Universidad Zaragoza
1542

Trabajo Fin de Grado

ODÓMETRO INTELIGENTE BASADO EN IOT PARA LA NAVEGACIÓN RALLY CON VEHÍCULOS

IoT-based smart odometer for rally navigation with vehicles

Autor

Andrés Mallada Artigas

Director

Roberto Casas Nebra
Álvaro Marco Marco

Ingeniería de Tecnologías y Servicios de Telecomunicación

Escuela de Ingeniería y Arquitectura
2021

Resumen

Las competiciones de orientación mediante roadbook, así como los entrenamientos, son pruebas duras en las que los pilotos se enfrentan a situaciones adversas tales como desorientación, accidentes o averías repentinas; generalmente en medio de campos y caminos alejados de la red de carreteras y núcleos urbanos.

En estas pruebas, los pilotos usan un sistema electrónico llamado tripmaster que les indica la distancia recorrida, así como el rumbo al que se dirigen. De esta manera, siguen secuencialmente las indicaciones de distancia y rumbo de las diferentes viñetas del roadbook para conseguir completar el recorrido.

El objetivo de este proyecto es crear un prototipo de producto totalmente funcional, comparable a los que existen actualmente en el mercado, superándolos tanto en prestaciones como fiabilidad. Se integrará en el odómetro un sistema IoT que reporte datos tanto de ubicación, como de la viñeta actual en la que se encuentra el piloto, la velocidad, el rumbo, etc., así como detección de caídas, y en general, de todos los parámetros de la instrumentación que sirven al piloto para orientarse.

Se valorará también la posibilidad de añadir un módulo de comunicación por satélite, lo que otorgará al sistema una robustez de comunicación en cualquier ubicación en la que se encuentre el piloto, cuando no hubiera cobertura de datos móviles.

La plataforma de visualización de datos permitirá consultar en tiempo real todos los datos reportados por el sistema que lleva equipado el piloto, así como hacer un volcado de los mismos en una tarjeta de memoria para poder analizar los datos posteriormente, algo muy útil para el equipo del piloto.

La plataforma se enfocará en visualizar los datos del prototipo construido; sin embargo, se contempla la opción de una visualización multidispositivo, ideal para eventos y competiciones, en las que llevar un control de todos los pilotos participantes es esencial para el correcto desarrollo de la prueba, y garantizar así tanto la seguridad de los pilotos como asegurar una prueba limpia y sin trampas.

Se buscará también la compatibilidad y versatilidad con el resto de equipos que existen actualmente en el mercado, así como con los diferentes periféricos que los integran (mandos de control, porta-roadbook, etc.), facilitando así la instalación en la moto y mejorando la experiencia de usuario al no tener que prescindir del equipo que ya usan por adquirir este nuevo sistema.

Índice

Lista de figuras

1. Introducción

1.1. Contexto

Las competiciones de vehículos campo a través se remontan a 1979 cuando se disputó el primer Dakar Rally (París-Argel-Dakar). Desde sus inicios, la competición ya incluía categoría de coches y de motocicletas aunque, con el paso de los años, fueron apareciendo nuevas modalidades llegando a incluir actualmente categorías de camiones, quads, etc.

Dado que se trata de un deporte muy extendido mundialmente y la participación en el Dakar Rally solo está abierta a pilotos profesionales con experiencia (por motivos de seguridad), surgen competiciones amateurs de ámbito nacional e internacional por todo el mundo. En España y alrededores se organizan anualmente la conocida Baja Aragón, la 1000 Dunas, la Basella Race, etc. todas ellas abiertas tanto a pilotos amateurs como profesionales, a los que les sirve de entrenamiento para adquirir experiencia y renombre.

Sin embargo, se trata de un deporte muy exigente que requiere mucho entrenamiento aparte del que se realiza en competiciones y, por ello, los pilotos y aficionados deben entrenar por su cuenta. Este entrenamiento se lleva a cabo muchas veces en solitario, por lo que en ocasiones puede entrañar un gran riesgo ya que es una actividad que produce mucho desgaste físico, además de averías mecánicas y accidentes que pueden surgir durante la conducción. Añadiendo a todo esto que las rutas y recorridos suelen realizarse en su mayor parte por caminos secundarios, monte y campos, por lo que los pilotos, en caso de accidente, pueden verse en una situación de grave peligro.

