0b1
    tachometer_value = int(bin_num,2) >> 10 & 0xFF
    speedometer_value = int(bin_num,2) >> 18 & 0xFF
    fuel_value = int(bin_num,2) >> 26 & 0x7F

    with open('Data_recorded.csv', mode='w') as file:
        writer = csv.writer(file)
        writer.writerow(battery_voltage_value, oil_pressure_value, neutral_gear_value, tachometer_value, speedometer_value, fuel_value)
        data_received.append(battery_voltage_value, oil_pressure_value, neutral_gear_value, tachometer_value, speedometer_value, fuel_value)

    data_matrix.append(data_received)
```

58. GUI Telemetría – Tratamiento de los datos entrantes. Imagen de elaboración propia.

4. Coste del proyecto

Este proyecto se ha realizado en colaboración con la Universidad de Vic, siendo financiado el importe íntegro del material.

Teniendo en cuenta que tanto la recerca de información, el planteamiento, cálculos, estudio, diseño de los circuitos, diseño de PCBs, soldadura y ensamblaje de componentes, programación y testing han sido realizados de forma íntegra por mí a lo largo de innumerables horas, no se incluirán para el cómputo del coste de este presente proyecto. Por lo que sólo se tendrá en cuenta el coste de materiales empleados en el proyecto.

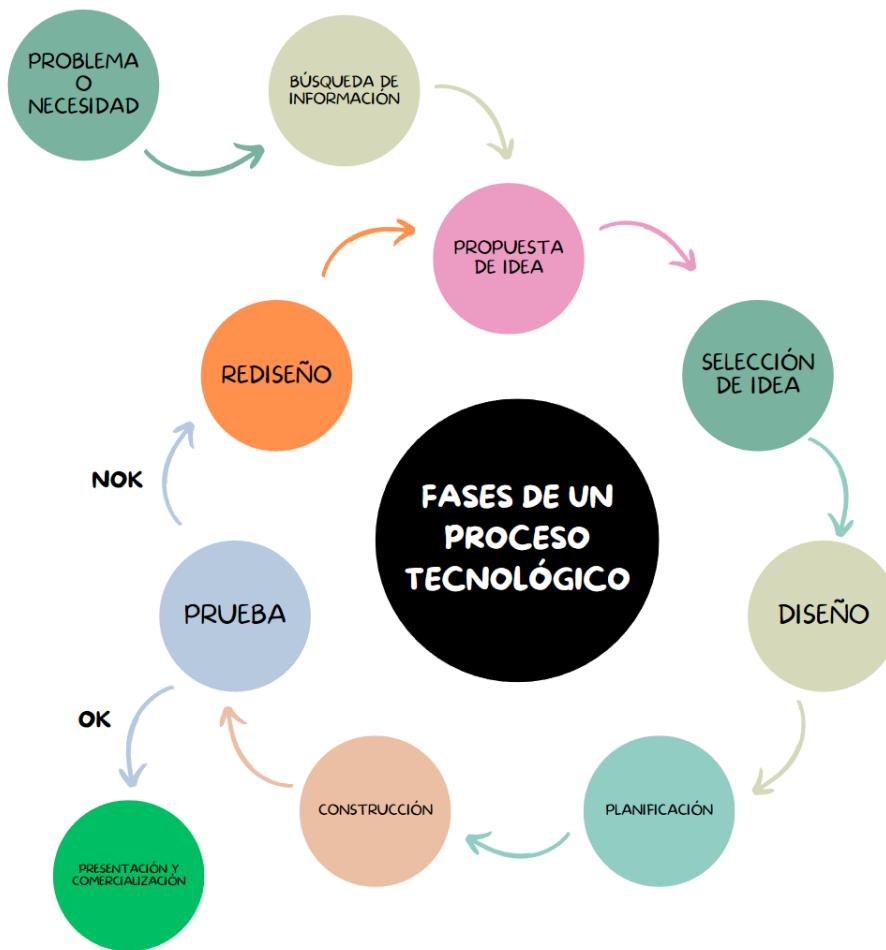
#	ÍTEM-REFERENCIA	DESCRIPCIÓN	CANTIDAD	PRECIO UNITARIO (I.V.A INCLUIDO)	SUB-TOTAL
1	MX34016SF1	Conecotor porta-hembra original Kawasaki	3	2.40 €	7.20 €
2	MX34016NF1	Conecotor macho original Kawasaki	3	2.57 €	7.71 €
3	M34S75C4F1	Pines hembra conector original Kawasaki	25	0.11 €	2.75 €
4	RYLR998	Módulo transceptor LoRa	2	15.86 €	31.72 €
5	158-P02EK381V8-E	Bloques terminales conectables para PCB con paso de 3.81 mm	3	2.80 €	8.40€
6	LM2673S-3.3/NOPB	Reguladores de tensión de conmutación 3A 3,3V	2	6.16 €	12.32 €
7	B360B-13F	Diodo Schottky 3A	4	0.44 €	1.76 €
8	SRP1038WA-100M	Inductor 10uH 8A	2	1.54 €	3.08 €
9	CP2102N	Bridge UART-USB	3	4.11 €	12.33 €
10	SP0503BAHTG	Diodo TVS supresor ESD	5	0.79 €	3.79 €
11	2174507-2	Conecotor USB B hembra para PCB	3	2.76 €	8.28 €
12	TC1017-3.3VLTR	Regulador lineal LDO 3.3V	2	0.52 €	1.04 €
13	C1206C395K8RACTU	Condensador cerámico 4 uF	5	0.83 €	4.15 €
14	Fabricación	PCB JLCPCB	1	70.30 €	70.30 €
15	Costes de envío	CORSA TECHNIC, MOUSER, FIRST COMPONENTS	-	81.46 €	81.46€
TOTAL					256.29 €

Es relevante destacar que los demás componentes empleados en este proyecto, tales como resistencias, diodos, condensadores, LEDs, cables dupont, cables unifilares y materiales consumibles como flux, estaño, pasta de estaño, malla desoldadora, entre otros, han sido suministrados a través del stock existente en el laboratorio de Can Muntanyola, perteneciente a la Universitat de Vic. En consecuencia, no ha sido necesario adquirirlos de forma expresa.

5. Resultados y conclusiones

Mi intención inicial de desarrollar un trabajo de final de grado en el que poner en práctica los conocimientos adquiridos en las distintas asignaturas cursadas a lo largo del grado en Ingeniería de la Automoción se ha cumplido con creces. Este proyecto me ha permitido explorar y aprender un poco más sobre electrónica, programación y las comunicaciones por radiofrecuencia. Me ha parecido un proyecto muy interesante y con buena base para poder seguir desarrollándose con mejoras e innovaciones.

A pesar de algunos errores de diseño que se han detallado en el Anexo “7.3 Limitaciones y Mejoras para futuros proyectos”, me siento muy satisfecho con el rendimiento del sistema de telemetría desarrollado en este proyecto. Es importante tener en cuenta que el proceso de desarrollo de un producto tecnológico consta de diversas fases, muchas de las cuales se repiten en bucle hasta obtener el producto final.

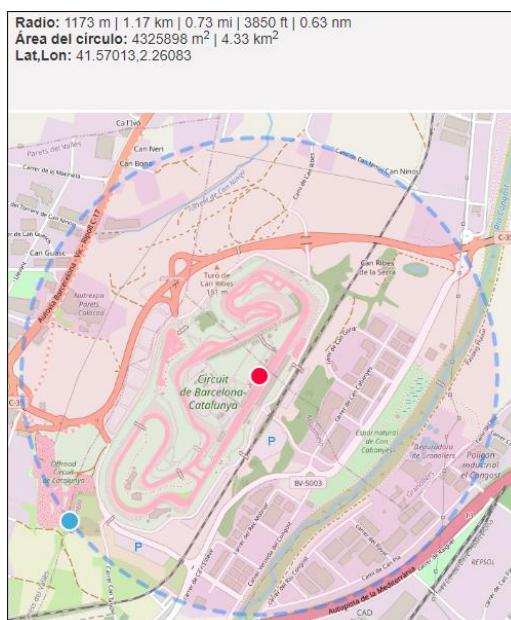


59. Esquema de fases de un proceso tecnológico. Imagen de elaboración propia.

A pesar de las limitaciones de tiempo y el nivel de ambición del proyecto de fin de grado, se realizaron un par de iteraciones en el bucle de desarrollo, lo que reveló puntos débiles y de mejora del actual diseño. Aun así, considero que este trabajo es muy completo, permitiendo la aplicación de numerosos conocimientos de ingeniería de la automoción adquiridos a lo largo del grado. Aunque hay aspectos que requieren mejoras, la fase actual de este desarrollo proporciona una base sólida para futuros proyectos o investigaciones en telemetría. La experiencia adquirida al desarrollar este proyecto me ha brindado lecciones valiosas y una comprensión profunda de la importancia de la iteración y el proceso de mejora continua en el desarrollo de soluciones tecnológicas.

Como se puede observar en el apartado de pruebas (Anexo “7.2. Pruebas”), se ha logrado obtener una latencia máxima de 19 milisegundos y alcanzar una distancia de hasta 1’17 km. Esta distancia es más que suficiente para adquirir datos de un monoplaza, como por ejemplo de la competición Formula Student, incluso en un escenario como el circuito donde se celebra el gran premio de España de Formula 1 ubicado en Montmeló (Barcelona).

Estos resultados validan la efectividad y la capacidad de este sistema de comunicación inalámbrica para transmitir datos de manera confiable, incluso en entornos de alta exigencia. Además, se evidencia que la latencia y distancia alcanzada cumplen con los requisitos necesarios para adquirir y procesar datos en tiempo real, lo que sería crucial en aplicaciones como la competición Formula Student.

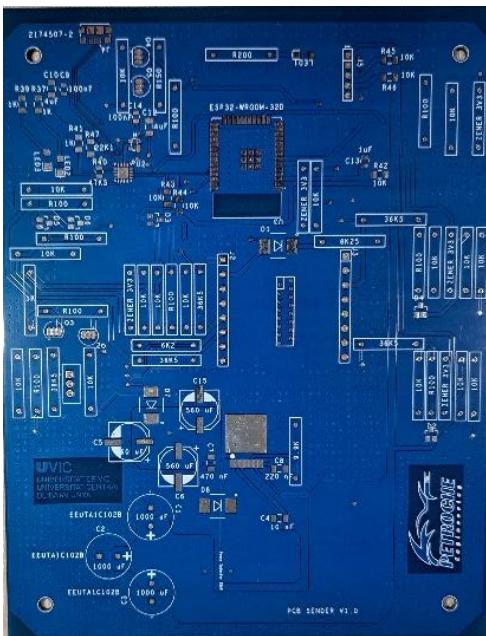


60. Radio de comunicación asegurada de este sistema de telemetría. Imagen de elaboración propia.

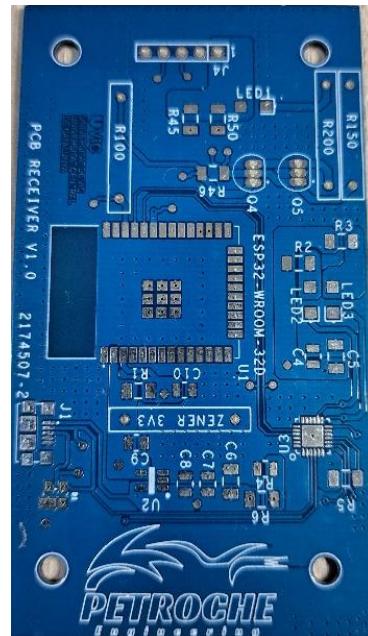
La tecnología inalámbrica por radiofrecuencia LoRa se presenta como una solución altamente efectiva y de fácil implementación en el campo de la telemetría automotriz. A lo largo de este trabajo, se ha demostrado que la implementación de la comunicación mediante tecnología LoRa es una opción potente y económica, como se ha expuesto en el apartado sobre el coste del proyecto.

Pese a que esta primera versión del sistema de telemetría se realiza para un vehículo específico, es relativamente fácil poder adaptar el sistema entero a cualquier otro vehículo. Incluso, añadiendo un módulo de transceptor CAN, es posible poder adquirir los datos de una red de comunicación CAN BUS de cualquier vehículo.

Me gustaría animar a todos los lectores de este trabajo a continuar con el desarrollo de este proyecto, ya que estoy seguro de que disfrutarán tanto o más que yo. Ha sido una experiencia emocionante investigar y diseñar cada aspecto de este sistema de telemetría, y espero que esta investigación sirva de inspiración y motivación para futuros trabajos en este campo.



61. PCB ECU Transmisora V1.0.



62. PCB ECU Receptora V1.0.
Imagen de elaboración propia.

6. Bibliografía

- [1] Semtech Corporation. (2020). “LoRa Fills a Technology Gap”. En Semtech. Recuperado de: <https://www.semtech.com/lora/why-lora>
- [2] Kawasaki Heavy Industries, LTD. (2012). “Sistema Eléctrico: Diagrama del cableado”. En Manual de taller oficial de motocicleta Kawasaki Z800.
- [3] Espressif Systems Co., LTD. (2016). “FreeRTOS”. En Espressif. Recuperado de: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/freertos.html>
- [4] Reyax Technology Co., LTD. (2017). “RYLR998”. En Reyax. Recuperado de: <https://reyax.com/products/RYLR998>
- [5] Nexperia, LTD. (2022) .”Datasheet del componente BAT54S”. En Mouser. Recuperado de: <https://www.mouser.es/datasheet/2/916/BAT54S-3081480.pdf>.
- [6] Paul Horowitz and Windfield Hill. (2015). “The art of electronics”. New York, NY 10013-247: Cambridge University Press.
- [7] D. A. Hodges. (1999). “Darlington's Contributions to Transistor Circuit Design”. Berkeley, University of California.
- [8] S. Darlington. (1952). “Semiconductor signal translating device”. US 2663806 Patente. Washington, DC: Oficina de Patentes y Marcas de Estados Unidos.
- [9] Rainer Knäpper. (2007). “Streifenrasterleiterplatte”. En Wikipedia. Recuperado de: https://commons.wikimedia.org/wiki/File:Streifenrasterleiterplatte_IMGP5364.jpg
- [10] Sonic Potions. (2013). “Frontpanel PCB”. En Wikipedia. Recuperado de: https://commons.wikimedia.org/wiki/File:Sonic_Potions_LXR_frontpanel_PCB_2.jpg
- [11] Reyax Technology Co., LTD. (2017). “RYLR998 Datasheet: Block diagram”. En Reyax. Recuperado de: https://reyax.com//upload/products_download/download_file/RYLR998_EN.pdf
- [12] Reyax Technology Co., LTD. (2017). “LoRa AT Command RYLR998-RYLR498”. En Reyax. Recuperado de: https://reyax.com//upload/products_download/download_file/LoRa_AT_Command_RYL998_RYLR498_EN.pdf
- [13] CBurnett. (2007). “Circular buffer”. En Wikipedia. Recuperado de: https://es.m.wikipedia.org/wiki/Archivo:Circular_buffer.svg
- [14] Tom Schimansky. (2022). “CustomTkinter Documentation and Tutorial”. En página oficial. Recuperado de: <https://customtkinter.tomschimansky.com/>

- [15] The Matplotlib development team. (2022). “Matplotlib Documentation and Tutorial”. En página oficial. Recuperado de: <https://matplotlib.org/stable/tutorials/index>
- [16] Travis Oliphant. (1995). “Numpy Documentation and Tutorial”. En página oficial. Recuperado de: <https://numpy.org/>
- [17] Roger Meier. (2015). “PySerial’s Documentation”. En página oficial. Recuperado de: <https://pythonhosted.org/pyserial/>
- [18] Semtech Corporation. (2015). “Network Trial”. En Semtech. Recuperado de: <https://www.frugalprototype.com/wp-content/uploads/2016/08/an1200.22.pdf>
- [19] Cyril Buttay. (2006). “Buck conventions”. En Wikipedia. Recuperado de: https://es.wikipedia.org/wiki/Archivo:Buck_conventions.svg
- [20] Achim Pieters. (2020). “ESP-WROOM-32 CHIP PINOUT”. En studiopieters. Recuperado de: <https://www.studiopieters.nl/esp32-pinout/>
- [21] Espressif Systems Co., LTD. (2017). “ESP32_DevKitc_V4”. En Espressif. Recuperado de: https://dl.espressif.com/dl/schematics/esp32_devkitc_v4-sch.pdf
- [22] Texas Instruments. (2000). “LM1117 Low-Dropout Linear Regulator Datasheet”. En Texas Instruments. Recuperado de: https://www.ti.com/lit/ds/symlink/lm1117.pdf?HQS=dis-mous-null-mousermode-dsf-pf-null-wwe&ts=1685952046895&ref_url=https%253A%252F%252Fwww.mouser.es%252F

7. Anexos

7.1. Diseño mecánico

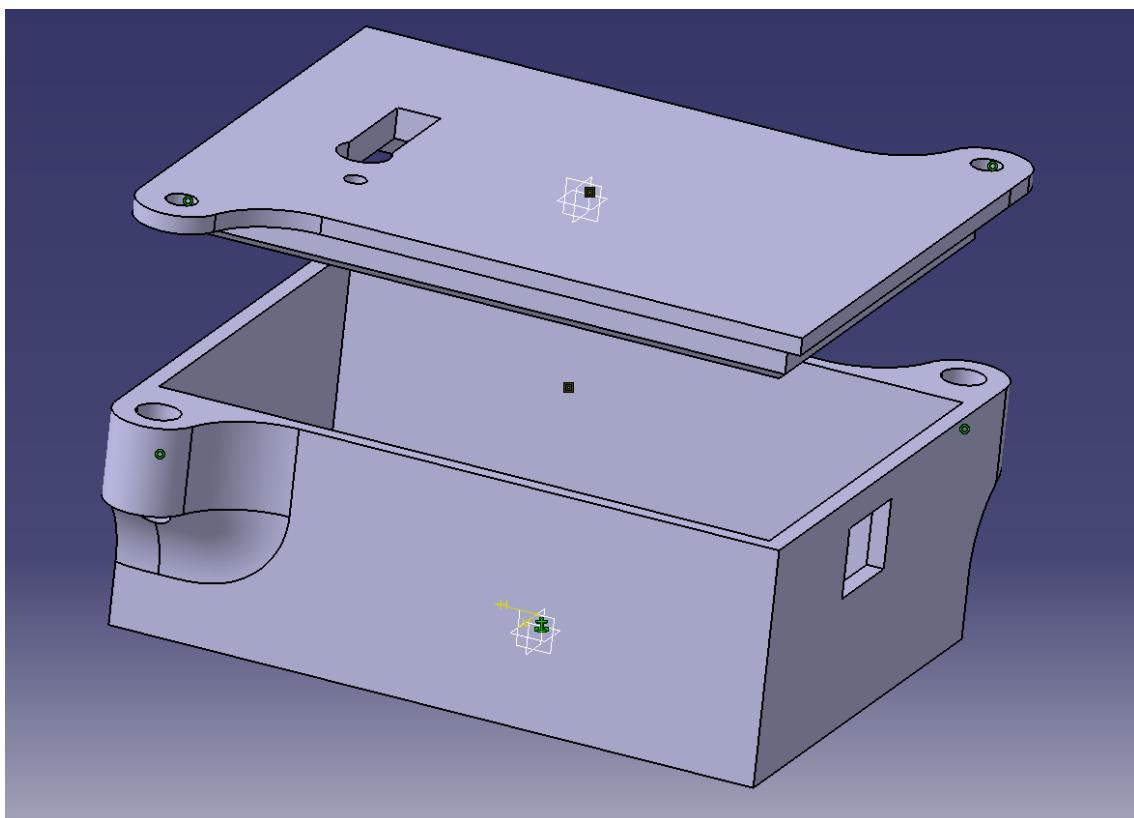
Para este proyecto se ha realizado el diseño mecánico de la carcasa de la unidad de control receptora. Por falta de tiempo, no se ha podido realizar también el diseño de la carcasa para la ECU transmisora integrada en la moto.