El desafío de esta disciplina de deporte se encuentra no solo en el manejo del vehículo en entornos extremos, sino también en la orientación y la navegación mediante la instrumentación que se le proporciona al piloto.[1]

Figura 1: Ejemplo de roadbook desenrollado.

La navegación se lleva a cabo mediante el denominado "**roadbook**". Se trata de un rollo de papel continuo en el cual están impresas una serie de viñetas que detallan el recorrido a seguir por el piloto, indicando tanto el rumbo como la distancia a recorrer en cada tramo; puede incluir a su vez símbolos y elementos visuales que sirvan de referencia para la orientación.

[1] El equipamiento básico de un piloto se explica detalladamente en el apartado 2.1.

1.2. Objetivos

De entre todas las categorías de vehículos mencionadas anteriormente este proyecto se centra en las motocicletas. Se trata de la categoría más extendida entre los pilotos amateurs por la fácil adquisición de vehículos y el respectivo equipamiento.
A su vez se trata de la modalidad más peligrosa pues el piloto va al descubierto sin un chasis o estructura que lo proteja, quedando así mucho más expuesto a los peligros y obstáculos del recorrido.

Figura 2: Porta-roadbook instalado en una motocicleta.

Debido a ese riesgo al que se ven expuestos los pilotos, surge la idea de este proyecto: conseguir integrar en los instrumentos de navegación que equipan los vehículos, un sistema de telemetría que permita el seguimiento del recorrido del piloto a través de una plataforma web, así como de diferentes parámetros relevantes que puedan garantizar su seguridad. Por lo tanto, en base a las exigencias y características requeridas, este proyecto se basará en la tecnología del **Internet de las cosas (IoT).**

El principal **objetivo** de este proyecto es desarrollar un sistema que permita poder seguir en tiempo real la ruta que el piloto esté llevando, así como la detección de caídas, velocidad y el resto de los parámetros que se decidan sensorizar. De esta manera se podrán visualizar en una sola pantalla los datos de la instrumentación que el piloto está utilizando para navegar, así como su ubicación en el mapa.

Los actuales sistemas de navegación comerciales no permiten la extracción de los datos que reportan para su tratamiento externo. Por ello es por lo que se decide realizar íntegramente un dispositivo que presente todas las funcionalidades de los sistemas comerciales actuales, y que incluya, además, el sistema de telemetría y reporte de datos expuesto en el párrafo anterior.

Los escenarios de aplicación son muy variados. Este dispositivo permitirá realizar un seguimiento a los pilotos que deciden entrenar en solitario, asegurando así que siempre van a estar localizados y en caso de incidencia podrán ser socorridos. También se puede usar como herramienta para la organización y gestión de carreras amateurs y quedadas entre aficionados, llevando control de todos los participantes que porten el dispositivo en su motocicleta, y asegurando así un transcurso seguro de la competición; incluso llegando a proporcionar a los espectadores una forma de observar el transcurso del evento y mejorar así la experiencia.

Se trata además de una herramienta que puede ayudar a la conservación de caminos y terrenos naturales en este tipo de eventos, pues se conocerá la ubicación exacta de los participantes y el recorrido que han seguido, pudiendo penalizar e incluso llegar a denunciar a las autoridades a aquellos que se salgan de las rutas establecidas.

Debemos destacar que, aunque se haya pensado y optimizado el sistema para su uso en motocicletas, el sistema que se va a implementar es compatible con el resto de los vehículos ya que la instrumentación y los métodos de navegación son similares en todos ellos.

1.3. Planificación y procedimiento

El proyecto se dividió en tres fases: hardware, firmware y software; abarcando así las diferentes ramas de la ingeniería de telecomunicaciones, incluyendo conceptos de algunas de las materias cursadas a lo largo del grado.

En la primera fase del proyecto se abordó todo lo relacionado con el **hardware** del prototipo. Esta fase incluyó la búsqueda y selección de componentes, esquemático del circuito electrónico, diseño de PCBs y soldadura y ensamblaje del mismo.

Uno de los objetivos de este proyecto es maximizar la compatibilidad con el resto de productos actualmente en el mercado. Para ello se realizó un estudio de las tecnologías, estándares, dimensiones y especificaciones de los productos comerciales ya existentes, incluyendo en dicho estudio una evaluación de los diferentes periféricos que requieren estos dispositivos: receptor GPS, mando de control, etc.; así como los elementos de instalación y conexionado: estructuras, soportes, cableado, conectores etc.

Además de diseñar y construir una versión propia tanto del dispositivo principal como de dichos periféricos, se fabricó una carcasa totalmente adaptada a las características del prototipo y sus módulos. Todos estos elementos se diseñaron mediante el software CAD Shapr3D y se fabricaron a través de una impresora 3D (Artillery X1 Sidewinder).

Una vez se tuvo el hardware implementado y funcionando correctamente, se procedió a programar el **firmware** que corre el microprocesador del dispositivo. Como se detalla más adelante, se decidió basar el sistema en un microprocesador ESP32. Para su programación se utilizó PlatformIO sobre el IDE de Visual Studio Code.

La parte **software** de este proyecto se corresponde con la plataforma web para la visualización de datos. Para ello se empleó la herramienta Thingsboard diseñada para la gestión de dispositivos, la recepción de datos, su tratamiento y visualización mediante el protocolo MQTT, muyutilizado en el ámbito del IoT. El objetivo principal de esta fase era crear la parte correspondiente al frontend del servidor, diseñando un panel de visualización y creandocreando las funciones de procesado de datos, ya que el backend del servidor ya se encuentra desarrollado por la propia plataforma.

2. Estado del arte. Elección de tecnología.

2.1. Equipamiento básico de un piloto

Como ya se ha comentado antes, la orientación en las competiciones y entrenamientos de esta disciplina de deporte se realiza mediante instrumentos digitales denominados odómetros y compases, que muestran la información en tiempo real sobre la distancia recorrida y el rumbo (ángulo que forma la moto con respecto al norte geográfico).

Aparte del odómetro y del compás (y sus periféricos necesarios) los pilotos llevan en la moto una caja estanca denominada **"porta-roadbook"** que, mediante un interruptor basculante en el manillar, permite al piloto controlar el sentido de giro de un pequeño motor reductor eléctrico y, por consiguiente, el avance y retroceso del roadbook. De esta manera el piloto puede visualizar las diferentes viñetas que definen la ruta a seguir.

Hay que destacar que en competiciones oficiales como el Dakar, los vehículos incluyen algunos sistemas extra que aseguran la integridad y seguridad del piloto, como es el caso del **"Sentinel"**, que detecta a competidores cercanos a través de radiofrecuencia, y emite una señal acústica y luminosa en caso de adelantamiento. También equipan un dispositivo de comunicación por satélite para detectar y socorrer situaciones de emergencia o accidentes.

Este aparato recibe el nombre de **"Iritrack"** y funciona a través de la red de satélites Iridium. En las competiciones, es la propia organización quien proporciona el instrumento ya que tiene un elevado coste y no es especialmente útil para los pilotos adquirir uno a nivel personal.

Figura 3: Iritrack de la marca Marlink.

Figura 4: Kit de equipamiento básico f2r (1300€).

Es por ello por lo que, para este proyecto, se considerará tan solo el equipamiento básico [2] del piloto y se incorporará al prototipo un sistema que desempeñe el papel básico del Iritrack. De esta forma, tanto en entrenamientos como en pequeñas competiciones amateurs, los pilotos podrán equipar un sistema asequible de seguridad y alerta.

[2] Porta-roadbook eléctrico, mandos de control, odómetros, antena GPS y distribuidor de alimentación.

También conviene destacar un tercer dispositivo que sirve a la organización de la competición para validar que los pilotos no hacen trampas y siguen las rutas navegando por ellos mismos y no fijándose en las marcas y rodadas que otros pilotos hayan podido dejar a su paso.

Al organizar una carrera, se establecen unos puntos de control (waypoints) por los que el piloto deberá pasar para dar por válida la ruta realizada. Si bien la navegación hasta dichos puntos debe ser indicada al piloto para que los valide, la navegación por roadbook es indispensable para realizar el recorrido completo, pues de orientarse únicamente con las indicaciones hasta los waypoints, no lograría realizar una ruta válida completa. Este aparato desempeña el papel de brújula hasta los puntos de control estratégicos distribuidos a lo largo del recorrido. Dicho guiado se implementará también en el prototipo.