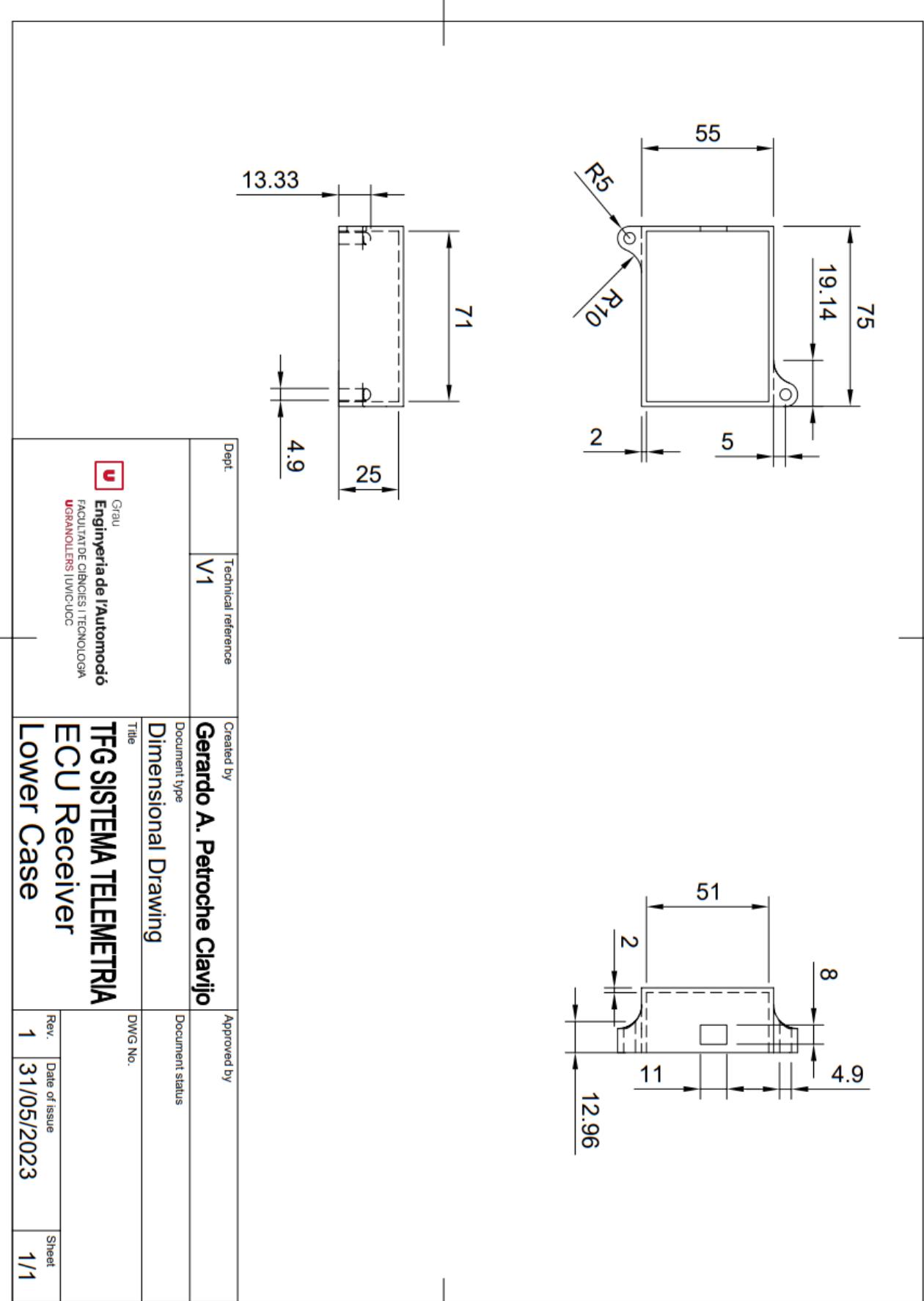
63. ECU Transmisora integrada en la motocicleta Kawasaki Z800. Imagen de elaboración propia.

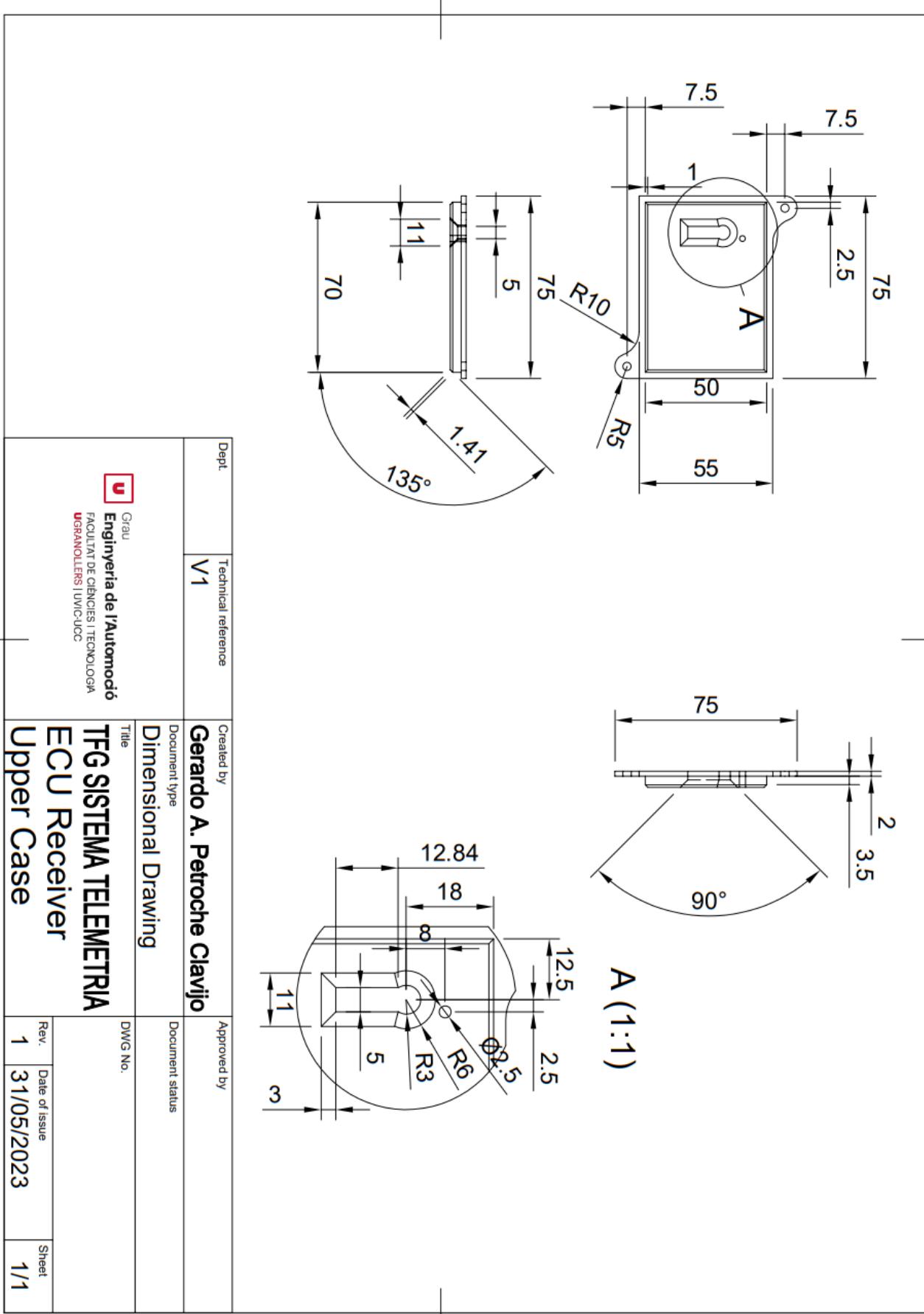
La carcasa no solo ofrece un aspecto estético mejorado, sino que también proporciona una protección adicional contra posibles cortocircuitos. Si alguna superficie conductora eléctrica entra en contacto con la unidad electrónica receptora, no se producirá ningún cortocircuito gracias a la presencia de la carcasa.

Aunque esta carcasa ofrecerá protección contra objetos no es efectiva ante líquidos conductores, pues no es estanca debido a que necesita un orificio para poder conectar el USB para tener comunicación con el ordenador, así como un orificio adicional para permitir el paso del haz de luz LED indicador de estado.



64. Ensamblaje del diseño mecánico de la carcasa para la unidad de control receptora. Imagen de elaboración propia.

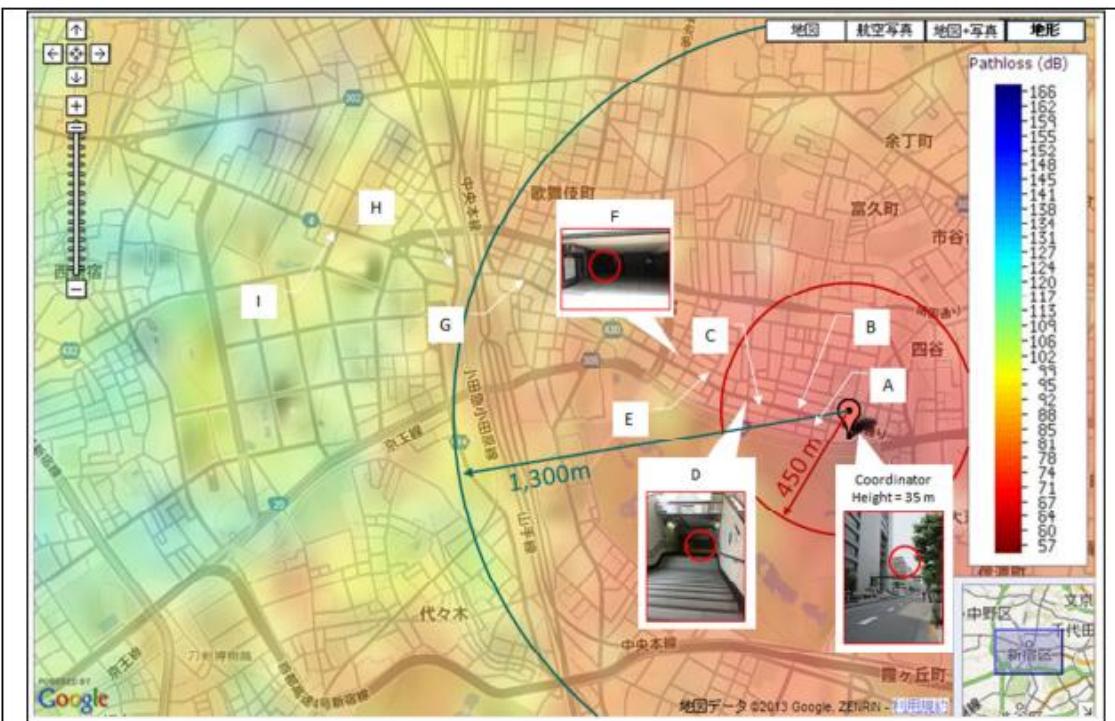




7.2. Pruebas

7.2.1. Distancia máxima de funcionamiento

Tomando como referencia la prueba realizada por la empresa Semtech (empresa que posee la patente de LoRa) y evidenciado en el documento “AN1200.22 LoRa™ Modulation Basics” [18], donde se realizaba la comparativa del desempeño de la tecnología LoRa respecto a una tecnología de modulación por desplazamiento de frecuencia (FSK) en la zona de Shinjuku (Tokyo Metropolis), Japón.



Ref. #	Distance (m)	2-FSK: 4.8 kb/s		LoRa: 125 kHz BW, SF = 8 (3.125 kb/s)	
		Rssi (dBm)	PER (%)	Rssi (dBm)	PER (%)
A	80	-97	0	-91	0
B	150	-100	0	-102	0
C	280	-112	1	-114	0
D	330	-	100	-124	10
E	480	-118	8	-120	0
F	560	-	100	-121	0
G	1180	-	100	-112	0
H	1350	-	100	-126	10
I	1750	-	100	-127	100

65. Prueba realizada por Semtech. Fuente: Semtech [18].

De todas las pruebas realizadas, considero que esta es la más relevante para escoger esta tecnología (LoRa) para aplicaciones en telemetría. La prueba se llevó a cabo utilizando ambos módulos con los parámetros de configuración descritos en el apartado “3.2.3. Comunicación con el módulo RYLR998 y comandos AT”. Estos parámetros fueron configurados específicamente para maximizar la potencia de transmisión, y por tanto la máxima distancia de comunicación posible.

Para la prueba, se utilizó una función implementada en el modo debug, la cual envía valores diferentes en cada variable de datos, las cuales van variando de forma incremental. De esta manera se podría detectar posibles fallos o bloqueos en alguna de las dos unidades de control electrónicas.

```
void TaskDataSampling(void *pvParameters)
{
    while (1)
    {
        if (getKeyStatus() == 1)
        {
            if (DEBUG_MODE == 1)
            {
                contador++;
                contador_fuel++;
                contador_binario=0;

                adc_battery_value = contador+1;
                adc_fuel_value = contador_fuel;
                NeutralGear_value = contador_binario;
                oilPressure_value = !contador_binario;
                //Key_value = 1;
                Speedometer_value = contador+2;
                Tachometer_value = contador+3;

                if (contador == 250)
                {
                    contador = 0;
                }
                else
                {
                    //do nothing
                }

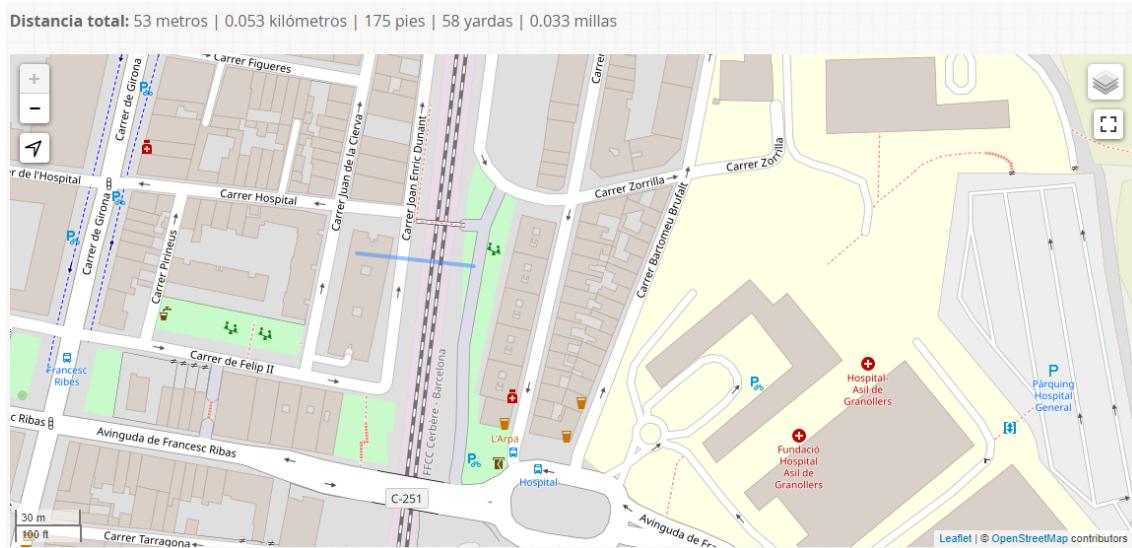
                if (contador_fuel == 100)
                {
                    contador = 0;
                }
                else
                {
                    //do nothing
                }

                if (contador_binario == 1)
                {
                    contador = 0;
                }
                else
                {
                    //do nothing
                }
            }
        }
    }
}
```

66. Función usada para crear valores distintos de forma incremental en cada variable. Imagen de elaboración propia.

Esta prueba se llevó a cabo en un entorno urbano, específicamente en la ciudad de Granollers. Se realizó incrementando progresivamente la distancia entre las unidades de control evaluando si aún había comunicación. Se realizaron 4 pruebas de comunicación a distintas distancias para evaluar la comunicación entre los módulos. La primera prueba se realizó con una distancia entre los módulos de 53 metros, la segunda a 201 metros, la tercera a 499 metros y la última a 1173 metros. En esta última medición, se observó que la unidad de control receptora sólo recibía datos cuando la antena LoRa estaba ubicada en determinadas posiciones.

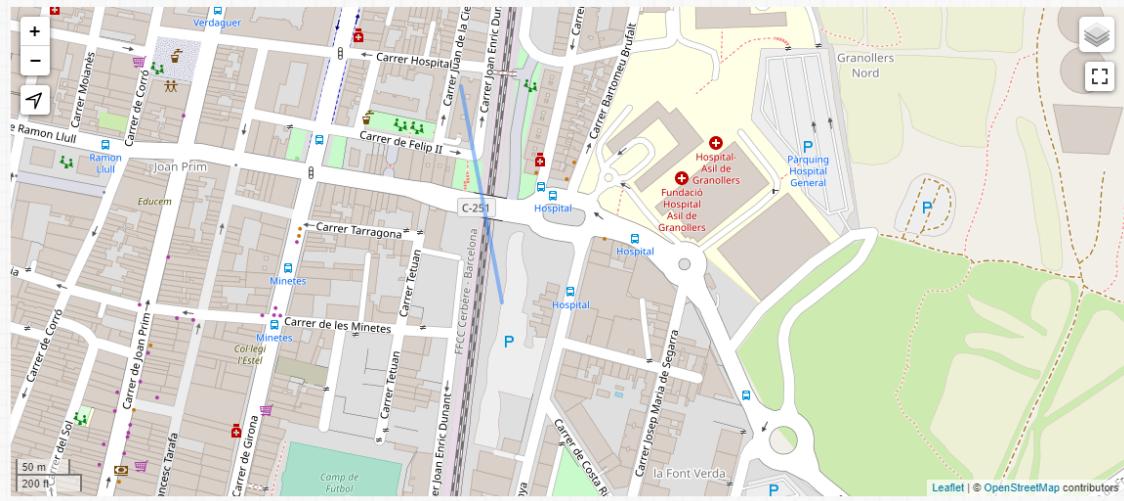
Basándonos en los resultados obtenidos, se determinó que la distancia límite de comunicación en un entorno urbano es de 1'17 kilómetros. Esto significa que, más allá de esta distancia, la transmisión de datos se vuelve poco confiable.



67. Distancia máxima de funcionamiento del sistema de telemetría – Prueba 1: 53 metros.

Imagen de elaboración propia.

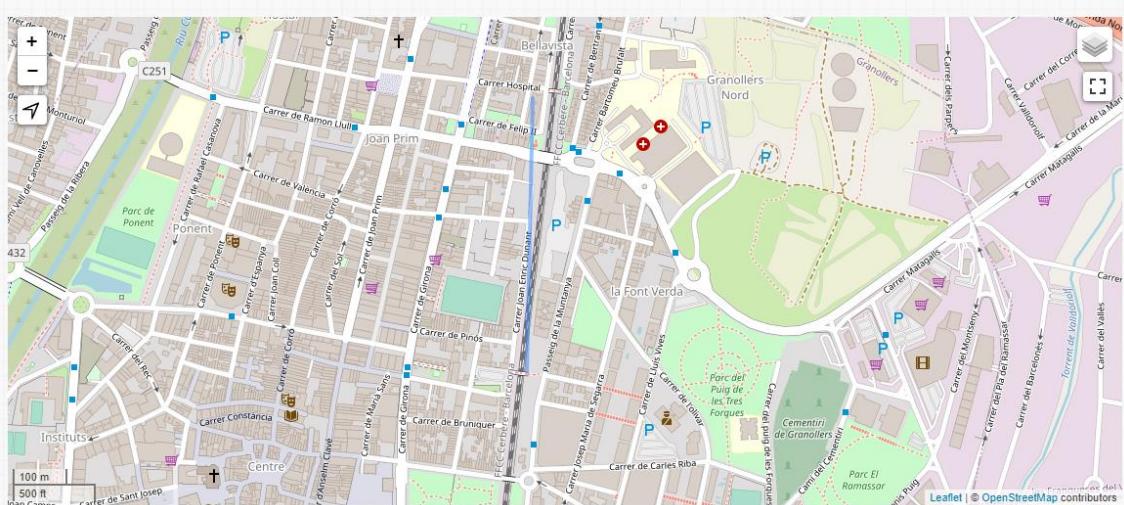
Distancia total: 201 metros | 0.201 kilómetros | 658 pies | 219 yardas | 0.125 millas



68. Distancia máxima de funcionamiento del sistema de telemetría – Prueba 2: 201 metros.

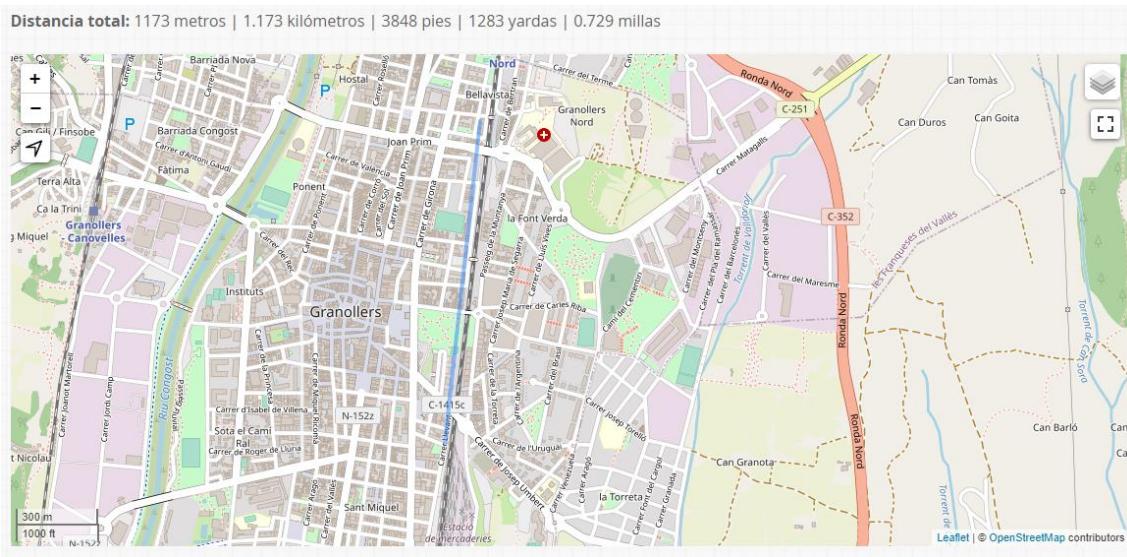
Imagen de elaboración propia.

Distancia total: 499 metros | 0.499 kilómetros | 1636 pies | 545 yardas | 0.310 millas



69. Distancia máxima de funcionamiento del sistema de telemetría – Prueba 3: 499 metros.

Imagen de elaboración propia.



70. Distancia máxima de funcionamiento del sistema de telemetría – Prueba 4: 1173 metros.

Imagen de elaboración propia.

Comparando los resultados del documento "AN1200.22 LoRaTM Modulation Basics" de Semtech con los resultados que he obtenido personalmente, se puede concluir que ambos reflejan resultados similares y están en concordancia.

En el documento de Semtech, se realizaron pruebas con los módulos LoRa configurados con parámetros destinados a lograr la mayor distancia de comunicación posible. Estos parámetros incluyen un ancho de banda de 125 kHz y un factor de dispersión de 8. Utilizando esta configuración, se logró una distancia máxima de funcionamiento entre 1350 y 1750 metros.

En mi propia prueba, decidí configurar los módulos para obtener la mayor distancia posible manteniendo una baja latencia y considerando la cantidad de datos a enviar. Para lograr esto, utilicé un ancho de banda de 500 kHz y un factor de dispersión de 7. Los resultados mostraron que, a pesar de no comprometer significativamente la distancia máxima de comunicación, pude lograr una menor latencia en comparación con la configuración de Semtech.