Como en el caso del Iritrack, no es un dispositivo que suelan equipar los pilotos amateurs ya que, en caso de requerir algún tipo de validación del recorrido seguido, utilizan navegadores y trackers GPS convencionales para la posterior comprobación de la ruta seguida.

Figura 5: Garmin GPS Montana 680 (579€)

2.2. Tipos de odómetros

De todo el equipamiento que puede llevar un piloto en su vehículo, el instrumento más importante e indispensable es el odómetro; aunque la mayoría de ellos realizan más funciones que marcar distancias y pueden llegar a considerarse ordenadores a bordo. En la actualidad existen numerosas marcas que fabrican y comercializan instrumentación de este tipo. Si bien cada una incorpora características propias a sus productos y periféricos, existen dos tipos principales de odómetros: los basados en sensor hall y los basados en la recepción de señal GPS.

2.2.1. Basados en sensor hall

Se tratan de odómetros que funcionan de forma similar a un velocímetro común de una bicicleta. Se coloca un sensor hall y un imán en la rueda delantera de la moto y se introduce en el dispositivo el valor del radio al que se ha colocado el sensor. El aparato detecta cada vuelta completa de la rueda y realiza los cálculos pertinentes de distancia y velocidad.

Este tipo de ordenadores a bordo son los más simples y baratos y sólo ofrecen la siguiente información: distancia recorrida total, distancia recorrida parcial (el piloto puede ajustarla y reiniciarla), velocidad y tiempo de trayecto. A parte del dispositivo principal, requiere de un sensor hall y un mando externo para controlar y navegar desde el manillar por los diferentes menús. Marcas como f2r comercializa este tipo de dispositivo en torno a los 325€ incluyendo los periféricos.

2.2.2. Basados en receptor GPS

Este tipo es el más extendido por la precisión y prestaciones que ofrece y es sobre el que basaremos nuestro prototipo. El ordenador a bordo recibe las tramas GPS a través de un módulo receptor externo (por separado, la marca f2r comercializa el módulo por 89€) y realiza los cálculos de distancia en base a los datos que obtiene.

Este tipo de odómetros, además de todas las funcionalidades que proporciona el dispositivo basado en sensor hall, incluye también el rumbo que lleva el vehículo y un reloj sincronizado a través de los datos GPS. Además, la precisión que ofrece este sistema es notablemente superior a la del dispositivo anterior; en el sistema basado en sensor hall existen mayores desajustes entre la distancia indicada y la recorrida realmente, ocasionados por las variaciones en el diámetro del neumático, derrapes, etc.

Para corregir dichos desajustes, pues en mayor o menor medida los dos sistemas pueden tener errores de medición, ambos sistemas incorporan la función de **recalado,** que permite al piloto ajustar la distancia parcial para que cuadre con las indicaciones del roadbook a través de los botones del mando.

El distribuidor f2r comentado en el anterior apartado comercializa este dispositivo con todos los periféricos por 425€.

El fabricante de ambos dispositivos y periféricos es la empresa americana ICO Racing, que encabeza el oligopolio internacional de estos instrumentos.

2.2.3. Prototipo propuesto

Tras un análisis e investigación de los diferentes productos que hay en el mercado, se detectó que, salvo para competiciones oficiales, los instrumentos que existen para competidores y pilotos amateurs no incluyen sistemas de seguridad y son escasos en cuanto a prestaciones para el elevado precio que tienen.

Es por ello por lo que se decide realizar un dispositivo basado en GPS (no necesariamente con el módulo comercial de 89€), compatible con cualquier receptor GPS serie programable (para fijar tasa de baudios y frecuencia de refresco) y que incorpore además un sistema de detección y aviso de caídas, además de pantallas e interfaces más completas y con valores promedio, datos de la ubicación actual, guiado hasta los waypoints de control en caso de que se requieran y, como característica adicional, un detector de la viñeta del roadbook en la que se encuentra el piloto a través de un módulo de infrarrojos del que posteriormente detallaremos su diseño.

Todo ello integrado en un único dispositivo que incorpore el envío de mensajes a través del protocolo MQTT mediante un punto de acceso WiFi creado por el smartphone del piloto, descartando así la opción de comunicación a través de la red de satélites Iridium por su elevado coste y su tarificación.