Estos hallazgos respaldan la validez de los resultados presentados en el documento de Semtech y también demuestran que es posible ajustar los parámetros de configuración de los módulos LoRa para adaptarse a diferentes requisitos y objetivos de comunicación.

7.2.2. Latencia entre módulos

Para realizar esta prueba, se realizará una transmisión de datos generados aleatoriamente y se enviarán del módulo transmisor al módulo receptor. Mediante el monitor serial se registrará la hora, minuto y segundo tanto de la transmisión como de la recepción de los datos. La latencia entre los módulos se establece por el tiempo entre el dato enviado y el primer dato que se reciba. Esto se realizará diversas veces con tal de obtener un promedio de este valor.

Monitor Mode	Serial	Port	COM6 - Silicon Labs CP210x USB to UART Bridge (COM6)	Baud rate
19:27:48:634	->	110110111101101111100110111100	922155452	
19:27:48:638	->	111010111010111110110111101	989527485	
19:27:48:642	->	1111101111111000000110111110	1056899518	
19:27:48:646	->	100001100000011000001011011111	1124271551	
19:27:48:650	->	10001110000111000010011100000	1191643584	
19:27:48:654	->	1001011000010110000110111000001	1259015617	
19:27:48:658	->	1001111000011100001000111000010	1326387650	
19:27:48:662	->	10100110001001100010111000011	1393759683	
19:27:48:666	->	101011100010111000100111000100	1461131716	
19:27:48:670	->	101101100011011000110111000101	1528503749	
19:27:48:674	->	1011110001111000110000111000110	1595875782	
19:27:48:678	->	1100001100100010010010111000111	1663247815	
19:27:48:682	->	11000110010011001010011001000	1730619848	
19:27:48:686	->	1100101100101011001011011001001	1797991881	
19:27:48:689	->	1101111001011110011000111001010	1865363914	
19:27:48:693	->	11100011001100010011001011001011	1932735947	
19:27:48:698	->	111001100110110011000111001100	20000107980	
19:27:48:701	->	111101100110110011011001101	2067480013	
19:27:48:705	->	111111100111110100000111001110	2134852046	
19:27:48:709	->	10000011010000110100010111001111	-2092743217	
19:27:48:714	->	10000111010001110100100111010000	-2025371184	
19:27:48:718	->	100001010100100101001011010001	-1957999151	
19:27:48:722	->	100001110100111010100111010010	-1890627118	
19:27:48:726	->	1001001101010001010101011010011	-1823255085	
19:27:48:730	->	10010111010101101010101100110100	-1755883052	
19:27:48:734	->	1001101101010110101010110010101	-1688511019	
19:27:48:739	->	10011110101111010000110101010	-1621138986	
19:27:48:743	->	10100011010000101000101101010111	-1553766953	
19:27:48:747	->	101001110100011010010011101000	-1486394920	
19:27:48:751	->	101010110101010101010110101001	-1419022887	
19:27:48:755	->	1010111010101101010101100101010	-1351650054	
19:27:48:759	->	101010010101000101010101011010111	-1284278821	
19:27:48:764	->	101010110101010101010101011010100	-1216986788	
19:27:48:768	->	10111010101110101011110101101101	-1149534755	
19:27:48:772	->	1011111010111110000001101011100	-1082162722	
19:27:48:782	->	11000011000000110000010111011111	-1014790689	
19:27:48:782	->	11000011100000110000010011100000	-947418656	
19:27:48:785	->	110010110000101000010111000001	-880046623	
19:27:48:788	->	11001111000011110001000111000010	-812674590	
19:27:48:793	->	11010011000100110001010101100011	-745302557	
19:27:48:797	->	11010111000101110001000111000000	-677930524	
19:27:48:801	->	11010101000101000101011000101	-610558491	
19:27:48:805	->	110111110001111001000011100010	-543186458	
19:27:48:809	->	111000110001000100010010111000111	-475814425	
19:27:48:813	->	11100111001001110010010111000000	-408442392	
19:27:48:817	->	1110101100101010101010101001	-341070359	
19:27:48:821	->	111011110101111001000111001010	-273698326	
19:27:48:825	->	SLEEP MACHINE STATE		
19:27:48:827	->			

71. Prueba de latencia entre módulos – Datos enviados por ECU transmisora. Imagen de elaboración propia.

```
Monitor Mode Serial Port COM9 - Silicon Labs CP210x USB to UART Bridge (COM9) Baud rate 115200
19:27:48:761 -> ,-1890627118,-
19:27:48:761 -> ,-1890627118,-
19:27:48:761 -> ,-1890627118,-
19:27:48:761 -> ,-1890627118,-
19:27:48:761 -> ,-1890627118,-
19:27:48:761 -> ,-1890627118,-
19:27:48:761 -> ,-1890627118,-
19:27:48:761 -> ,-1890627118,-
19:27:48:763 -> ,-1890627118,-
19:27:48:764 -> ,-1553766953,-
19:27:48:766 -> ,-1553766953,-
19:27:48:767 -> ,-1553766953,-
19:27:48:769 -> ,-1553766953,-
19:27:48:770 -> ,-1553766953,-
19:27:48:772 -> ,-1553766953,-
19:27:48:773 -> ,-1553766953,-
19:27:48:782 -> ,-1553766953,-
19:27:48:782 -> ,-1553766953,-
19:27:48:784 -> ,-1553766953,-
19:27:48:785 -> ,-1553766953,-
19:27:48:786 -> ,-1149534755,-
19:27:48:788 -> ,-1149534755,-
19:27:48:789 -> ,-1149534755,-
19:27:48:791 -> ,-1149534755,-
19:27:48:792 -> ,-1149534755,-
19:27:48:794 -> ,-1149534755,-
19:27:48:795 -> ,-1149534755,-
19:27:48:800 -> ,-1149534755,-
19:27:48:801 -> ,-1149534755,-
19:27:48:804 -> ,-1149534755,-
19:27:48:805 -> ,-1149534755,-
19:27:48:807 -> ,-812674590,-->
19:27:48:808 -> ,-812674590,-7
19:27:48:810 -> ,-812674590,-7
19:27:48:811 -> ,-812674590,-7
19:27:48:813 -> ,-812674590,-7
19:27:48:814 -> ,-812674590,-7
19:27:48:816 -> ,-812674590,-7
19:27:48:822 -> ,-812674590,-7
19:27:48:824 -> ,-812674590,-7
19:27:48:826 -> ,-812674590,-7
19:27:48:827 -> ,-812674590,-7
19:27:48:829 -> ,-475814425,-->
19:27:48:830 -> ,-475814425,-6
19:27:48:832 -> ,-475814425,-6
19:27:48:833 -> ,-475814425,-6
19:27:48:835 -> ,-475814425,-6
19:27:48:836 -> ,-475814425,-6
19:27:48:838 -> ,-475814425,-6
```

72. Prueba de latencia entre módulos – Datos recibidos por ECU receptora. Imagen de elaboración propia.

Por ejemplo, el mensaje que contiene el dato “-475814425” se envió desde la ECU transmisora a las 19:27:48:809 (formato hh:mm:ss:ms) y se recibió en la otra unidad de control a las 19:27:48:829 (formato hh:mm:ss:ms) obteniendo una latencia de comunicación entre módulo de 20ms.

La latencia promediada obtenida entre los dos módulos en esta prueba es de 19 milisegundos.

7.2.3. Overflow de variables

Codificar los datos en una misma variable mediante el desplazamiento de bits es una estrategia muy eficiente en términos de uso de memoria en soluciones embebidas, pero también comporta algún riesgo. Uno de los principales inconvenientes se produce cuando no se limitan correctamente el tamaño o posición de las variables y, consecuentemente, afectando al resto de bits. Esto puede resultar en valores incorrectos durante la decodificación, lo cual puede o no ser detectado.

Con el fin de evitar este problema (comúnmente denominado “*overflow de variables*”), se han realizado pruebas exhaustivas con el tamaño máximo de las variables y sus valores máximos correspondientes. Al realizar estas pruebas, se garantiza que se tenga un conocimiento claro de los valores máximos que las variables pueden alcanzar, evitando así problemas potenciales durante la decodificación.

Los datos codificados en una misma variable incluyen la tensión de la batería, el porcentaje de combustible, la marcha neutral engranada, el manocontacto de presión de aceite, la velocidad actual y las revoluciones por minuto del motor. Para la primera prueba, hemos configurado en el código el valor máximo de cada una de estas variables.

Hemos establecido los valores máximos correspondientes a cada variable para garantizar que no se excedan sus rangos. La tensión de la batería tiene un máximo de 160, el porcentaje de combustible puede alcanzar hasta el 100, la marcha neutral está representada por el valor 1, la presión de aceite también es 1, el velocímetro tiene un valor máximo de 255 y el tacómetro puede llegar hasta 150. De esta manera, hemos asegurado que las variables estén configuradas correctamente para la prueba inicial, teniendo en cuenta sus límites máximos.

Dato	Valor mínimo	Valor máximo	Tamaño de la variable [decimal]
Tensión de batería	0	160	255
Nivel de combustible	0	100	127
Marcha neutral	0	1	1
Presión de aceite	0	1	1
Velocímetro	0	255	255
Tacómetro	0	150	255

Como resultado de esta prueba no obtuvimos ningún problema, los datos se enviaron y se decodificaron correctamente sin ningún error en ellos.

Con el fin de verificar que no haya fallos en ningún valor o combinación de valores dentro de los rangos posibles, se llevó a cabo una prueba exhaustiva. Esta prueba consistió en enviar de forma incremental todos los números desde el valor mínimo hasta el máximo para cada una de las variables. Se implementó este proceso de prueba para todas las variables simultáneamente, con un desfase (offset) entre ellas. De esta manera, en caso de que se produzca algún error, cada variable tendría un valor distinto en cada instante preciso de la prueba.

Esta metodología de prueba exhaustiva permitió evaluar el comportamiento de las variables en diferentes situaciones y detectar posibles fallas o comportamientos inesperados. Al enviar gradualmente los valores y cubrir todo el rango permitido, se aseguró que las variables fueran sometidas a diversas condiciones y se verificó su correcto funcionamiento en cada una de ellas.



```
void TaskDataSampling(void *pvParameters)
{
    while (1)
    {
        if (getKeyStatus() == 1)
        {
            if (DEBUG_MODE == 1)
            {
                contador++;
                contador_fuel++;
                contador_binario=0;

                adc_battery_value = contador+1;
                adc_fuel_value = contador_fuel;
                NeutralGear_value = contador_binario;
                OilPressure_value = !contador_binario;
                //Key_value = 1;
                Speedometer_value = contador+2;
                Tachometer_value = contador+3;

                if (contador == 250)
                {
                    contador = 0;
                }
                else
                {
                    //do nothing
                }

                if (contador_fuel == 100)
                {
                    contador = 0;
                }
                else
                {
                    //do nothing
                }

                if (contador_binario == 1)
                {
                    contador = 0;
                }
                else
                {
                    //do nothing
                }
            }
        }
    }
}
```

73. Código Fuente de la función utilizada para verificar el comportamiento de las variables según los valores de los datos codificados. Imagen de elaboración propia.

Después de observar el comportamiento de los valores decodificados recibidos por la unidad de control receptora, se pudo constatar que todos ellos respondían de manera correcta y presentaban un comportamiento consistente. Durante la prueba, se pudo apreciar que el valor de los datos transmitidos aumentaba gradualmente, y una vez alcanzado el valor máximo permitido, se reiniciaba a cero y volvía a comenzar el bucle.

Este patrón de incremento continuo seguido de un reinicio del valor en el límite máximo permitido reflejaba el funcionamiento esperado de las variables. Este comportamiento indicaba que las variables estaban configuradas correctamente y que el sistema era capaz de manejar los valores dentro de sus rangos establecidos.

La consistencia en el comportamiento de las variables transmitidas durante la prueba brindó confianza en su correcto funcionamiento y validó que el sistema estaba diseñado para operar de manera estable y predecible.

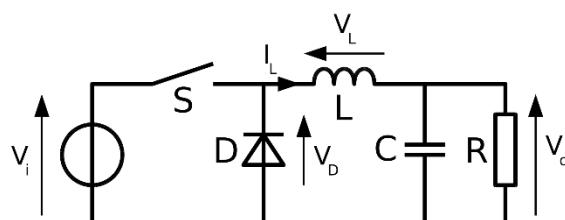
En la siguiente figura, se observa la lectura del monitor serial. En el recuadro rojo se muestra la tensión de batería, en el verde está el tacómetro, en el amarillo está el velocímetro, en el morado se encuentra el nivel de combustible y entre la tensión de batería y el tacómetro están los bits de la presión de aceite y la marcha neutral (derecha e izquierda respectivamente).

19:27:48:693 ->	11100011001100110011010111001011	1932735947
19:27:48:698 ->	111011100110111001110011001100	2000107980
19:27:48:701 ->	111010110011101100111100111001101	2067480013
19:27:48:705 ->	1111111001111110100000111001110	2134852046
19:27:48:709 ->	10000011010000110100010111001111	-2092743217
19:27:48:714 ->	1000011101000111010010111010000	-2025371184
19:27:48:718 ->	100010110100101101001101000111010001	-1957999151
19:27:48:722 ->	100011110100111010100011010010	-1890627118
19:27:48:726 ->	1001001101010011010101011010011	-1823255085
19:27:48:730 ->	1001011101010110101010011010100	-1755883052
19:27:48:734 ->	10011011010110110101101011010101	-1688511019
19:27:48:739 ->	100111110101111010000111010110	-1621138986
19:27:48:743 ->	1010001101100011011001011010111	-1553766953
19:27:48:747 ->	1010011101100110110100111011000	-1486394920
19:27:48:751 ->	1010101101101101101101101101001	-1419022887
19:27:48:755 ->	101011110110111011011000111011010	-1351650854
19:27:48:759 ->	101010011011000110110101011011011	-1284278821
19:27:48:764 ->	1010101101101101101110011011100	-1216906788
19:27:48:768 ->	101110110111101101111011011101	-1149534755
19:27:48:772 ->	10111111011111101100000111011110	-1082162722
19:27:48:782 ->	11000011100000111000010111011111	-1014790689
19:27:48:782 ->	1100011110000111000100111100000	-947418656
19:27:48:785 ->	1100101110001011100011011100001	-880046623
19:27:48:788 ->	1100111110001111001000111100010	-812674590
19:27:48:793 ->	11010011100010011100101011100011	-745302557
19:27:48:797 ->	110101110001011100100111000100	-677930524
19:27:48:801 ->	1101101110001001110011101100101	-610558491
19:27:48:805 ->	11011111000111101000111100110	-543186458
19:27:48:809 ->	11100011100011101001110100111100111	-475814425
19:27:48:813 ->	11100111101001111010100111101000	-408442392
19:27:48:817 ->	111010111010101101101011011101001	-341070359

74. Lectura monitor serial durante la prueba de desbordamiento de variables. Imagen de elaboración propia.

7.2.4. Análisis del convertidor de potencia de la unidad de control transmisora.

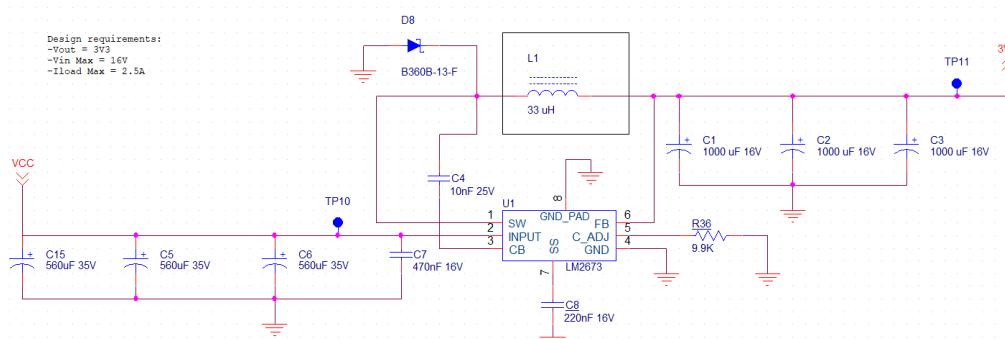
Con el fin de no consumir demasiada corriente y evitar así descargar la batería en exceso, la unidad de control transmisora integrada en el vehículo cuenta con su propio convertidor de potencia de alta eficiencia para alimentar al microcontrolador, módulo LoRa y periféricos. Se trata de un conversor Buck o “Step-down” converter, en específico el circuito integrado regulador de voltaje de commutación LM2673 del fabricante Texas Instruments.



75. Esquema básico de un convertidor reductor Buck. Fuente: Wikipedia [19].

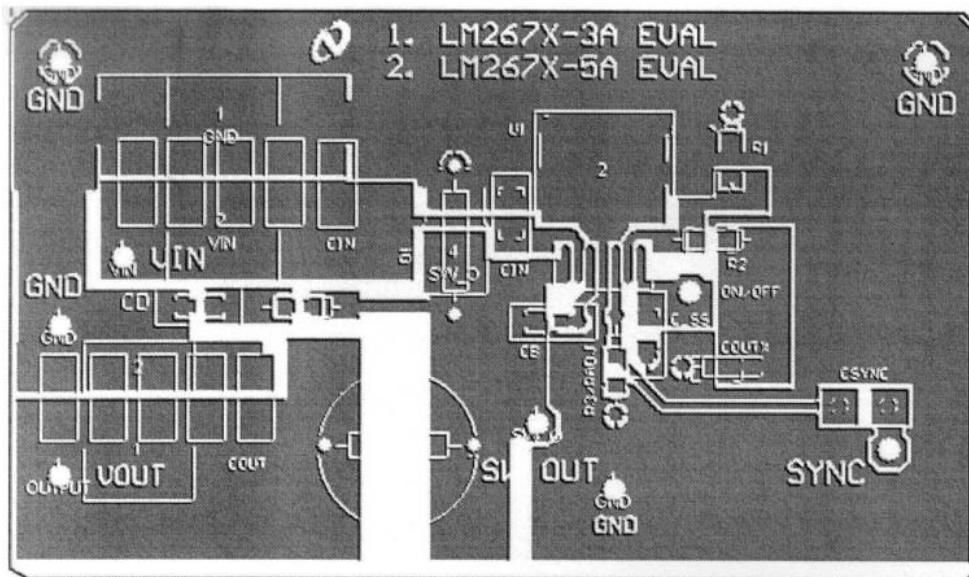
Para este sistema electrónico, se ha seleccionado la variante LM2673S-3.3 del regulador de voltaje LM2673. Esta variante proporciona una tensión de salida fija de 3'3 V y puede suministrar una corriente constante de hasta 2'5 A. Es importante tener en cuenta que estas especificaciones se cumplen dentro de un rango de voltaje de entrada de 8 a 40 V. Sin embargo, para lograr la máxima eficiencia, se recomienda una tensión de entrada entre 13 y 16 V (para más detalles del integrado, ver datasheet del componente en el apartado de ANEXOS “7.8. Datasheets”).

Se han seleccionado los valores de los componentes utilizados mediante el análisis de ecuaciones y tablas proporcionadas por el fabricante en el datasheet del componente.

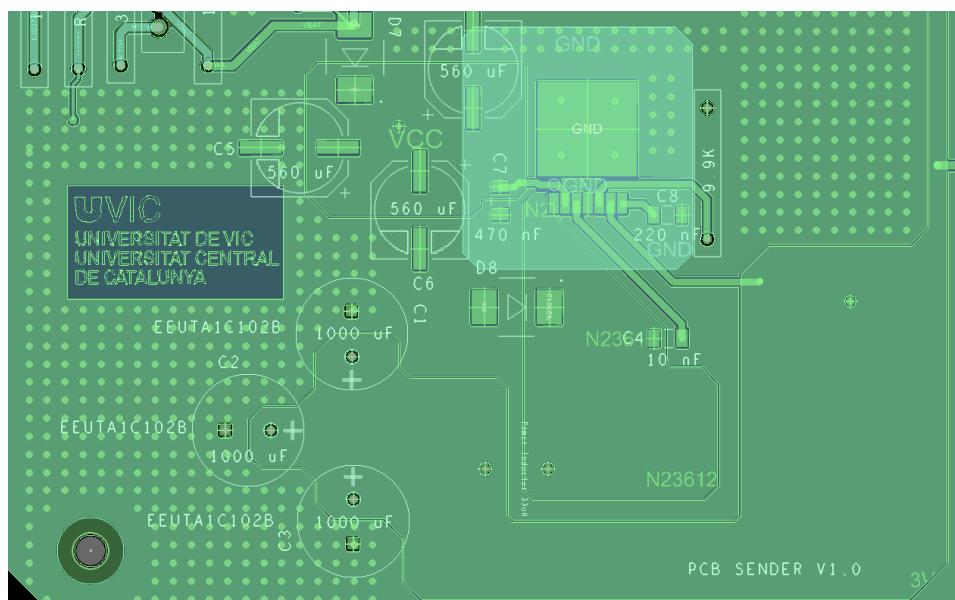


76. Esquema eléctrico ECU transmisora – Convertidor de potencia. Imagen de elaboración propia.

Es muy importante ceñirse al diseño recomendado del layout de este integrado para así evitar consecuencias negativas como ruido electromagnético que provocaría problemas de compatibilidad electromagnética con componentes cercanos, sobrecalentamiento por una mala disipación del calor, ruido y fluctuaciones del voltaje de salida, etc. En este caso Texas nos incluye un ejemplo visual de cómo debe ser el layout óptimo aprovechando el diseño de su placa de desarrollo de la familia LM267X. Se ha intentado replicar el mismo diseño de layout para el Buck converter en este caso, siguiendo el ejemplo proporcionado.



77. Layout ejemplo placa de desarrollo LM267X. Fuente: Texas Instruments datasheet.

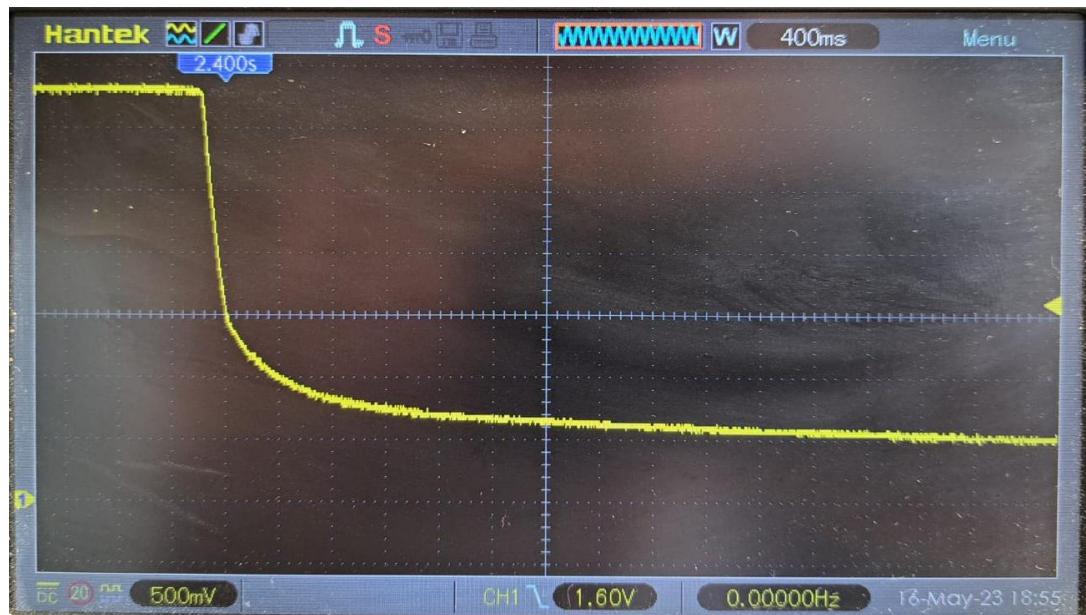


78. Layout actual del conversor de potencia buck integrado en la ECU transmisora. Imagen de elaboración propia.

Se han realizado mediciones exhaustivas utilizando un osciloscopio en el actual diseño de PCB (PCB SENDER V1.0) para garantizar su correcto funcionamiento y diseño. A continuación, se presentan las capturas obtenidas con el osciloscopio para visualizar los resultados de estas mediciones.



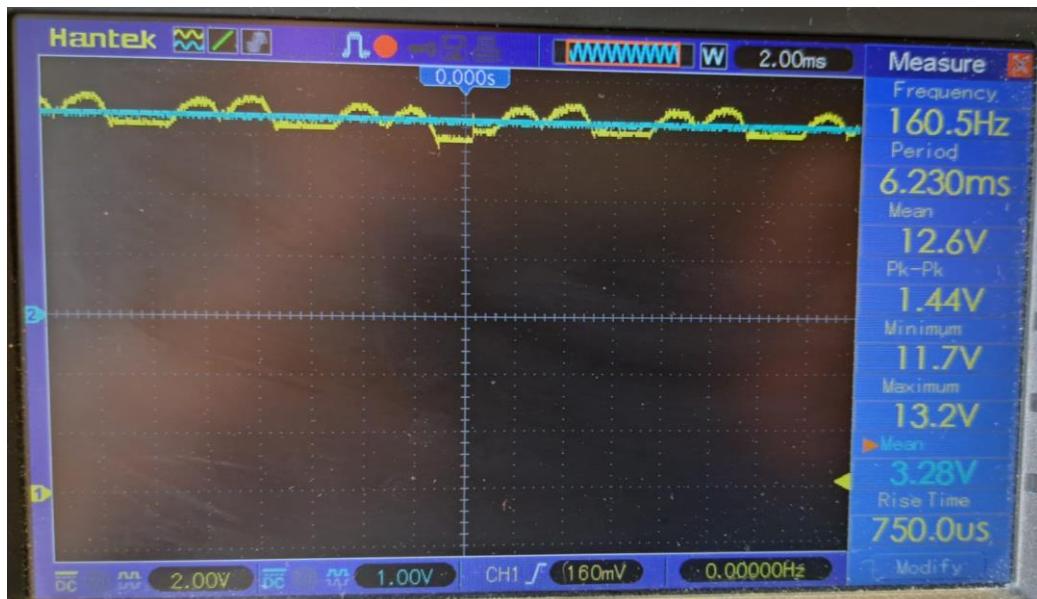
79. Convertidor Buck - Tensión de salida durante el encendido (soft start). Imagen de elaboración propia.



80. Convertidor Buck - Tensión de salida durante el apagado. Imagen de elaboración propia.



82. Convertidor Buck – Tensión pre inductor (canal amarillo) y tensión post inductor (canal azul). Imagen de elaboración propia.



81. Convertidor Buck – Tensión de salida (canal azul) y tensión de entrada de la batería (canal amarillo).
Imagen de elaboración propia.

El arranque suave o soft start se ha logrado en el convertidor Buck mediante la cuidadosa selección de los valores de los componentes, como se muestra en la figura 79. La implementación de esta característica tiene como objetivo principal proteger los componentes del sistema y mejorar su fiabilidad. Al evitar transitorios bruscos en la salida durante el arranque del convertidor step-down, se previenen daños o fallos prematuros que podrían ser causados por picos de sobretensión. La obtención de un soft start contribuye a un funcionamiento más seguro y confiable del sistema en general.

Algo no tan crucial pero como el arranque suave pero sí muy importante es el tiempo de descarga cuando de forma súbita se interrumpe la tensión de alimentación del módulo convertidor reductor.

Cuando la tensión de alimentación se interrumpe de forma abrupta, es importante saber cuánto tiempo se tiene de tensión mínima de funcionamiento para poder apagar los componentes sensibles que se puedan dañar. En este caso, como se puede observar en la figura 80, pasan aproximadamente 400 ms hasta que la tensión de salida cae hasta 1 V. Teniendo en cuenta que el microcontrolador tarda aproximadamente 10-15 microsegundos a dar una vuelta entera al programa, sería factible monitorizar la tensión de batería y apagar el microcontrolador en caso de desconexión repentina de la batería para evitar dañar el microcontrolador, módulo LoRa o el bridge UART-USB CP210X.

Es crucial destacar que, al considerar la homologación de este producto electrónico para su venta comercial, el tiempo de descarga de los condensadores adquiere una gran importancia en términos de seguridad para el usuario. En situaciones en las que se desconecta la alimentación y se manipula o realiza mantenimiento en el producto, existe un riesgo potencial de descarga eléctrica si la energía residual almacenada en los condensadores no se descarga lo suficientemente rápido.

El objetivo principal de un tiempo de descarga adecuado es garantizar que los condensadores se descarguen de manera segura y eficiente, minimizando así cualquier riesgo de descarga eléctrica al operario o al usuario final. Esto es especialmente relevante en aplicaciones donde la manipulación de los componentes internos del producto es necesaria o se espera, como en el caso de reparaciones, mantenimiento o cualquier situación en la que se requiera acceso al interior del dispositivo.

Al asegurarse de que los condensadores se descarguen rápidamente después de la desconexión de la alimentación, se reducen significativamente las posibilidades de que el operario sufra una descarga eléctrica peligrosa. Esto no solo cumple con los estándares de seguridad requeridos para la homologación del producto, sino que también brinda una mayor tranquilidad al usuario final al garantizar un entorno de trabajo seguro y confiable.

7.2.5. Consumo de la unidad de control transmisora

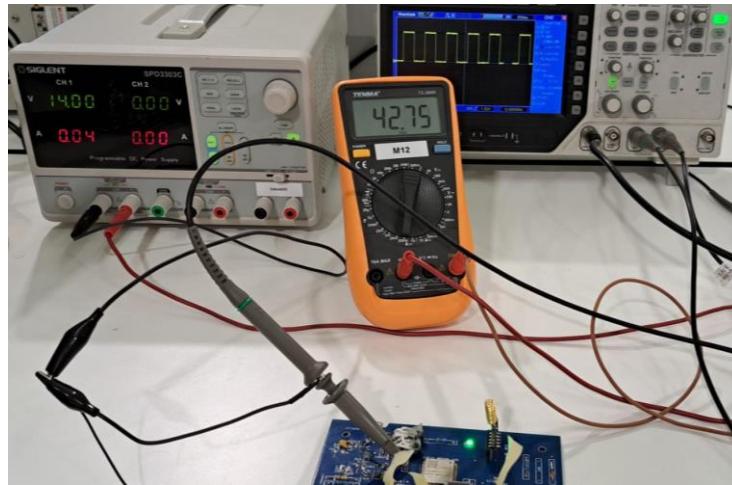
Dado que la unidad de control transmisora tiene dos modos de funcionamiento principales, es fundamental realizar mediciones en ambos modos con las mismas condiciones para obtener resultados objetivos y comparables. Estos modos son el modo standby, donde la unidad de control transmisora está en espera a que se accione la llave, y el modo de transmisión de datos, que es cuando la unidad de control transmisora consume más energía.

Al realizar mediciones en ambos modos de funcionamiento, se pueden evaluar de manera precisa y confiable los niveles de consumo de la unidad de control transmisora en situaciones reales. Esto proporciona información valiosa sobre la eficiencia energética del sistema y permite identificar cualquier variación significativa en el consumo de energía entre los modos de funcionamiento.

Al obtener resultados objetivos y consistentes, se pueden tomar decisiones sobre el diseño y la optimización del sistema, como la implementación de estrategias de gestión de energía o la selección de componentes más eficientes. Además, esta información es relevante para estimar la duración de la batería (durante la inactividad del módulo cuando el motor térmico se encuentra apagado) y evaluar el rendimiento global del sistema.

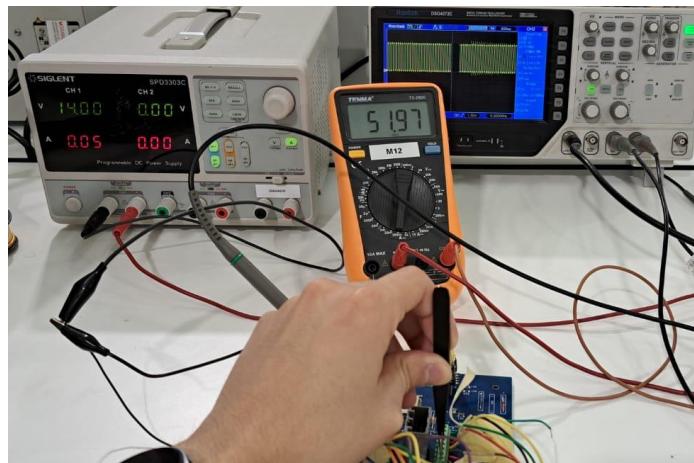
Para cuantificar el consumo de la unidad de control se ha utilizado un multímetro del fabricante “TENMA” modelo “72-2600” para ambas mediciones, cedido por el laboratorio de Can Muntanyola (Universidad de Vic).

Como se puede apreciar en las siguientes imágenes, se ha empleado un multímetro conectado en serie en la escala de miliamperios para realizar una medición precisa del consumo de la unidad de control transmisora en modo Standby. Además, se puede constatar que la ECU está en estado Standby al analizar la frecuencia de los pulsos de activación del LED indicador de estado, los cuales están siendo monitorizados mediante un osciloscopio ubicado detrás del multímetro.



83. Medición de consumo de la unidad de control transmisora en Standby.
Imagen de elaboración propia.

De igual manera, se ha llevado a cabo la medición del consumo de la ECU transmisora mientras se transmiten datos utilizando el módulo LoRa.



84. Medición de consumo de la unidad de control transmisora en comunicación. Imagen de elaboración propia.

Gracias a las anteriores mediciones, hemos podido comprobar que la Unidad de Control Electrónica Transmisora consume aproximadamente 42'75 mA en modo espera (standby), mientras que durante la transmisión de datos su consumo aumenta hasta los 51'97 mA.

Por último, se realizará una estimación del impacto que tendría esta unidad de control conectada a la moto, considerando el consumo base de todo su sistema eléctrico de la moto sin ECU transmisora, y mediante los datos previamente medidos.

Para ello, he obtenido el consumo de todo el sistema eléctrico de la moto sin la ECU transmisora.



85. Medición del consumo de todo el sistema eléctrico de la motocicleta sin ECU transmisora. Imagen de elaboración propia.

Esta motocicleta en particular viene equipada de fábrica con una batería YUASA de 12V y 8'4Ah. Utilizando la fórmula de la capacidad de una batería, podemos calcular que el vehículo tardaría aproximadamente 513'13 horas a descargarse entera la batería.

$$\text{Tiempo de descarga} = \frac{\text{Capacidad de la batería}}{\text{Consumo de corriente}}$$

$$\text{Tiempo} = \frac{8'4 \text{ Ah}}{0'01637 \text{ A}} = 513'13 \text{ horas}$$

Ahora, en caso de incorporar la ECU Transmisora al circuito eléctrico de la moto y que éste quede en Standby. El sistema eléctrico total de la moto aumentaría su consumo hasta los 59'12 mA. De forma que, con la misma batería el vehículo tardaría aproximadamente 142'08 horas a descargar completamente la batería.

$$\text{Tiempo} = \frac{8'4 \text{ Ah}}{0'04275 \text{ A} + 0'01637 \text{ A}} = \frac{8'4 \text{ Ah}}{0'05912 \text{ A}} = 142'08 \text{ horas}$$

Al integrar la unidad de control transmisora del sistema de telemetría desarrollado a lo largo de este proyecto, se ha observado un aumento del sistema eléctrico de la moto. Como resultado, la duración de la batería se vería reducida en un 72'28% en comparación a su tiempo de descarga sin la unidad de control transmisora.

Es crucial tener en cuenta los resultados de este análisis al considerar la integración permanente del sistema de telemetría en un vehículo. En tal caso, se deben implementar soluciones de gestión de consumo para reducir la corriente demandada por la ECU transmisora mientras se encuentra en modo de espera. De lo contrario, el vehículo podría agotar por completo la batería antes de lo previsto.

7.3. Limitaciones y mejoras que realizar en proyectos futuros

Con el objetivo de que este proyecto se pueda seguir desarrollando y con la intención de ayudar o facilitar a diseñar futuras mejoras, a continuación, expongo las mejoras que yo realizaría para una versión de PCB v2.0 tanto para la unidad de control transmisora como la receptora.

7.3.1. Secuencia de programación del microcontrolador ESP-32:

Una mejora indispensable para futuros diseños, con la intención de integrar este microcontrolador en cualquier unidad de control, es la incorporación de un circuito específico para la programación del microcontrolador.

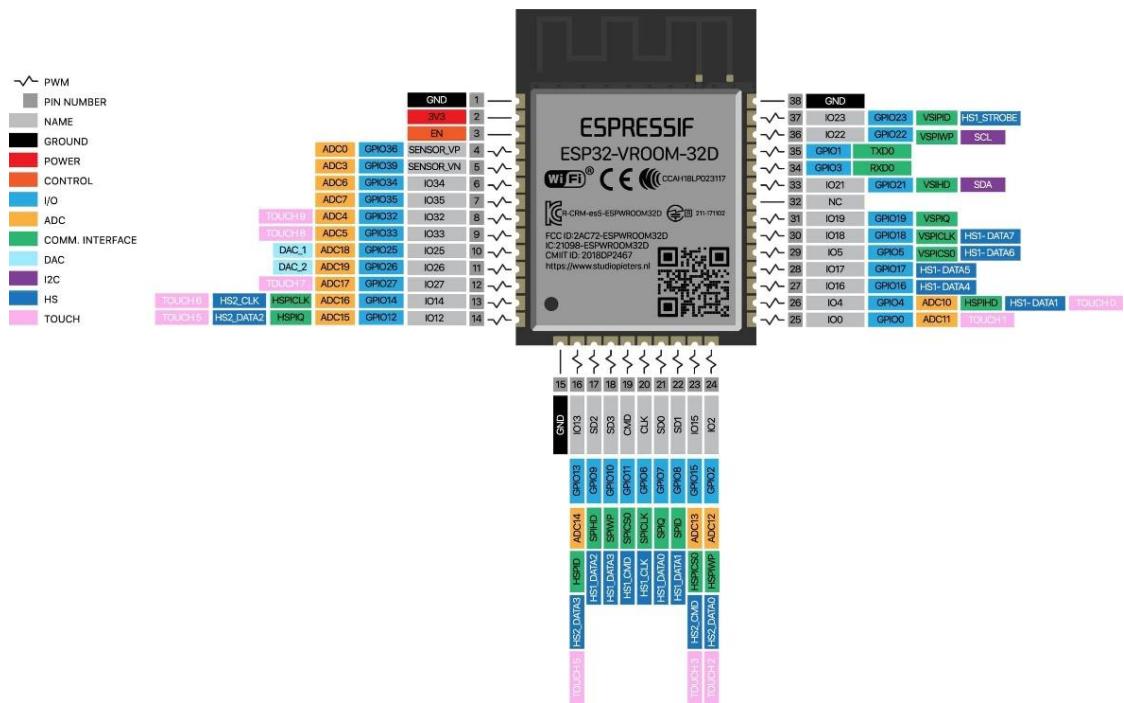
En el caso del ESP-32 y la mayoría de los microcontroladores, es necesario seguir una serie de pasos para poder reprogramarlos y modificar el código almacenado en su memoria interna. Estos pasos permiten pasar de la aplicación del firmware (código previamente cargado por el usuario) al modo bootloader. Este modo actúa como el gestor de arranque del microcontrolador y actúa como intermediario entre el hardware y el firmware programado por el usuario. Para realizar la reprogramación del firmware del ESP-32, es necesario acceder al modo bootloader, de lo contrario, el nuevo código no se grabará en la memoria del microcontrolador.

En específico, para el modelo de microcontrolador usado en este proyecto (ESP-WROOM-32D), se debe seguir la siguiente secuencia para acceder al bootloader:

1. Tensión de alimentación estable de +3'3 V.
2. Hacer un puente del GPIO 3 (EN) a masa y mantenerlo conectado hasta que se indique lo contrario.
3. Hacer un puente del GPIO 00 (IO00) a masa y mantenerlo conectado.
4. Retirar el puente hecho del GPIO 3 (EN) a masa mientras se mantiene el puente del GPIO 00.
5. Transcurridos un par de segundos, retirar el puente del GPIO 00 a masa.

Si se ha seguido correctamente esta secuencia, se observará una reducción en el consumo de corriente del microcontrolador ESP-32.

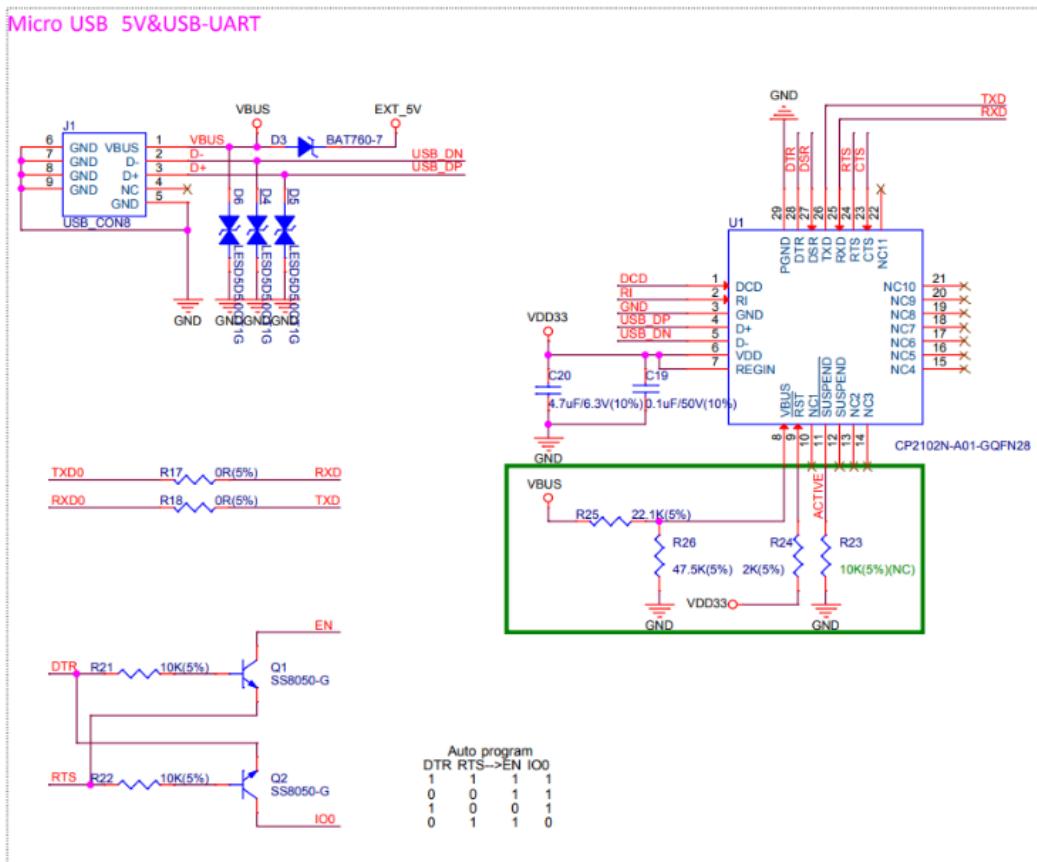
Una vez que se ha ingresado al modo GPIO, se puede flashear el firmware de manera habitual utilizando el puerto USB. Después de que el PC indique que la programación del código ha finalizado, es posible que el microcontrolador permanezca en el modo bootloader. Para asegurarnos de que el ESP-32 ejecute nuestro nuevo código, es necesario reiniciar el microcontrolador. Para lograrlo, simplemente debemos desconectar la alimentación del módulo y volver a conectarla, lo cual será suficiente para salir del modo bootloader.



86 Distribución de pines del microcontrolador ESP-32. Fuente: Espressif [20].

La mejora que sugiero para automatizar este proceso de reflasheo del firmware, basándome en el diseño de la placa de desarrollo de Espressif [21], es la incorporación de dos transistores dedicados específicamente para este propósito.