También se incluirá el hardware necesario para implementar una interfaz CAN-BUS y, de esta manera, poder conectar el dispositivo al puerto de diagnóstico OBD-II[3] del vehículo y tener la posibilidad de reportar también los datos de la ECU[4] interna del vehículo (revoluciones, nivel de combustible, temperatura del motor, averías…).

A continuación se muestra una tabla comparativa de las prestaciones de los principales dispositivos del mercado junto con el prototipo propuesto en este proyecto.

	ICO Rallye MAX 2	ICO Rallye MAX-G	Smart CAP (Prototipo TFG)
Tecnología	Sensor Hall	GPS (5 Hz)	GPS (5 Hz)
Pantalla	LCD 7 Segmentos	LCD 7 Segmentos	LCD Matriz de puntos (128x64)
Retroiluminación	Sí	Sí	Sí
Alimentación	12 V	12 V	12V / 24V / USB 5V
Batería	Interna de emergencia	No	Externa a través de micro-USB
Mandos compatibles	Sí	Sí	Sí
Compatible soportes estándar	Sí	Sí	Sí
Odómetro parcial ajustable	Sí	Sí	Sí
Odómetro distnacia total	Sí	Sí	Sí
Indicador velocidad	Actual/Máxima	Actual/Máxima	Actual/ Máxima/ Promedio
Indicador Rumbo (CAP)	No	Sí	Sí
Reloj	No	Sí	Sí
Tiempo de trayecto	Tiempo acumulado	Tiempo acumulado	Cronómetro controlable
Unidades	km/millas	km/millas	km/millas
Idioma	Inglés	Inglés	Español, Inglés, Francés, Portugués, Alemán, Catalán
Indicador ubicación actual	No	No	Sí
Guiado waypoints	No	No	Sí
Log de ruta seguida	No	No	Sí
Telemetría IoT	No	No	Sí
Conexión CAN-BUS	No	No	Sí
Visualización simultánea Dist-CAP	No	No	Sí
Detector de viñeta actual	No	No	Sí
Detección de caída y alerta	No	No	Sí
Precio	325,00 €	425,00 €	-

Figura 6: Tabla comparativa de los principales dispositivos.

Se puede apreciar como el prototipo del proyecto, bautizado como "**Smart CAP**", supera notablemente en prestaciones al resto de dispositivos.

Al tratarse de un prototipo no se ha establecido un precio de venta, sin embargo, en el BOM (Anexo I) puede verse el coste de todos los componentes empleados. También se ha diseñado una versión propia tanto de los mandos de control como del receptor GPS debido al elevado precio individual de estos. La versión diseñada de ambos resulta totalmente compatible con el resto de dispositivos comerciales.

Figura 7: Logotipo diseñado para el producto.

[3] OBD-II: On Board Diagnostics version 2
[4] ECU: Engine Control Unit

3. Diseño del hardware

3.1. Descripción de bloques funcionales

3.1.1. Bloque de alimentación

El sistema se ha diseñado para alimentarse a partir de los 12 voltios de la batería de la propia motocicleta en la que se instale el dispositivo. Esta es la fuente de alimentación que usan todos los odómetros comerciales actualmente en el mercado, por lo que se consideró la principal opción, descartando el uso de una batería interna recargable como se pudo plantear en un principio. El hecho de que se alimente directamente de la batería proporciona compatibilidad con las instalaciones y cableado que los pilotos ya equipan en sus motos, por lo que no se necesita ninguna adaptación o instalación extra ya que se ha diseñado con el conector M8 de 3 pines típico de estos instrumentos, ampliamente utilizado en el ámbito de la instrumentación electrónica.

Sin embargo, dado que era necesario incluir una interfaz USB para programar el microcontrolador y realizar las tareas de debug, el sistema se ha diseñado para ser alimentado también con 5 voltios a través de un conector micro-USB. Esto resulta especialmente útil para instalaciones rápidas y temporales o pruebas de funcionamiento. Además, lo hace muy versátil y adaptable, incluso para entrenamientos en bicicleta, un deporte muy común entre las rutinas de acondicionamiento físico de los pilotos.