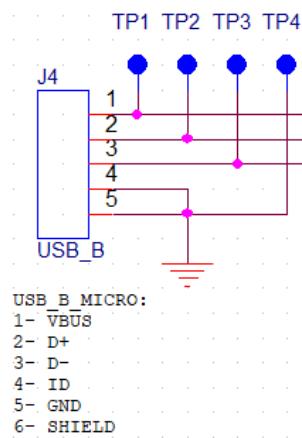
Mediante estos transistores BJT NPN y un bridge UART-USB de la familia CP2102, es posible automatizar este proceso. Esto se debe a que el integrado CP2102 cuenta con las señales DTR (Data Terminal Ready) y RTS (Request to Send), las cuales pueden activar los transistores para replicar la secuencia descrita anteriormente de forma automática.



87. Diseño de placa de desarrollo ESP-32 ESPRESSIF – Automatización del proceso para regrabar el microcontrolador.
Fuente: Espressif [21].

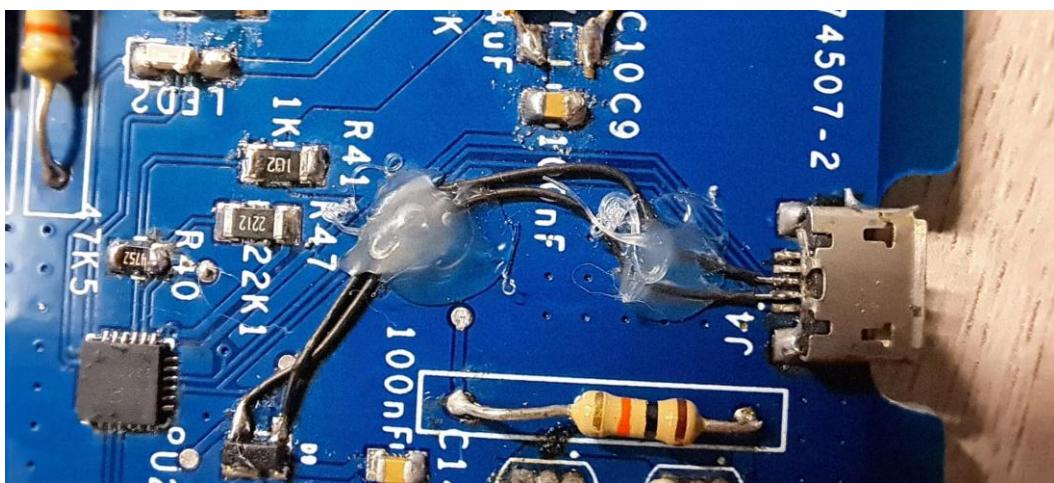
7.3.2. Pistas USB:

En la fase de diseño cometí un error al confundir las señales “DATA+” y “DATA-” correspondientes al conector USB tipo B. Este error se replicó tanto en la ECU receptora como en la ECU transmisora. Este error se solventa cambiando el orden en las pistas USB. De forma que, el pin 1 corresponde a la señal VBUS (5V), el pin 2 corresponde a la señal Data – (en vez de Data +), el pin 3 corresponde a la señal Data + (en vez de DATA -) y los dos pines restantes conectados a masa.



88. Distribución incorrecta de las señales USB presentes en los diseños actuales. Imagen de elaboración propia.

En las PCB ya fabricadas se ha podido solventar este fallo cortando las pistas de cobre y realizando el conexionado de forma externa a la PCB con cable “wire wrap”. El “wire wrap” consiste en unos cables unifilares de calibre pequeño maleables comúnmente usados para realizar puentes y poder solventar errores de diseño en prototipos.



89. Corrección de las pistas USB mediante wire wrap. Imagen de elaboración propia.

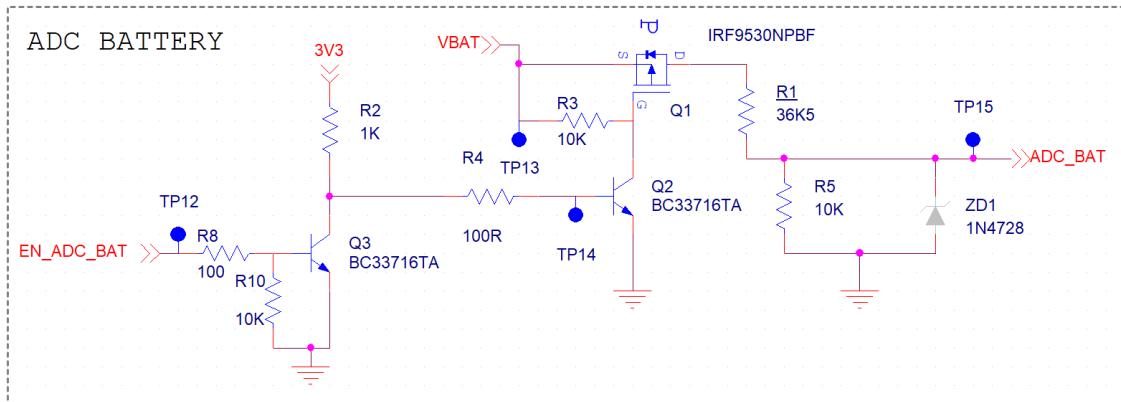
7.3.3. Conexionados transistores "Through Hole"

En el proceso de diseño de una PCB, como se mencionó anteriormente en el apartado "3.1.3. PCB", se siguen varias etapas clave. Estas etapas incluyen la creación de símbolos, su uso en el esquema eléctrico con las conexiones adecuadas, y luego la exportación del circuito al software de diseño de PCB. A continuación, se detallan los pasos involucrados en este proceso:

1. **Creación de símbolos:** En primer lugar, se crean los símbolos correspondientes a los componentes electrónicos que se utilizarán en el diseño de la PCB. Estos símbolos representan visualmente cada componente y sus conexiones eléctricas.
2. **Esquema eléctrico:** Utilizando los símbolos creados, se construye el esquema eléctrico, colocando los componentes en las posiciones adecuadas y conectándolos según las especificaciones del circuito. Esta etapa es fundamental para garantizar la correcta funcionalidad del circuito.
3. **Exportación del circuito:** Una vez finalizado el esquema eléctrico, se exporta el circuito al software de diseño de PCB. Esta exportación permite llevar toda la información del esquema, incluyendo los símbolos y las conexiones, al entorno de diseño de PCB para su posterior manipulación y optimización.
4. **Creación de footprints:** En el software de diseño de PCB, se crea el footprint o huella correspondiente a cada componente electrónico utilizado en el circuito. El footprint es una representación física del componente en la PCB, que incluye la disposición de los pines y las dimensiones precisas para su correcta colocación y soldadura.
5. **Vinculación de footprints:** Una vez que se ha creado el footprint para cada componente, se vincula correctamente al componente electrónico correspondiente en el esquema eléctrico. Esta vinculación asegura que, al abrir el circuito en el software de diseño de PCB, los footprints se muestren correctamente y se correspondan con los componentes en el esquema.

Durante la fase de diseño del hardware, en especial la creación de símbolos y su uso en el esquema eléctrico, se cometió el error de confundir la numeración de los pines tanto para los transistores BJT como para el transistor MOSFET.

A continuación, se muestra un circuito ejemplo donde se ha utilizado un transistor BJT.



90. Circuito para leer la tensión de la batería. Imagen de elaboración propia.

Como se puede observar, el esquema eléctrico del transistor NPN Q3 está correctamente diseñado. La tensión de 3.3V se aplica al colector del transistor, la señal de activación del transistor se aplica a la base y el pin emisor está conectado a masa.

Si revisamos el datasheet del transistor BC33716TA, vemos la siguiente numeración de los pines. El pin 1 se corresponde con el colector, el pin 2 con la base y el pin 3 con el emisor.

BC337 / BC338 NPN Epitaxial Silicon Transistor

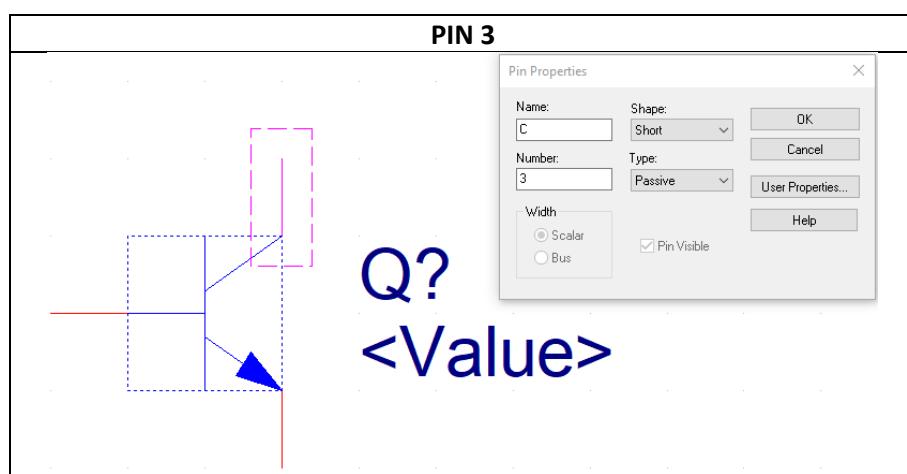
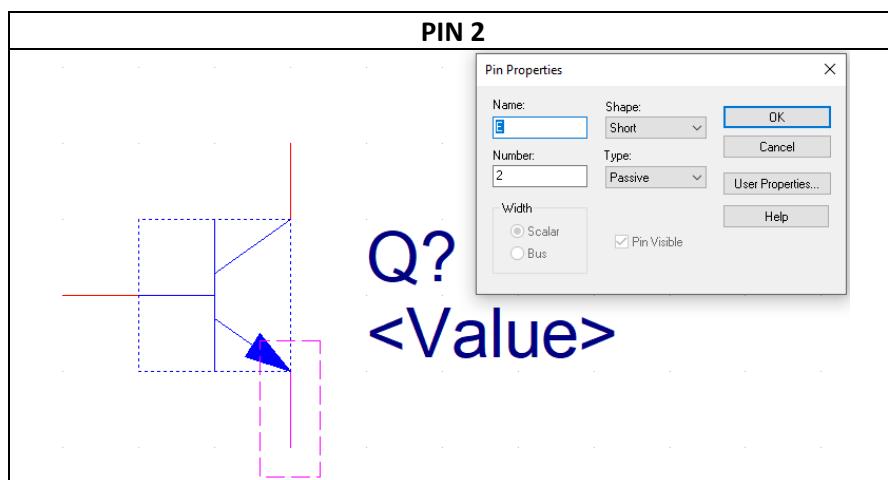
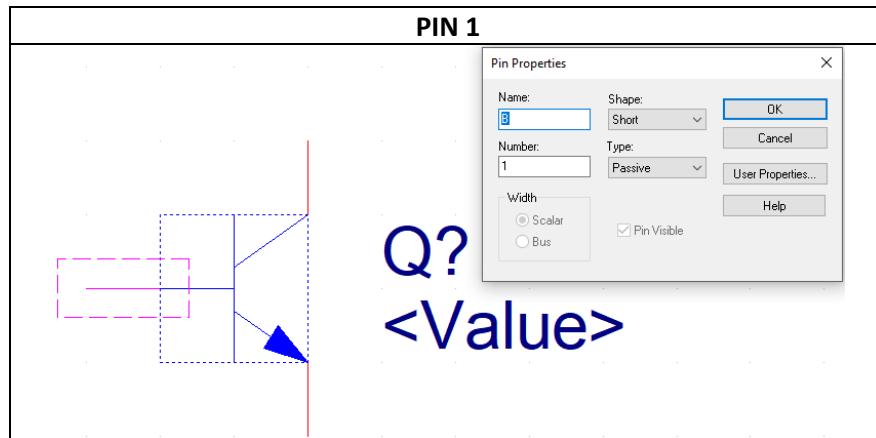
Features

- Switching and Amplifier Applications
- Suitable for AF-Driver Stages and Low-Power Output Stages
- Complement to BC327 / BC328



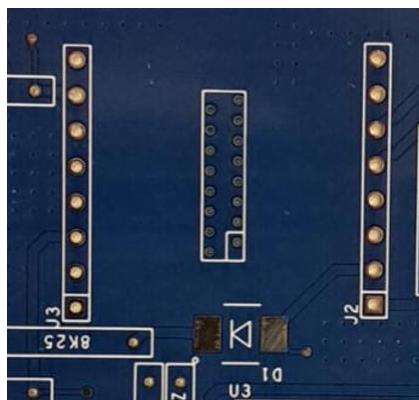
91. Recorte de la hoja de datos del transistor BC33716TA del fabricante FAIRCHILD. Fuente: Fairchild datasheet.

Al abrir el símbolo del transistor creado y se presta atención en la numeración de los pines se observa que el pin 1 corresponde con la base, el pin 2 con el emisor y el pin 3 con el colector. En lugar de esta numeración, se debería haber respetado la numeración propuesta por el fabricante o en su lugar haber usado la misma numeración del símbolo para el circuito y posteriormente para el diseño de la PCB.



7.3.4. *Footprint conector original Kawasaki "Through Hole"*

El footprint utilizado para el conector original de Kawasaki se diseñó con tolerancias muy ajustadas, lo que dificulta la inserción del conector en la PCB debido a la escasa holgura entre el conector y los agujeros destinados para su montaje. Para facilitar el montaje y la soldadura del componente, se recomienda encarecidamente aumentar el diámetro del agujero, así como el área de cobre alrededor de los agujeros pasantes que alojan al conector. Además, es importante ampliar el área de "anti-pad" para evitar la presencia de una zona conductora en una región específica alrededor de cada pad (destinado para el montaje del conector) y así reducir el riesgo de cortocircuitos.



92. Foto PCB Transmisora donde se muestra en el centro el footprint del conector de Kawasaki y en los laterales el footprint de los bloques terminales. Imagen de elaboración propia.

En la misma PCB, se puede tomar como referencia para las tolerancias los pads through-hole destinados para los bloques terminales situados en los laterales del conector original de Kawasaki.

7.3.5. *Layout y dimensiones generales PCB ECU Transmísora*

Esta primera versión de PCB ECU se ha realizado como un prototipo. Como tal, un prototipo necesita de varias iteraciones en su diseño para llegar a considerarse un producto final o producto acabado.

En este caso, para facilitar el estudio de errores en esta PCB y el comportamiento de todos los subsistemas que la conforman, se ha optado por un diseño con bastante distancia entre componentes y usar elementos through hole para las resistencias y diodos Zener así facilitando el intercambio de componentes y sus valores.

Lo ideal para obtener un mejor diseño y más compacto es utilizar componentes de montaje superficial SMD, por lo que, una vez se tenga un diseño mejorado se recomienda utilizar este tipo de componentes.

7.3.6. Encendido del microcontrolador ESP-32 en la PCB Transmisora

Al conectar la unidad de control transmisora por primera vez en la motocicleta Kawasaki Z800, se observa que el microcontrolador se enciende y entra automáticamente en modo bootloader en lugar de iniciar directamente al firmware programado. Como solución temporal, se requiere reiniciar el microcontrolador mediante un pulso de masa en el pin de EN para que funcione correctamente.

Con el fin de investigar esta situación, se decidió monitorear los pines relacionados con el modo bootloader durante el encendido del microcontrolador tanto en la ECU transmisora como en una placa de desarrollo ESP-32 diseñada por el fabricante ESPRESSIF.



93. Comportamiento del GPIO 00 (canal azul) y del pin EN (canal amarillo) durante el encendido de la ECU transmisora. Imagen de elaboración propia.



94. Comportamiento del GPIO 00 (canal azul) y del pin EN (canal amarillo) durante el encendido de la placa de desarrollo del ESP-32. Imagen de elaboración propia.

La principal diferencia entre el comportamiento de los pines durante el encendido en la placa de desarrollo y en la ECU transmisora radica en el tiempo que transcurre desde que el microcontrolador se alimenta con 3'3 V estables hasta que esta tensión llega al pin GPIO 00.

Efectivamente, como se puede observar, hay una diferencia significativa en el tiempo necesario para igualar las tensiones en ambos pines. Aproximadamente, en la placa de desarrollo se necesitan 11'2 ms a igualar las tensiones en ambos pines mientras que en la ECU transmisora se necesitan unos 20 ms. Muy probablemente esté ocurriendo que cuando el microcontrolador ya está encendido y completamente operativo, detecte que el GPIO 00 aún no ha superado la tensión umbral necesaria para detectar que tiene un valor 1 lógico y, por lo tanto, según la secuencia explicada en el punto “7.3.1. Secuencia de programación del microcontrolador ESP-32” el microcontrolador entra en modo bootloader.

Para resolver este problema en futuras mejoras de este proyecto se requiere ajustar el tiempo de estabilización de la tensión o implementar alguna otra solución que asegure que el pin GPIO 00 alcance la tensión adecuada antes de que el microcontrolador toma una decisión de arranque o modo de funcionamiento.

7.3.7. LDO ECU Receptora

La ECU receptora se alimenta mediante la tensión de 5V del conector USB. Para conseguir reducir la tensión de 5V a 3'3 V se ha decidido utilizar un LDO o regulador de baja caída de tensión. Este tipo de reguladores de tensión es mucho más eficiente que un regulador lineal pero más simple y menos eficiente que un regulador Buck.

Para este propósito se ha utilizado el regulador “*TC1017-3.3VLTR*” del fabricante Microchip. Este regulador permite una tensión de entrada mínima de 2'7 V y una tensión máxima de 6V. Suministra una tensión de salida de 5V estables con una corriente de 150 mA.

Este regulador de tensión ha resultado tener una corriente de salida máxima inferior a la necesaria por el sistema receptor. Sobre todo, en el encendido, que es cuando más corriente demanda el microcontrolador y el módulo “*RYLR998*”. Posterior al encendido la corriente se estabiliza, pero ese pico de consumo hace que el regulador “*TC1017-3.3VLTR*” se reinicie constantemente siendo inservible para suministrar la tensión adecuada a la ECU Receptora. Se eligió este componente durante la fase de desarrollo del hardware, en la cual se realizó la medición del consumo del sistema global pero no se tuvo en cuenta el pico de consumo de la unidad de control electrónica durante el encendido.

En su lugar, se recomienda encarecidamente el uso del regulador de voltaje LDO “*LM1117MPX-3.3/NOPB*” [22]. Este LDO se utiliza en el diseño de placa de desarrollo del microcontrolador ESP-32 [21] y ha funcionado correctamente durante todas las fases de pruebas y desarrollo de este proyecto. También, cuenta con un rango mayor de entre 1'4 V y 15 V tensión de entrada. Suministrando hasta una corriente máxima de 800 mA y 3'3 V estables.

7.4. Códigos de programación

En este proyecto, se ha desarrollado la programación de las unidades de control receptora y transmisora, cuyos códigos fuente ocupan aproximadamente 500 líneas de código cada archivo. Cada unidad de control contiene dos archivos (fichero “.ino” y “.h”). Además, hemos desarrollado una interfaz gráfica de usuario (GUI) con un código fuente compuesto por más de 600 líneas de código.

Para facilitar la revisión y el acceso a los códigos fuente, he decidido adjuntarlos en forma de archivos comprimidos junto con esta memoria del trabajo.

Dentro de la carpeta de los códigos fuente, se adjunta un archivo “Readme” con todo lo necesario para poder utilizar correctamente los archivos de programación.

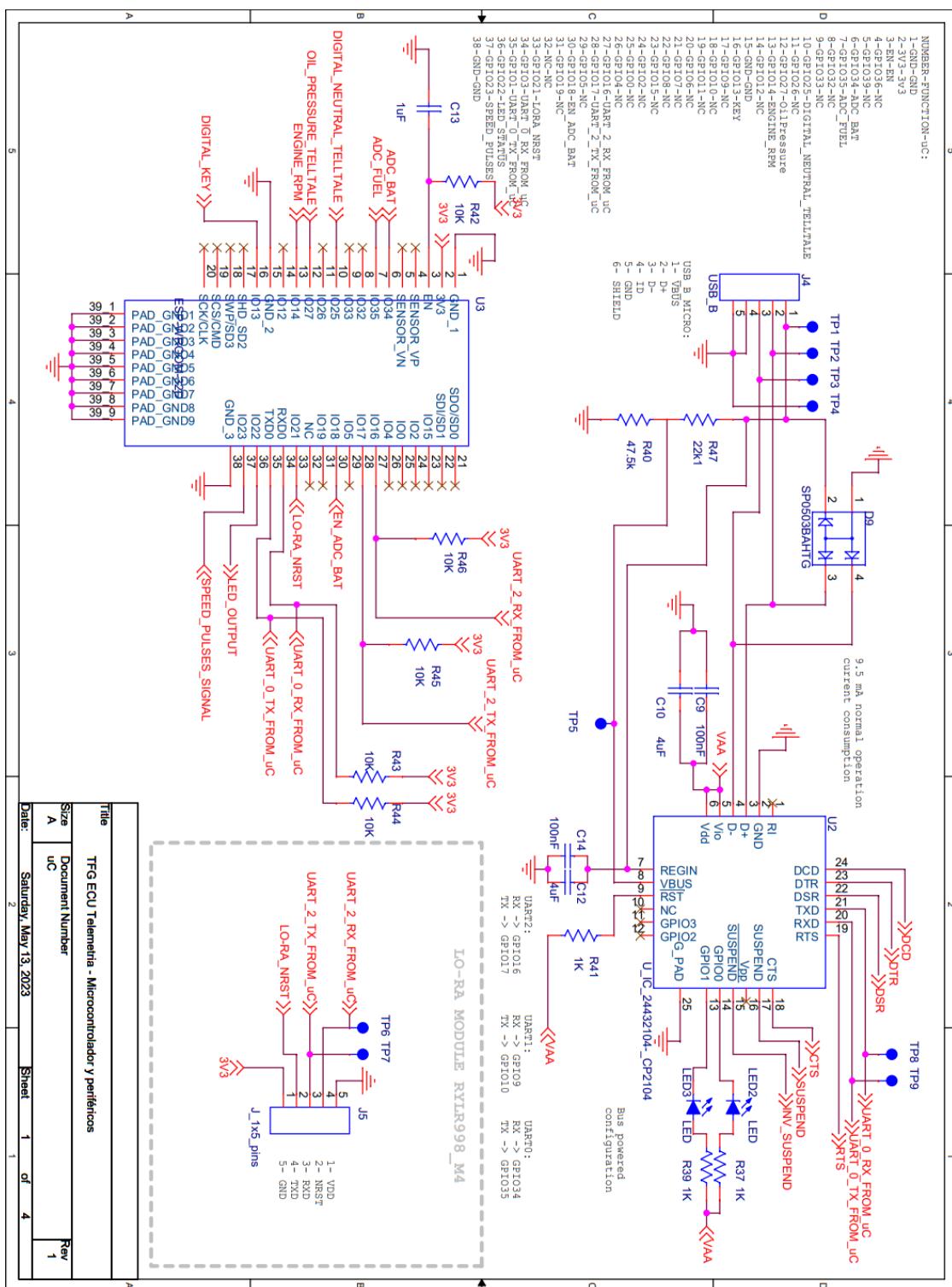
 ECU_Receiver_V1.00	04/06/2023 9:01	Carpeta de archivos
 ECU_Sender_V1.00	04/06/2023 8:47	Carpeta de archivos
 GUI_MAIN_Python	04/06/2023 9:15	Carpeta de archivos
 GUI_Submodulos_en_desarrollo PYTHON	04/06/2023 9:16	Carpeta de archivos
 README	04/06/2023 9:35	Documento de te... 2 KB

95. Carpeta de códigos fuente. Imagen de elaboración propia.

7.5. Esquemas de los circuitos electrónicos diseñados

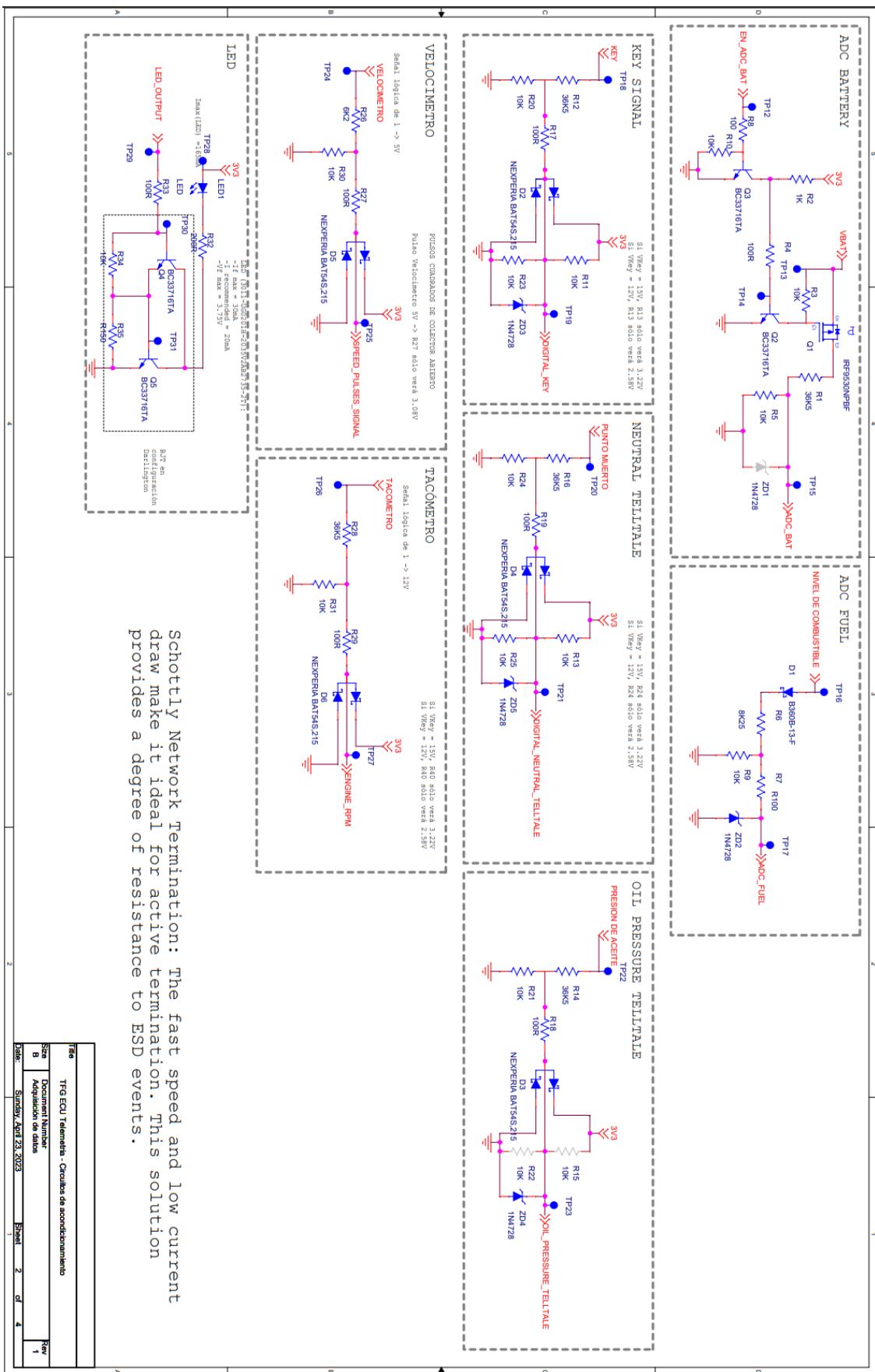
7.5.1. ECU Transmisora

7.5.1.1. ECU Transmisora – Microcontrolador y periféricos



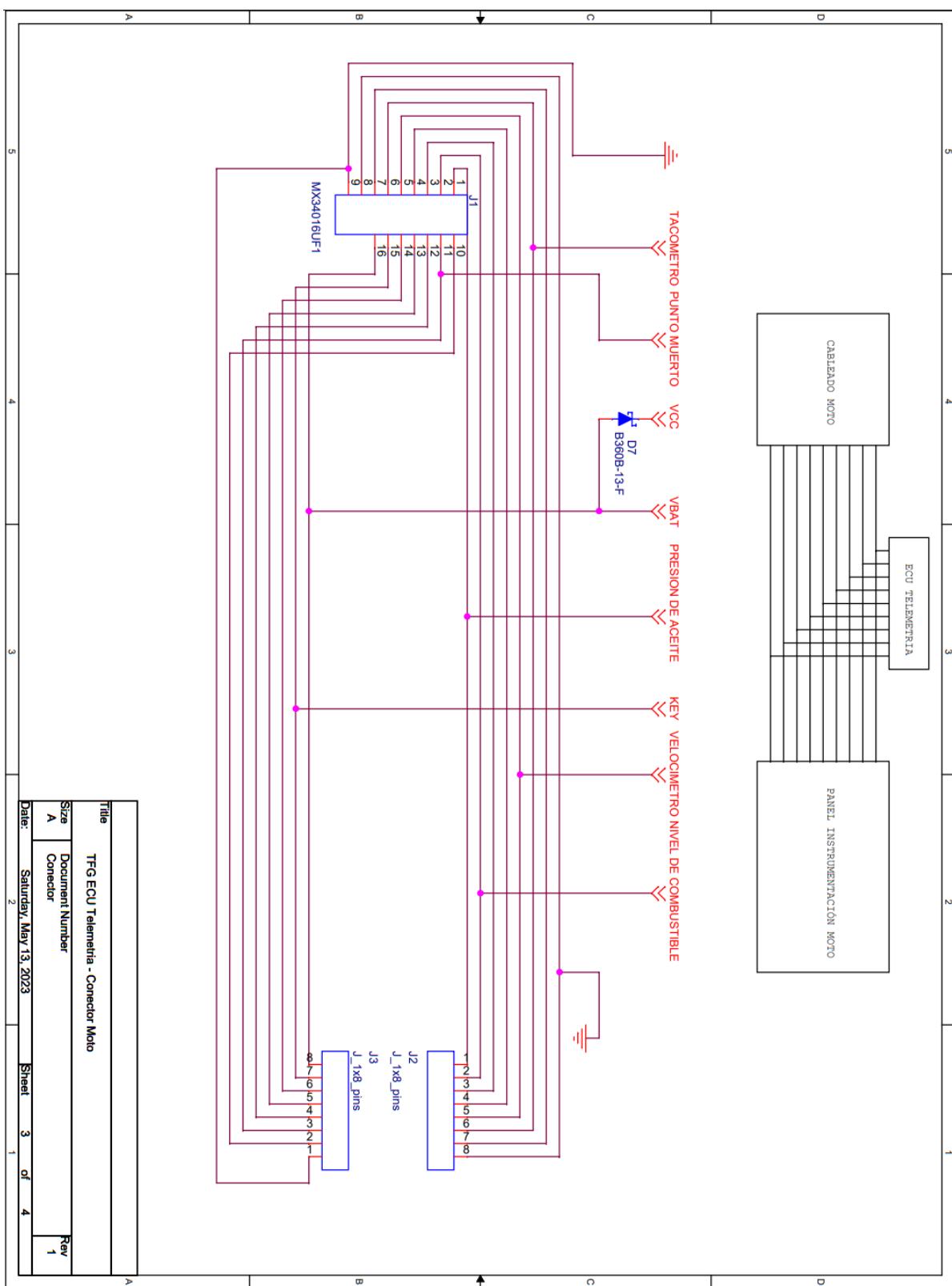
96. Esquema ECU Transmisora – Microcontrolador y periféricos. Imagen de elaboración propia.

7.5.1.2. ECU Transmisora – Circuitos de acondicionamiento



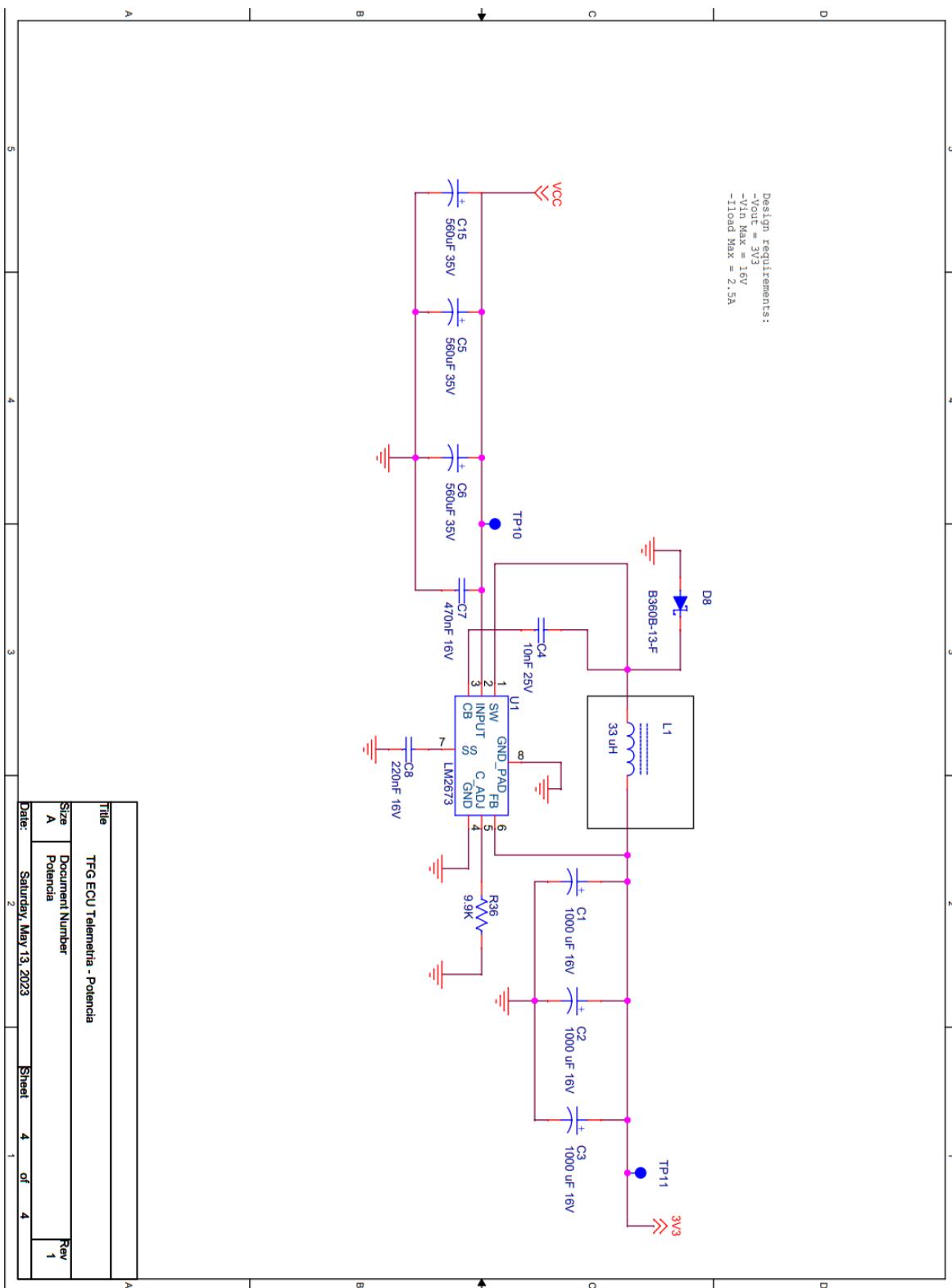
97. Esquema ECU Transmisora – Circuitos de acondicionamiento. Imagen de elaboración propia.

7.5.1.3. ECU Transmisora – Conector moto



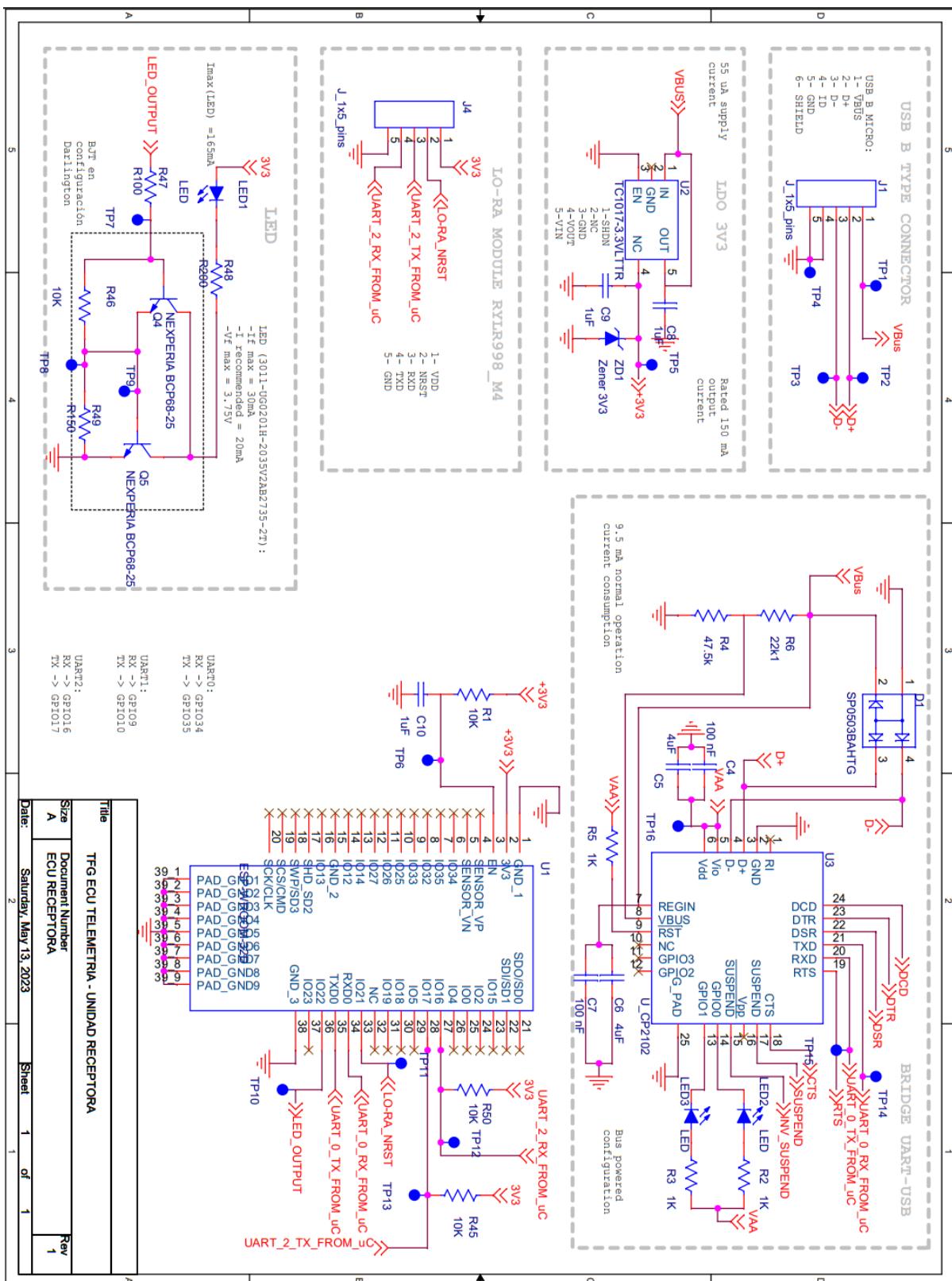
98. Esquema ECU Transmisora – Conector moto. Imagen de elaboración propia.

7.5.1.4. ECU Transmisora – Potencia



99. Esquema ECU Transmisora – Potencia. Imagen de elaboración propia.

7.5.2. ECU Receptora



100. Esquema ECU Receptora. Imagen de elaboración propia.

7.6. Lista de materiales

7.6.1. Unidad de control transmisora

Ítem	Cantidad	Referencia	Part
1	3	C1,C2,C3	1000 uF 16V
2	1	C4	10nF 25V
3	2	C5,C6	560uF 35V
4	1	C7	470nF 16V
5	1	C8	220nF 16V
6	2	C9,C14	100nF
7	2	C10,C12	4uF
8	1	C13	1uF
9	3	D1,D7,D8	B360B-13-F
10	5	D2,D3,D4,D5,D6	NEXPERIA BAT54S,215
11	1	D9	SP0503BAHTG
12	1	J1	MX34016UF1
13	2	J2,J3	J_1x8_pins
14	1	J4	USB_B
15	1	J5	J_1x5_pins
16	3	LED1,LED2,LED3	LED
17	1	L1	33 uH
18	1	Q1	IRF9530NPBF
19	4	Q2,Q3,Q4,Q5	BC33716TA
20	5	R1,R12,R14,R16,R28	36k5
21	4	R2,R37,R39,R41	1K
22	21	R3,R5,R9,R10,R11,R13,R15,R20,R21,R22, R23,R24,R25,R30,R31,R34,R42,R43,R44, R45,R46	10K
23	7	R4,R17,R18,R19,R27,R29,R33	100R
24	1	R6	8K25
25	2	R7,R8	100
26	1	R26	6k2
27	1	R32	200R
28	1	R35	R150
29	1	R36	9.9K
30	1	R40	47.5k
31	1	R47	22k1
32	1	U1	LM2673
33	1	U2	U_IC_24432104- _CP2104
34	1	U3	ESP-WROOM-32D
35	5	ZD1,ZD2,ZD3,ZD4,ZD5	1N4728

7.6.2. Unidad de control receptora

Ítem	Cantidad	Referencia	Part
1	3	C1,C2,C3	1uF
2	2	C4,C7	100 nF
3	2	C5,C6	4uF
4	1	D1	SP0503BAHTG
5	2	J1,J4	J_1x5_pins
6	3	LED1,LED2,LED3	LED
7	2	Q4,Q5	NEXPERIA BCP68-25
8	4	R1,R45,R46,R50	10K
9	3	R2,R3,R5	1K
10	1	R4	47.5k
11	1	R6	22k1
12	1	R47	100R
13	1	R48	200R
14	1	R49	R150
15	1	U1	ESP-WROOM-32D
16	1	U2	TC1017-3.3VLTR
17	1	U3	U_IC_24432104-_CP2104
18	1	ZD1	ZD_Zener

7.7. Comandos AT RYLR998

AT Command Set

It is required to key in “[enter](#)” or “[\r\n](#)” in the end of all AT Commands.

Add “? ” in the end of the commands to ask the current setting value.

It is required to wait until the module replies [+OK](#) so that you can execute the next AT command.

1. **AT** Test if the module can respond to Commands.

Syntax	Response
AT	+OK

2. **Software RESET**

Syntax	Response
AT+RESET	+RESET +READY

3. **AT+MODE** Set the wireless work mode.