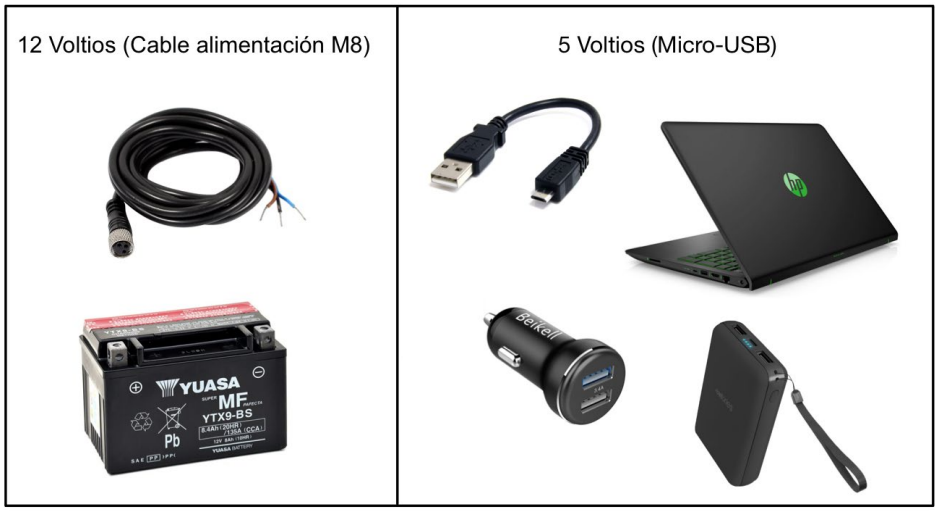


Figura 8: Ejemplos de las diferentes opciones para alimentar el sistema.

Para unificar ambas vías de alimentación, se ha optado por utilizar un convertidor DC-DC conmutado para pasar de 12V a 5V. De esta manera, tanto si se alimenta a través de la batería de la moto como si se opta por la alimentación mediante micro-USB, estaremos trabajando con una tensión de 5 voltios en ambos casos. Finalmente, a través de un LDO, reducimos esos 5V a 3.3V, con los que alimentamos el MCU y el

resto de los componentes del sistema[5]. Al reducir el voltaje de trabajo a 3.3V reducimos en consecuencia el consumo energético. En nuestro caso, el consumo no es un aspecto crítico para el dispositivo pues la alimentación es externa y no dependemos de ninguna batería interna, en cuyo caso sí que habría que tener un exhaustivo control del consumo en los diferentes modos de operación para optimizar su duración. Sin embargo, se tratará de reducir dicho consumo para evitar degradar la batería del vehículo.

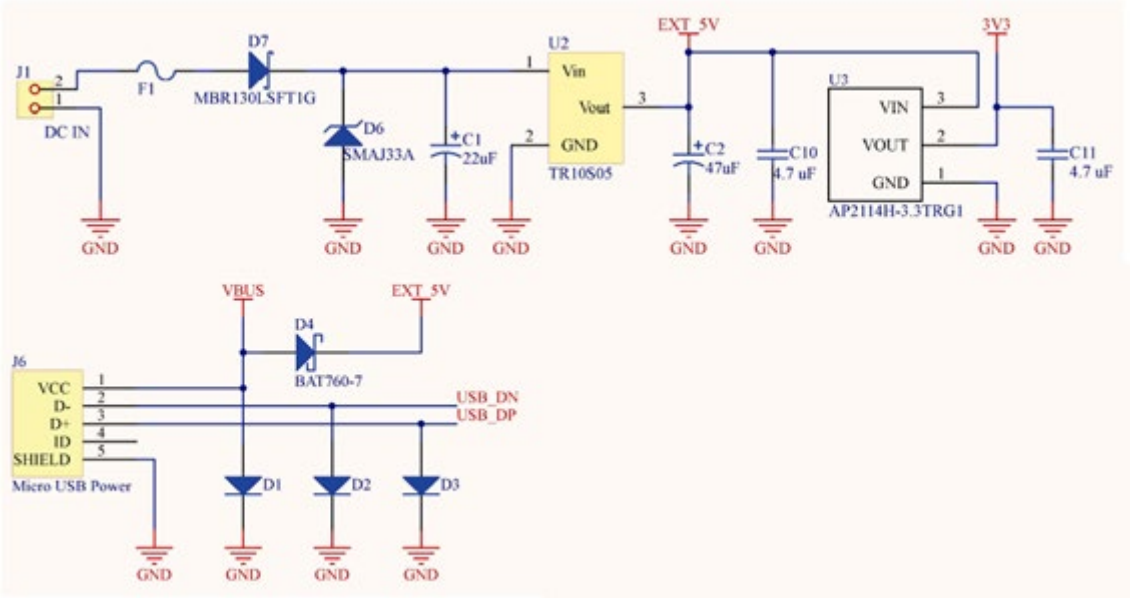


Figura 9: Esquemático del bloque de alimentación.

En la Figura 9 se muestra el circuito planteado para la etapa de alimentación del sistema. A continuación, procedemos a detallar los principales componentes que la integran y la función que desempeñan.

En primer lugar, encontramos los componentes F1, D7 y D6. Se corresponden con los **elementos de protección** del sistema. El fusible F1 se trata de un **fusible PPTC**[6]. Este tipo de fusibles son comúnmente usados en este tipo circuitos electrónicos dado que son rearmables. "*Cuando un exceso de corriente pasa por ese componente se calienta aumentando su resistencia protegiendo así a otros componentes de la sobreintensidad. Cuando cesa la corriente el componente se enfría y vuelve a su estado inicial de baja resistencia, es decir, se rearma.*" (Wikipedia, 2020). Esta característica lo hace idóneo para nuestro aparato pues no será necesario abrir el dispositivo para reemplazarlo si se produce una subida de corriente. El valor de corriente máxima se ha limitado a 750 mA.

Por otro lado, el diodo D7 es necesario para proteger el circuito ante una posible **polarización inversa**. Es un caso muy común que podría darse, por ejemplo, al invertir los cables de alimentación en la conexión a la batería de la moto. De esta manera, no se permite el paso de corriente y se protege así el resto del circuito.

[5] A excepción de la pantalla LCD, la cual se alimenta a los 5 voltios de salida del convertidor conmutado.
[6] PPTC: Polymeric Positive Temperature Coefficient.

El diodo D6 es un diodo TVS[7] . Su función principal es proteger el resto del circuito de **picos de alta tensión** (spikes) recortando el voltaje a valores seguros dentro del rango de operación del circuito. Este componente es necesario en la etapa puesto que la batería de una moto (y la de cualquier vehículo), se encuentra conectada a numerosos componentes y actuadores que forman parte de la electrónica y la mecánica del propio vehículo: alternador, motor de arranque, bobina, bujías, etc. Estos elementos pueden provocar elevados picos de tensión, sobre todo en la secuencia de arranque del motor y pueden llegar a dañar el circuito si no se protege adecuadamente. Se ha escogido para que en caso de producirse estos picos los recorte a 33 V.

Seguido a los elementos de protección encontramos el **convertidor DC-DC conmutado** U2 mencionado anteriormente, que reducirá el voltaje de los 12 V de la batería a 5 V. El motivo de usar un convertidor conmutado para reducir la tensión es porque al ser necesaria una reducción de voltaje notable, es la solución más eficiente en cuanto a términos energéticos se refiere (90%).

Seguidamente el **regulador lineal o LDO**[8] U3 se encarga de reducir esos 5 V a los 3.3 V que usaremos en el resto del circuito. En este caso, sí que tiene sentido emplear un regulador lineal para este propósito pues la diferencia de tensiones es reducida y por tanto la energía a disipar no es excesiva.

Finalmente destacamos el diodo D4, que permite la distribución de los 5 V del bus USB a la entrada del LDO en caso de que la alimentación se lleve a cabo mediante USB, y bloquea el paso de corriente proveniente de la etapa de entrada de 12 V para proteger el bus USB al que se encuentre conectado el dispositivo en caso de que se optara por ambas vías de alimentación.

3.1.2. Microcontrolador

En el mercado actual existen varios microcontroladores que podrían ser atractivos para usar en este proyecto. Dado que se trata de un sistema IoT, acotamos la búsqueda a MCUs que se utilicen en este contexto. Un chip bastante utilizado es el ESP8266 de Esspresif Systems. Sin embargo, existe un nuevo modelo, el **ESP32**, un MCU dual core del cual existen encapsulados que integran conectividad WiFi y Bluetooth mediante una antena PCB. Por sus reducidas dimensiones y su gran versatilidad nos decantamos por este chip.