Syntax	Response
<p>AT+MODE=<Parameter></p> <p><Parameter>range 0 to 1</p> <p>0 : Transceiver mode (default).</p> <p>1 : Sleep mode.</p> <p>Example : Set to sleep mode.</p> <p>AT+MODE=1</p> <p>2 : Smart receiving power saving mode</p> <p>The switch between receiving mode and sleep mode can be used to achieve the effect of power saving, and the appropriate transmission time must be adjusted by yourself to match this mode.</p> <p><RX time>=30ms~60000ms, (default 1000)</p> <p><Sleep time>=30ms~60000ms, (default 1000)</p> <p>When the correct LoRa® data format is received, it will return to the transceiver mode.</p> <p>When the received data is correct, +RCV format data will be output.</p> <p>Example : The Smart receiving power saving mode.</p> <p>AT+MODE=2,3000,3000</p> <p>Set to turn on receiving mode for 2 seconds and then sleep mode for two seconds to cycle until the correct signal is received.</p>	+OK
<p>AT+MODE?</p> <p>AT+MODE? Or Any digital signal</p> <p>AT+MODE? Or Any digital signal</p>	<p>'When MODE=0</p> <p>'When MODE=1</p> <p>'When MODE=2</p> <p>+MODE=0</p> <p>+MODE=0</p> <p>+MODE=0</p>

4. AT+IPR Set the UART baud rate.

Syntax	Response
<p>AT+IPR=<rate></p> <p><rate> is the UART baud rate :</p> <p>300 1200 4800 9600 19200 28800 38400 57600 115200(default)</p> <p>Example: Set the baud rate as 9600, <i>*The settings will be memorized in Flash.</i> AT+IPR=9600</p>	+IPR=<rate>
AT+IPR?	+IPR=9600

5. AT+BAND Set RF Frequency.

Syntax	Response
<p>AT+BAND=<parameter></p> <p><parameter>is the RF Frequency, Unit is Hz 470000000: 470000000Hz 915000000: 915000000Hz(default)</p> <p>Example: Set the frequency as 868500000Hz, AT+BAND=868500000</p>	+OK
AT+BAND?	+BAND=868500000

6. AT+PARAMETER Set the RF parameters.

Syntax	Response
<p>AT+PARAMETER=<Spreading Factor>, <Bandwidth>,<Coding Rate>, <Programmed Preamble></p> <p><Spreading Factor>7~11 (default 9)</p> <p>*SF7 to SF9 at 125kHz, SF7 to SF10 at 250kHz, and SF7 to SF11 at 500kHz</p> <p><Bandwidth>7~9, list as below :</p> <p>7: 125 KHz (default)</p> <p>8: 250 KHz</p> <p>9: 500 KHz</p> <p><Coding Rate>1~4, (default 1)</p> <p><Programmed Preamble>(default 12)</p> <p>When NETWORKID=18, The value can be configured to 4~24.</p> <p>Other NETWORKID can only be configured to 12.</p> <p>Example: Set the parameters as below, <Spreading Factor> 7, <Bandwidth> 500KHz, <Coding Rate> 4, <Programmed Preamble> 15. AT+PARAMETER=7,9,4,12</p>	+OK
AT+PARAMETER?	+PARAMETER=7,9,4,12

7. AT+ADDRESS Set the ADDRESS ID of module LoRa®.

Syntax	Response
<p>AT+ADDRESS=<Address></p> <p><Address>=0~65535 (default 0)</p> <p>Example: Set the address of module as 120.</p> <p>*The settings will be memorized in Flash.</p> <p>AT+ADDRESS=120</p>	+OK
AT+ADDRESS?	+ADDRESS=120

8. AT+NETWORKID Set the network ID.

Syntax	Response
AT+NETWORKID=<Network ID> <NetworkID>=3~15,18(default18) Example: Set the network ID as 6, *The settings will be memorized in Flash. AT+NETWORKID=6	+OK
AT+NETWORKID?	+NETWORK=6

9. AT+CPIN Set the domain password

Syntax	Response
AT+CPIN=<Password> <Password>An 8 character long password From 00000001 to FFFFFFFF, Only by using same password can the data be recognized. After resetting, the previously password will disappear. Example : Set the password to EEDCAA90 AT+CPIN=EEDCAA90	+OK
AT+CPIN? (default) AT+CPIN? (After setting the password)	+CPIN=No Password! +CPIN=EEDCAA90

10. AT+CRFOP Set the RF output power.

Syntax	Response
<p>AT+CRFOP=<power></p> <p><power>0~22 dBm</p> <p>22: 22dBm(default)</p> <p>21: 21dBm</p> <p>20: 20dBm</p> <p>.....</p> <p>01: 1dBm</p> <p>00: 0dBm</p> <p>Example: Set the output power as 10dBm, AT+CRFOP=10</p> <p>* RF Output Power must be set to less than AT+CRFOP=14 to comply CE certification.</p>	+OK
AT+CRFOP?	+CRFOP=10

11. AT+SEND Send data to the appointed address by Command Mode.

Syntax	Response
<p>AT+SEND=<Address>,<Payload Length>,<Data></p> <p><Address>0~65535, When the <Address> is 0, it will send data to all address (From 0 to 65535.)</p> <p><Payload Length> Maximum 240bytes</p> <p><Data>ASCII Format</p> <p>Example : Send HELLO string to the Address 50, AT+SEND=50,5,HELLO</p>	+OK
Search last transmit data, AT+SEND?	+SEND=50,5,HELLO

12. +RCV Show the received data actively.

Syntax	Response
<pre>+RCV=<Address>,<Length>,<Data>,<RSSI>,<SNR>, <Address> Transmitter Address ID <Length> Data Length <Data> ASCII Format Data <RSSI> Received Signal Strength Indicator <SNR> Signal-to-noise ratio</pre>	
Example: Module received the ID Address 50 send 5 bytes data, Content is HELLO string, RSSI is -99dBm, SNR is 40, It will show as below. +RCV=50, 5, HELLO, -99, 40	

13. AT+UID? To inquire module ID. 12BYTES

Syntax	Response
AT+UID?	+UID=104737333437353600170029

14. AT+VER? To inquire the firmware version.

Syntax	Response
AT+VER?	+VER=RYLRx8_Vx.x.x

15. AT+FACTORY Set all current parameters to manufacturer defaults.

Syntax	Response
<pre>AT+FACTORY Manufacturer defaults: BAND : 915MHz UART : 115200 Spreading Factor : 9 Bandwidth : 125kHz Coding Rate : 1 Preamble Length : 12 Address : 0 Network ID : 18 CRFOP : 22</pre>	+FACTORY

16. Other messages

Narrative	Response
After RESET	+RESET +READY

17. Error result codes

Narrative	Response
There is not "enter" or 0x0D 0x0A in the end of the AT Command.	+ERR=1
The head of AT command is not "AT" string.	+ERR=2
Unknow command.	+ERR=4
The data to be sent does not match the actual length	+ERR=5
TX is over times.	+ERR=10
CRC error.	+ERR=12
TX data exceeds 240bytes.	+ERR=13
Failed to write flash memory.	+ERR=14
Unknow failure.	+ERR=15
Last TX was not completed	+ERR=17
Preamble value is not allowed.	+ERR=18
RX failed, Header error	+ERR=19
The time setting value of the "Smart receiving power saving mode" is not allowed.	+ERR=20

Fuente: Reyax [12].

7.8. Datasheets

Transistor BJT: BC33716TA

Absolute Maximum Ratings

Stresses exceeding the absolute maximum ratings may damage the device. The device may not function or be operable above the recommended operating conditions and stressing the parts to these levels is not recommended. In addition, extended exposure to stresses above the recommended operating conditions may affect device reliability. The absolute maximum ratings are stress ratings only. Values are at $T_A = 25^\circ\text{C}$ unless otherwise noted.

Symbol	Parameter	Value	Unit
V_{CES}	Collector-Emitter Voltage	BC337	50
		BC338	30
V_{CEO}	Collector-Emitter Voltage	BC337	45
		BC338	25
V_{EBO}	Emitter-Base Voltage	5	V
I_C	Collector Current (DC)	800	mA
T_J	Junction Temperature	150	$^\circ\text{C}$
T_{STG}	Storage Temperature	-55 to 150	$^\circ\text{C}$

Electrical Characteristics

Values are at $T_A = 25^\circ\text{C}$ unless otherwise noted.

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
BV_{CEO}	Collector-Emitter Breakdown Voltage	BC337 BC338	$I_C = 10 \text{ mA}, I_B = 0$	45		
				25		
BV_{CES}	Collector-Emitter Breakdown Voltage	BC337 BC338	$I_C = 0.1 \text{ mA}, V_{BE} = 0$	50		
				30		
BV_{EBO}	Emitter-Base Breakdown Voltage	$I_E = 0.1 \text{ mA}, I_C = 0$	5			V
I_{CES}	Collector Cut-Off Current	BC337 BC338	$V_{CE} = 45 \text{ V}, I_B = 0$ $V_{CE} = 25 \text{ V}, I_B = 0$		2	100
					2	100
h_{FE1}	DC Current Gain		$V_{CE} = 1 \text{ V}, I_C = 100 \text{ mA}$	100		630
h_{FE2}			$V_{CE} = 1 \text{ V}, I_C = 300 \text{ mA}$	60		
$V_{CE(\text{sat})}$	Collector-Emitter Saturation Voltage	$I_C = 500 \text{ mA}, I_B = 50 \text{ mA}$			0.7	V
$V_{BE(\text{on})}$	Base-Emitter On Voltage	$V_{CE} = 1 \text{ V}, I_C = 300 \text{ mA}$			1.2	V
f_T	Current Gain Bandwidth Product	$V_{CE} = 5 \text{ V}, I_C = 10 \text{ mA}, f = 50 \text{ MHz}$		100		MHz
C_{ob}	Output Capacitance	$V_{CB} = 10 \text{ V}, I_E = 0, f = 1 \text{ MHz}$		12		pF

h_{FE} Classification

Classification	16	25	40
h_{FE1}	100 ~ 250	160 ~ 400	250 ~ 630
h_{FE2}	60 ~	100 ~	170 ~

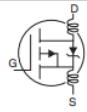
Fuente: Onsemi datasheet.

Transistor MOSFET de CANAL-P: IRF9530NPBF

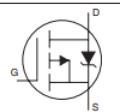
Absolute Maximum Ratings

	Parameter	Max.	Units
$I_D @ T_C = 25^\circ\text{C}$	Continuous Drain Current, $V_{GS} @ -10\text{V}$	-14	A
$I_D @ T_C = 100^\circ\text{C}$	Continuous Drain Current, $V_{GS} @ -10\text{V}$	-10	
I_{DM}	Pulsed Drain Current ①	-56	
$P_D @ T_C = 25^\circ\text{C}$	Power Dissipation	79	W
	Linear Derating Factor	0.53	W/ $^\circ\text{C}$
V_{GS}	Gate-to-Source Voltage	± 20	V
E_{AS}	Single Pulse Avalanche Energy ②	250	mJ
I_{AR}	Avalanche Current ③	-8.4	A
E_{AR}	Repetitive Avalanche Energy ③	7.9	mJ
dv/dt	Peak Diode Recovery dv/dt ③	-5.0	V/ns
T_J	Operating Junction and Storage Temperature Range	-55 to + 175	$^\circ\text{C}$
T_{STG}	Soldering Temperature, for 10 seconds	300 (1.6mm from case)	
	Mounting torque, 6-32 or M3 screw	10 lbf·in (1.1N·m)	

Electrical Characteristics @ $T_J = 25^\circ\text{C}$ (unless otherwise specified)

	Parameter	Min.	Typ.	Max.	Units	Conditions
$V_{(BR)DSS}$	Drain-to-Source Breakdown Voltage	-100	—	—	V	$V_{GS} = 0\text{V}, I_D = -250\mu\text{A}$
$\Delta V_{(BR)DSS}/\Delta T_J$	Breakdown Voltage Temp. Coefficient	—	-0.11	—	V/ $^\circ\text{C}$	Reference to 25°C , $I_D = -1\text{mA}$
$R_{DS(on)}$	Static Drain-to-Source On-Resistance	—	—	0.20	Ω	$V_{GS} = -10\text{V}, I_D = -8.4\text{A}$ ④
$V_{GS(th)}$	Gate Threshold Voltage	-2.0	—	-4.0	V	$V_{DS} = V_{GS}, I_D = -250\mu\text{A}$
g_{fs}	Forward Transconductance	3.2	—	—	S	$V_{DS} = -50\text{V}, I_D = -8.4\text{A}$
I_{DSS}	Drain-to-Source Leakage Current	—	—	-25	μA	$V_{DS} = -100\text{V}, V_{GS} = 0\text{V}$
		—	—	-250		$V_{DS} = -80\text{V}, V_{GS} = 0\text{V}, T_J = 150^\circ\text{C}$
I_{GSS}	Gate-to-Source Forward Leakage	—	—	100	nA	$V_{GS} = 20\text{V}$
	Gate-to-Source Reverse Leakage	—	—	-100		$V_{GS} = -20\text{V}$
Q_g	Total Gate Charge	—	—	58	nC	$I_D = -8.4\text{A}$
Q_{gs}	Gate-to-Source Charge	—	—	8.3		$V_{DS} = -80\text{V}$
Q_{gd}	Gate-to-Drain ("Miller") Charge	—	—	32		$V_{GS} = -10\text{V}$, See Fig. 6 and 13 ④
$t_{d(on)}$	Turn-On Delay Time	—	15	—	ns	$V_{DD} = -50\text{V}$
t_r	Rise Time	—	58	—		$I_D = -8.4\text{A}$
$t_{d(off)}$	Turn-Off Delay Time	—	45	—		$R_G = 9.1\Omega$
t_f	Fall Time	—	46	—		$R_D = 6.2\Omega$, See Fig. 10 ④
L_D	Internal Drain Inductance	—	4.5	—	nH	Between lead, 6mm (0.25in.) from package and center of die contact
L_S	Internal Source Inductance	—	7.5	—		
C_{iss}	Input Capacitance	—	760	—	pF	$V_{GS} = 0\text{V}$
C_{oss}	Output Capacitance	—	260	—		$V_{DS} = -25\text{V}$
C_{rss}	Reverse Transfer Capacitance	—	170	—		$f = 1.0\text{MHz}$, See Fig. 5

Source-Drain Ratings and Characteristics

	Parameter	Min.	Typ.	Max.	Units	Conditions
I_S	Continuous Source Current (Body Diode)	—	—	-14	A	MOSFET symbol showing the integral reverse p-n junction diode.
I_{SM}	Pulsed Source Current (Body Diode) ①	—	—	-56		
V_{SD}	Diode Forward Voltage	—	—	-1.6		$T_J = 25^\circ\text{C}, I_S = -8.4\text{A}, V_{GS} = 0\text{V}$ ④
t_{rr}	Reverse Recovery Time	—	130	190	ns	$T_J = 25^\circ\text{C}, I_F = -8.4\text{A}$
Q_{rr}	Reverse Recovery Charge	—	650	970	nC	$di/dt = -100\text{A}/\mu\text{s}$ ④
t_{on}	Forward Turn-On Time	Intrinsic turn-on time is negligible (turn-on is dominated by L_S+L_D)				

Fuente: Infineon datasheet.

Regulador de conmutación Buck: LM2673 S-3.3



LM2673

SNVS030O –APRIL 2000–REVISED JUNE 2016

LM2673 SIMPLE SWITCHER® 3-A Step-Down Voltage Regulator With Adjustable Current Limit

1 Features

- Efficiency Up to 94%
- Simple and Easy to Design With (Using Off-The-Shelf External Components)
- Resistor Programmable Peak Current Limit Over a Range of 2 A to 5 A
- 150-mΩ DMOS Output Switch
- 3.3-V, 5-V, 12-V Fixed Output and Adjustable (1.2 V to 37 V) Versions
- ±2% Maximum Output Tolerance Over Full Line and Load Conditions
- Wide Input Voltage Range: 8 V to 40 V
- 260-KHz Fixed Frequency Internal Oscillator
- Soft-Start Capability
- -40 to 125°C Operating Junction Temperature Range

2 Applications

- Simple-to-Design, High Efficiency (>90%) Step-Down Switching Regulators
- Efficient System Preregulator for Linear Voltage Regulators
- Battery Chargers

3 Description

The LM2673 series of regulators are monolithic integrated circuits which provide all of the active functions for a step-down (buck) switching regulator capable of driving up to 3-A loads with excellent line and load regulation characteristics. High efficiency (>90%) is obtained through the use of a low ON-resistance DMOS power switch. The series consists of fixed output voltages of 3.3 V, 5 V, and 12 V and an adjustable output version.

The SIMPLE SWITCHER® concept provides for a complete design using a minimum number of external components. A high fixed frequency oscillator (260 kHz) allows the use of physically smaller sized components. A family of standard inductors for use with the LM2673 are available from several manufacturers to greatly simplify the design process.

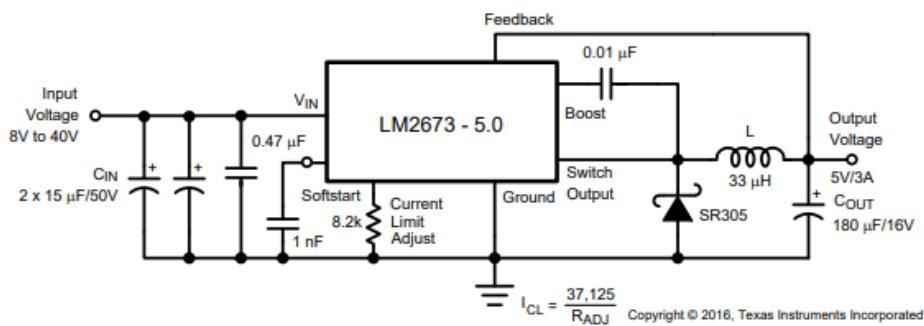
Other features include the ability to reduce the input surge current at power on by adding a soft-start timing capacitor to gradually turn on the regulator. The LM2673 series also has built-in thermal shutdown and resistor programmable current limit of the power MOSFET switch to protect the device and load circuitry under fault conditions. The output voltage is ensured to a ±2% tolerance. The clock frequency is controlled to within a ±11% tolerance.

Device Information⁽¹⁾

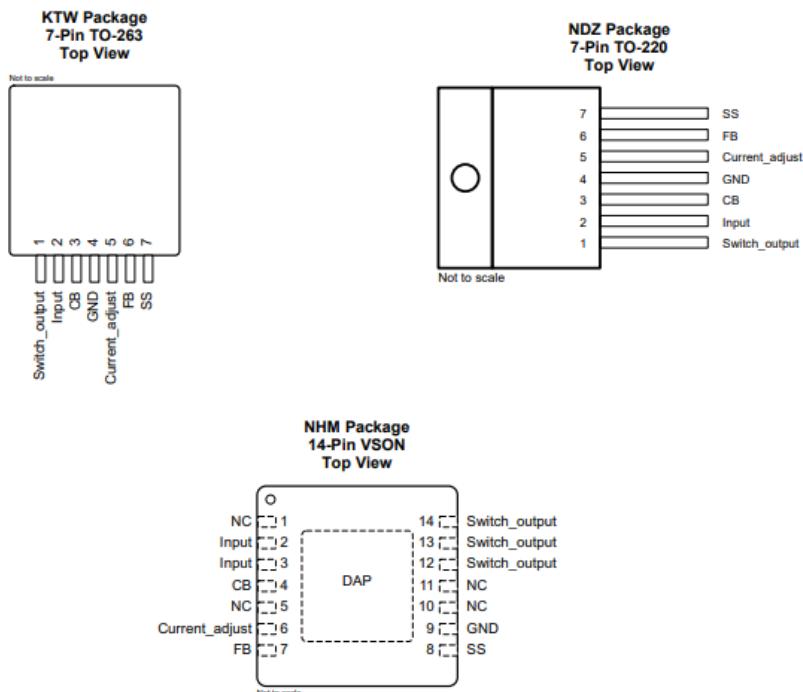
PART NUMBER	PACKAGE	BODY SIZE (NOM)
LM2673	TO-263 (7)	10.10 mm × 8.89 mm
	TO-220 (7)	14.986 mm × 10.16 mm
	VSON (14)	6.00 mm × 5.00 mm

(1) For all available packages, see the orderable addendum at the end of the data sheet.

Typical Application



5 Pin Configuration and Functions



Connect DAP to pin 9 on PCB.

Pin Functions

NAME	PIN		I/O	DESCRIPTION
	TO-263, TO-220	VSON		
Switch output	1	12, 13, 14	O	Source pin of the internal High Side FET. This is a switching node. Attached this pin to an inductor and the cathode of the external diode.
Input	2	2, 3	I	Supply input pin to collector pin of high side FET. Connect to power supply and input bypass capacitors CIN. Path from VIN pin to high frequency bypass CIN and GND must be as short as possible.
CB	3	4	I	Boot-strap capacitor connection for high-side driver. Connect a high quality 100-nF capacitor from CB to VSW Pin.
GND	4	9	—	Power ground pins. Connect to system ground. Ground pins of CIN and COUT. Path to CIN must be as short as possible.
Current adjust	5	6	I	Current Limit adjust pin. Connect a resistor from this pin to GND to set the current limit of the part.
FB	6	7	I	Feedback sense input pin. Connect to the midpoint of feedback divider to set VOUT for ADJ version or connect this pin directly to the output capacitor for a fixed output version.
SS	7	8	I	Soft-start pin. Connect a capacitor from this pin to GND to control the output voltage ramp. If the feature not desired, the pin can be left floating.
NC	—	1, 5, 10, 11	—	No connect pins

6 Specifications

6.1 Absolute Maximum Ratings⁽¹⁾⁽²⁾

		MIN	MAX	UNIT	
Input supply voltage		45	V		
Soft-start pin voltage		-0.1	6	V	
Switch voltage to ground ⁽³⁾		-1	V _{IN}	V	
Boost pin voltage		V _{SW} + 8 V		V	
Feedback pin voltage		-0.3	14	V	
Power dissipation		Internally Limited			
Soldering temperature	Wave, 4 s	260		°C	
	Infrared, 10 s	240			
	Vapor phase, 75 s	219			
Storage temperature, T _{stg}		-65	150	°C	

- (1) Stresses beyond those listed under *Absolute Maximum Ratings* may cause permanent damage to the device. These are stress ratings only, which do not imply functional operation of the device at these or any other conditions beyond those indicated under *Recommended Operating Conditions*. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.
- (2) If Military/Aerospace specified devices are required, please contact the Texas Instruments Sales Office/Distributors for availability and specifications.
- (3) The absolute maximum specification of the *Switch Voltage to Ground* applies to DC voltage. An extended negative voltage limit of -10 V applies to a pulse of up to 20 ns, -6 V of 60 ns and -3 V of up to 100 ns.

6.2 ESD Ratings

		VALUE	UNIT
V _(ESD)	Electrostatic discharge Human-body model (HBM), per ANSI/ESDA/JEDEC JS-001 ⁽¹⁾⁽²⁾	±2000	V

- (1) JEDEC document JEP155 states that 500-V HBM allows safe manufacturing with a standard ESD control process.
- (2) ESD was applied using the human-body model, a 100-pF capacitor discharged through a 1.5-kΩ resistor into each pin.

6.3 Recommended Operating Conditions

		MIN	MAX	UNIT
Supply voltage		8	40	V
Junction temperature (T _J)		-40	125	°C

6.5 Electrical Characteristics: LM2673 – 3.3 V

Specifications apply for T_A = T_J = 25°C unless otherwise noted. R_{ADJ} = 5.6 kΩ.

PARAMETER	TEST CONDITIONS		MIN ⁽¹⁾	TYP ⁽²⁾	MAX ⁽¹⁾	UNIT
V _{OUT} Output voltage	V _{IN} = 8 V to 40 V, 100 mA ≤ I _{OUT} ≤ 5 A	over the entire junction temperature range of operation -40°C to 125°C	3.234	3.3	3.366	V
η Efficiency	V _{IN} = 12 V, I _{LOAD} = 5 A			86%		

- (1) All room temperature limits are 100% tested during production with T_A = T_J = 25°C. All limits at temperature extremes are specified via correlation using standard Quality Control (SQC) methods. All limits are used to calculate Average Outgoing Quality Level (AOQL).
- (2) Typical values are determined with T_A = T_J = 25°C and represent the most likely norm.

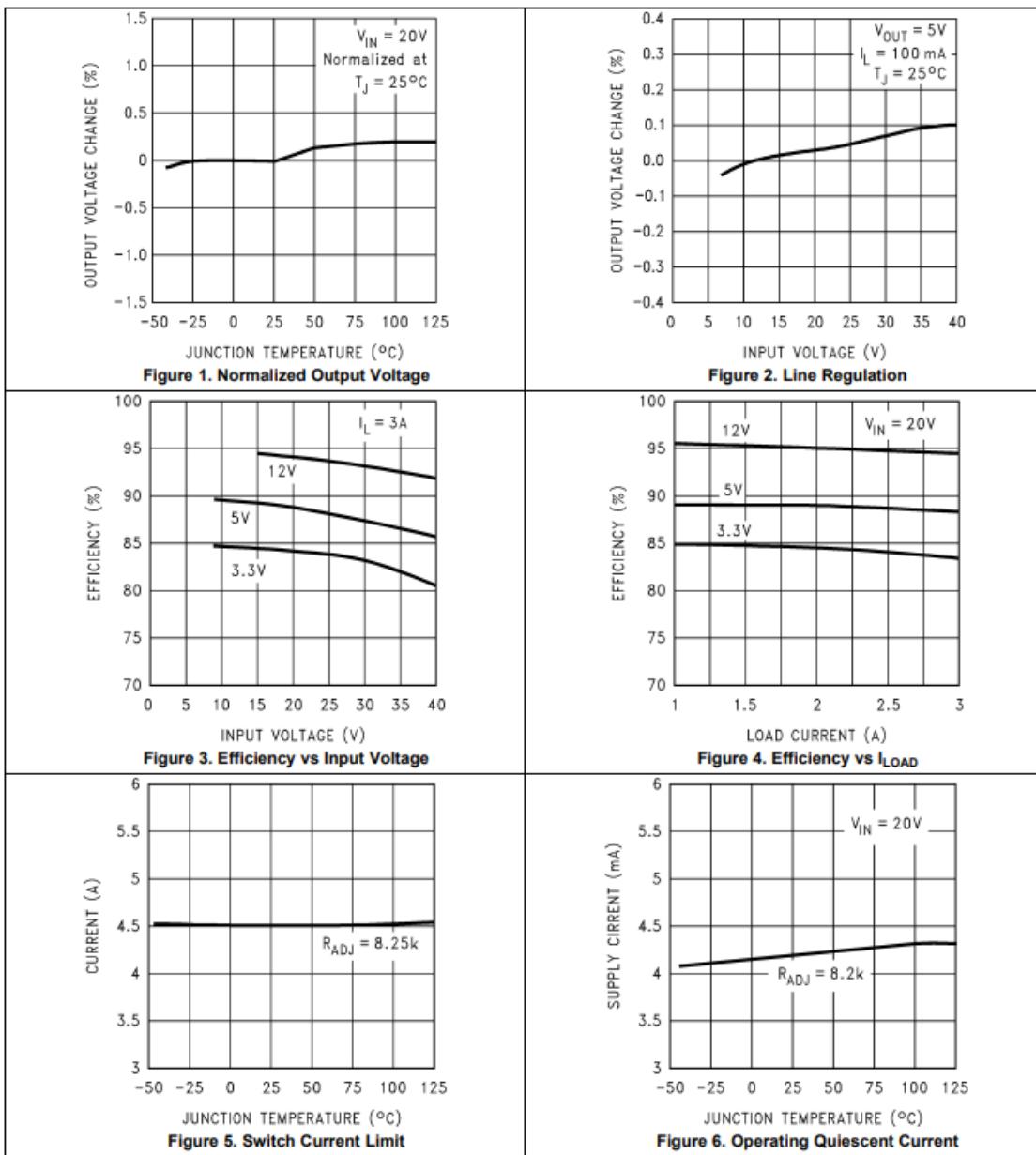
6.9 Electrical Characteristics – All Output Voltage Versions

Specifications are for $T_A = T_J = 25^\circ\text{C}$ unless otherwise specified. Unless otherwise specified $V_{IN} = 12\text{ V}$ for the 3.3-V, 5-V, and Adjustable versions and $V_{IN} = 24\text{ V}$ for the 12-V version.

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
DEVICE PARAMETERS						
I_Q	Quiescent current	$V_{FEEDBACK} = 8\text{ V}$ for 3.3-V, 5-V, and ADJ versions, $V_{FEEDBACK} = 15\text{ V}$ for 12-V versions		4.2	6	mA
V_{ADJ}	Current limit adjust voltage		1.181	1.21	1.229	V
		over the entire junction temperature range of operation -40°C to 125°C	1.169		1.246	
I_{CL}	Current limit	$R_{ADJ} = 5.6\text{ k}\Omega$, ⁽¹⁾	5.5	6.3	7.6	A
		over the entire junction temperature range of operation -40°C to 125°C	5.3		8.1	
I_L	Output leakage current	$V_{IN} = 40\text{ V}$, soft-start pin = 0 V	$V_{SWITCH} = 0\text{ V}$	1	1.5	mA
			$V_{SWITCH} = -1\text{ V}$	6	15	
$R_{DS(ON)}$	Switch ON-resistance	$I_{SWITCH} = 5\text{ A}$		0.12	0.14	Ω
			over the entire junction temperature range of operation -40°C to 125°C		0.225	
f_0	Oscillator frequency	Measured at switch pin		260		kHz
			over the entire junction temperature range of operation -40°C to 125°C	225	280	
D	Duty cycle	Maximum duty cycle		91%		
		Minimum duty cycle		0%		
I_{BIAS}	Feedback bias current	$V_{FEEDBACK} = 1.3\text{ V}$ ADJ version only		85		nA
V_{SFST}	Soft-start threshold voltage			0.63		V
		over the entire junction temperature range of operation -40°C to 125°C		0.53	0.74	
I_{SFST}	Soft-start pin current	Soft-start pin = 0 V		3.7		μA
			over the entire junction temperature range of operation -40°C to 125°C		6.9	

(1) The peak switch current limit is determined by the following relationship: $I_{CL} = 37,125 / R_{ADJ}$.

6.10 Typical Characteristics



Typical Characteristics (continued)

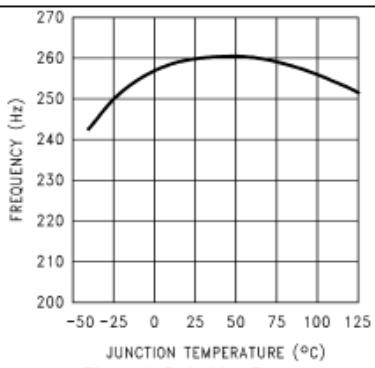


Figure 7. Switching Frequency

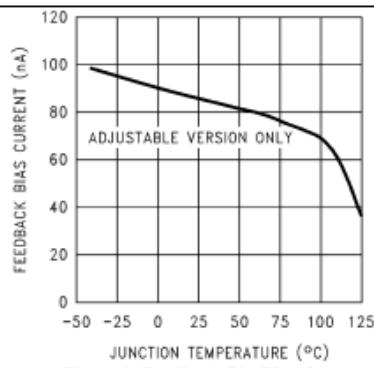
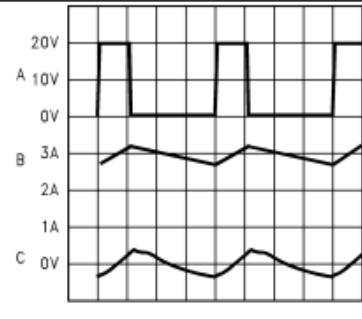


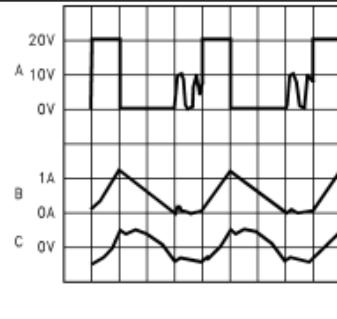
Figure 8. Feedback Pin Bias Current



1 μ sec/Div

Continuous Mode Switching Waveforms $V_{IN} = 20$ V,
 $V_{OUT} = 5$ V, $I_{LOAD} = 3$ A $L = 33 \mu$ H, $C_{OUT} = 200 \mu$ F,
 $C_{OUT,ESR} = 26 m\Omega$
A: V_{SW} Pin Voltage, 10 V/div
B: Inductor Current, 1 A/div
C: Output Ripple Voltage, 20 mV/div AC-Coupled

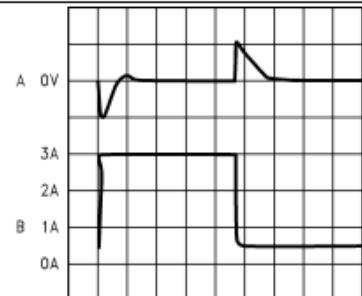
Figure 9. Horizontal Time Base: 1 μ s/div



1 μ sec/Div

Discontinuous Mode Switching Waveforms $V_{IN} = 20$ V,
 $V_{OUT} = 5$ V, $I_{LOAD} = 500$ mA $L = 10 \mu$ H, $C_{OUT} = 400 \mu$ F,
 $C_{OUT,ESR} = 13 m\Omega$
A: V_{SW} Pin Voltage, 10 V/div
B: Inductor Current, 1 A/div
C: Output Ripple Voltage, 20 mV/div AC-Coupled

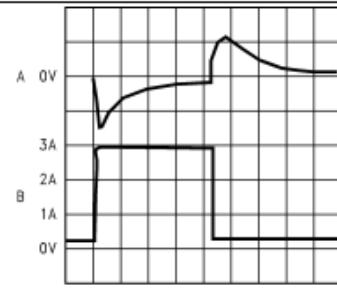
Figure 10. Horizontal Time Base: 1 μ s/div



100 μ sec/Div

Load Transient Response for Continuous Mode $V_{IN} = 20$ V,
 $V_{OUT} = 5$ V $L = 33 \mu$ H, $C_{OUT} = 200 \mu$ F, $C_{OUT,ESR} = 26 m\Omega$
A: Output Voltage, 100 mV/div, AC-Coupled.
B: Load Current: 500-mA to 3-A Load Pulse

Figure 11. Horizontal Time Base: 100 μ s/div



200 μ sec/Div

Load Transient Response for Discontinuous Mode $V_{IN} = 20$ V,
 $V_{OUT} = 5$ V, $L = 10 \mu$ H, $C_{OUT} = 400 \mu$ F, $C_{OUT,ESR} = 13 m\Omega$
A: Output Voltage, 100 mV/div, AC-Coupled.
B: Load Current: 200-mA to 3-A Load Pulse

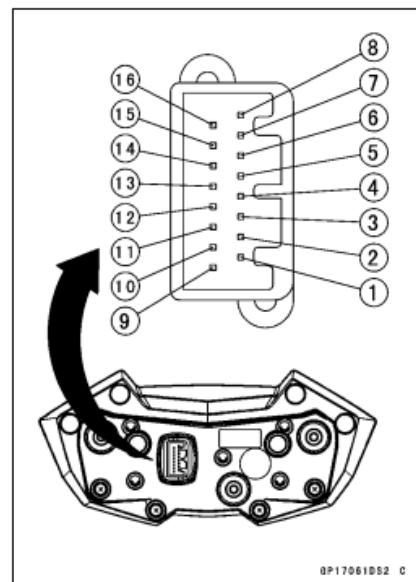
Figure 12. Horizontal Time Base: 200 μ s/div

Fuente: Texas Instruments datasheet.

7.9. Otros documentos

Inspección de la unidad del panel de instrumentos de combinación electrónica

- Extraiga la unidad de instrumentos (consulte Desmontaje/instalación de la unidad de instrumentos).
 - [1] Masa de la luz de aviso (LED) roja (-)
 - [2] Sensor de nivel de combustible
 - [3] No utilizado
 - [4] No utilizado
 - [5] No utilizado
 - [6] Pulso del tacómetro
 - [7] Pulso del sensor de velocidad
 - [8] Masa (-)
 - [9] Luz LED de color verde del indicador del intermitente izquierdo (+)
 - [10] Masa de la luz indicadora (LED) amarilla de ABS (-) (modelos equipados)
 - [11] Tierra de la luz indicadora (LED) verde de punto muerto (-)
 - [12] Testigo (LED) azul de luz de carretera
 - [13] Luz (LED) verde del intermitente derecho (+)
 - [14] Línea de comunicación de la ECU
 - [15] Encendido
 - [16] Batería (+)



Fuente: Manual de taller oficial de motocicleta Kawasaki Z800.

16-20 SISTEMA ELÉCTRICO

Diagrama del cableado (ZR800B sin unidad GPS)

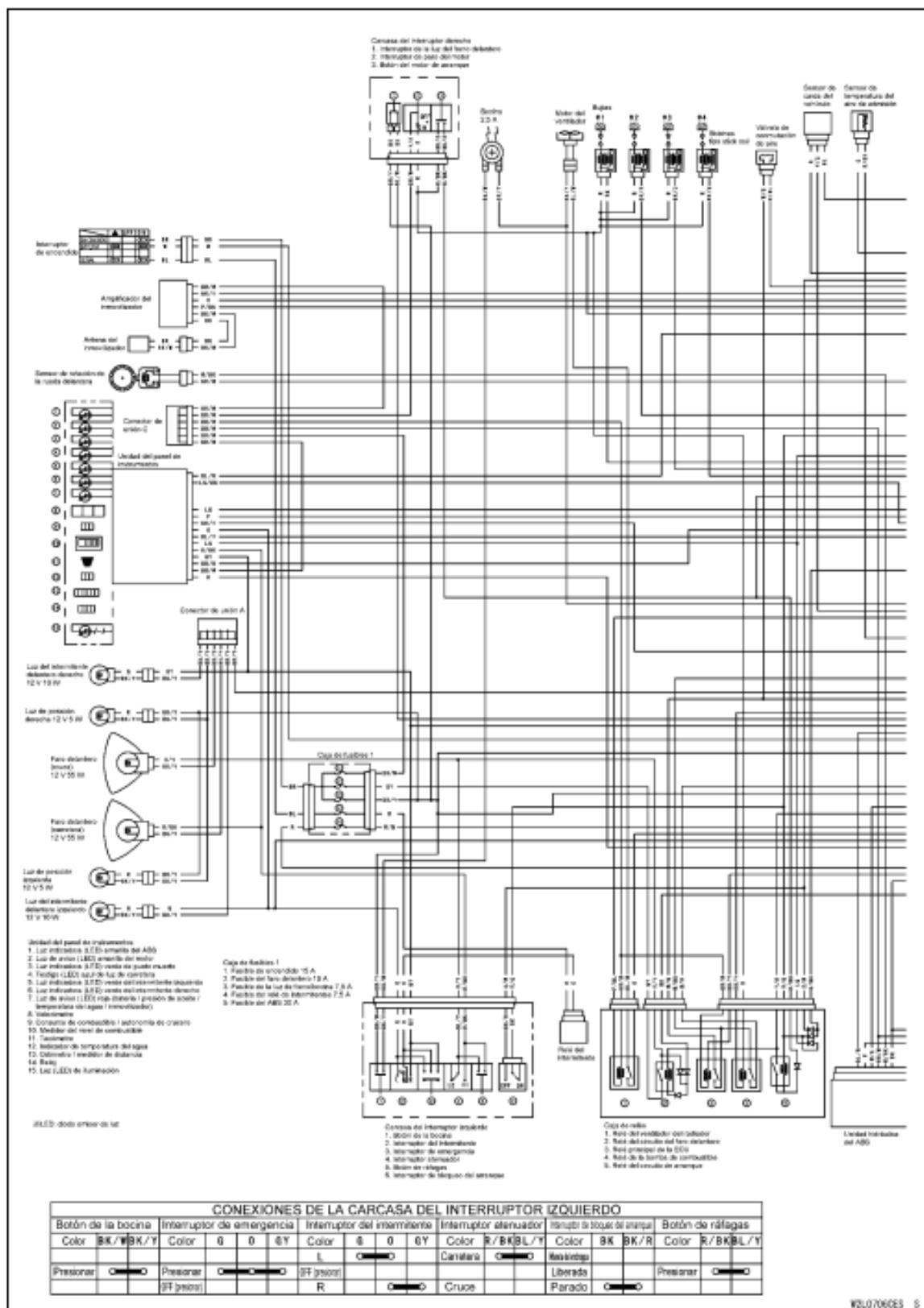
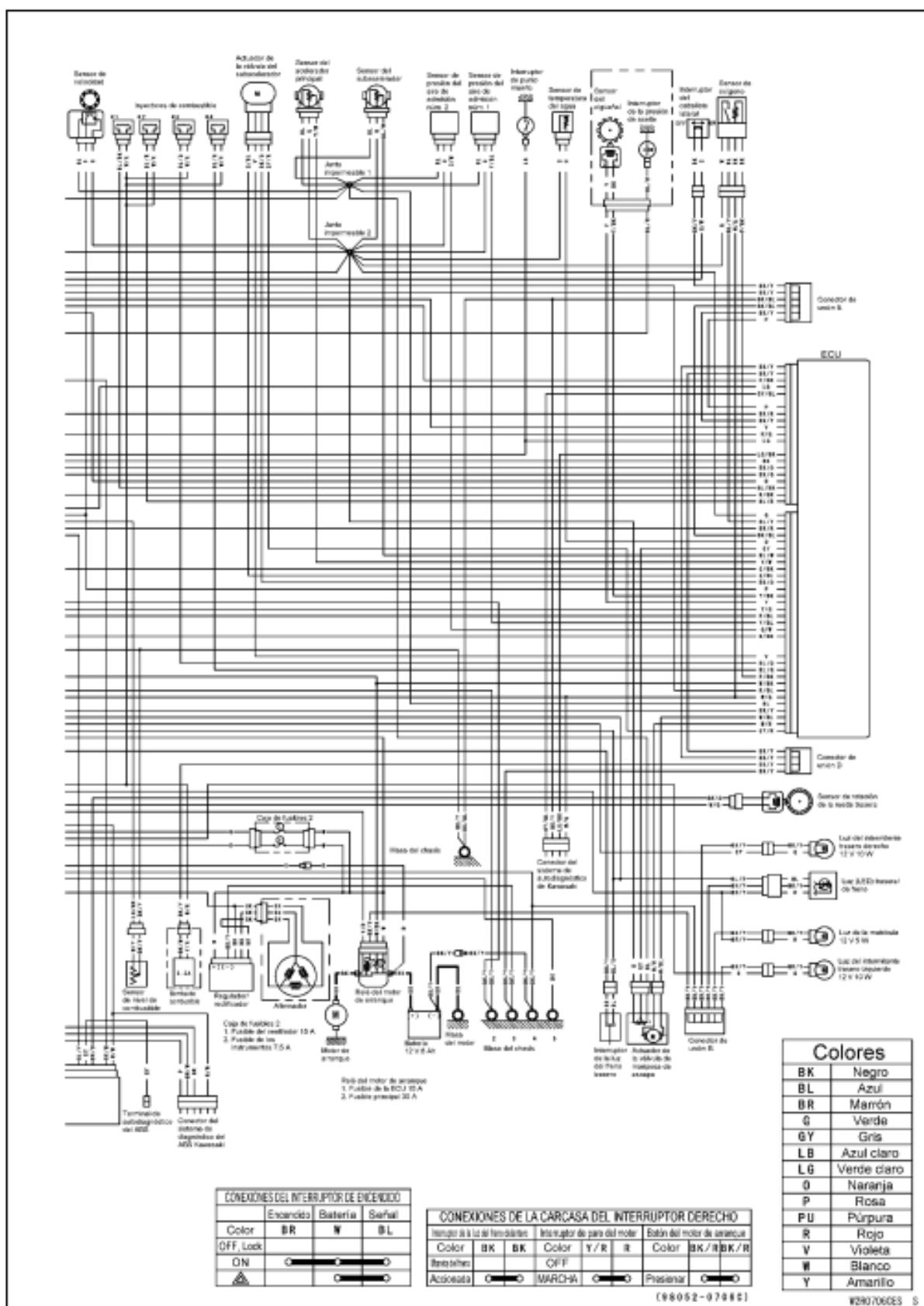


Diagrama del cableado (ZR800B sin unidad GPS)



Fuente: Manual de taller oficial de motocicleta Kawasaki Z800.