Figura 10: Módulo ESP32-WROOM-32

[7] TVS: Transient Voltaje Supressor. Conocido popularmente como “transil”.
[8] LDO: Low-Dropout Regulator

En la siguiente tabla se pueden ver las prestaciones de cada uno de los microcontroladores.

Características	ESP8266	ESP32	Características	ESP8266	ESP32
Procesador	Tensilica LX106	Tensilica Xtensa X36	Hardware / Software PWM	No / 8	ene-16
Nº bits	32 bits				
Nº nucleos	Single core	Dual core	ADC	1 (10 bits)	18 (12 bits)
Velocidad	80Mhz (hasta 160 Mhz)	160 MHz (hasta 240 MHz)	ADC con preamplificador	No	Sí (bajo ruido 60 dB)
SRAM	160 kB	512 kB			
SPI FLASH	Hasta 16MiB		DAC	No	2 (8 bits)
Alimentación	3.0 a 3.6V	2.2 a 3.6V	UART	2 (en una sólo puede usarse pin TX)	4
Rango de temperaturas	-40ºC a 125ºC		I2C	1	2
Consumo de corriente	80 mA (promedio), 225 mA (máximo)		SPI	2	4
			I2S	1	2
Consumo en deep sleep	20 uA (RTC + memoria RTC)	2.5 uA (10 uA RTC + memoria RTC)	1-Wire	Implementado por software	
			CAN BUS	No	1 x 2.0
Consumo en bajo consumo	No	Inferior a 150 uA	Sensor Touch	No	10
			Sensor temperatura	No	Sí
Wifi	802.11 b/g/n (hasta +20 dBm) WEP, WPA				
Soft-AP	Sí		Sensor HALL	No	Sí
Encriptación por hardware	No (TLS 1.2 por software)	Sì	IR	Sí	
			Timers	3	4 (64 bits)
Bluetooth	No	v4.2 BR/EDR + BLE	Gen. Núm. Aleatorios	No	Sí
Ethernet MAC Interface	No	10/100 Mbps			
			Encriptación de la flash	No	Sí
GPIO (utilizables)	17	36			
			Arranque seguro	No	Sí

Figura 11: Tabla comparativa entre el ESP32 y el ESP8266

(LLamas, 2018)

El principal motivo de elección del ESP32 es que presenta más del doble de GPIO que el ESP8266, y el superior número de buses integrados que presenta entre ellos I2C, SPI y CAN; a parte del resto de prestaciones que lo hacen notablemente superior.

En la Figura 12 se pueden observar las conexiones a cada uno de los GPIO disponibles, incluyendo los pines IO0 y EN, que se describen en el siguiente apartado y servirán para poner al chip en modo de auto programación.

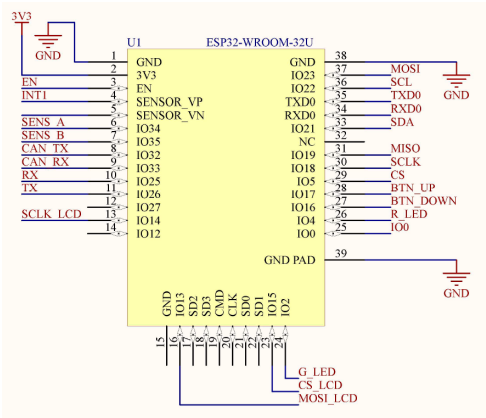


Figura 12: Esquemático de conexiones del ESP32

3.1.3. Bloque de programación. Conversión USB-UART

Existen varios módulos de desarrollo que integran el ESP32 y los circuitos externos necesarios para simplificar las labores de desarrollo, y esta fue la opción escogida para realizar las pruebas de funcionamiento del sistema en protoboard antes de diseñar la PCB del sistema. Dado que son módulos básicos completamente funcionales, se trató replicar el bloque de programación que integran estas placas de desarrollo, en concreto el DevKit-C. Esta idea se descartó finalmente porque los componentes necesarios estaban descatalogados y existían otras opciones más atractivas.

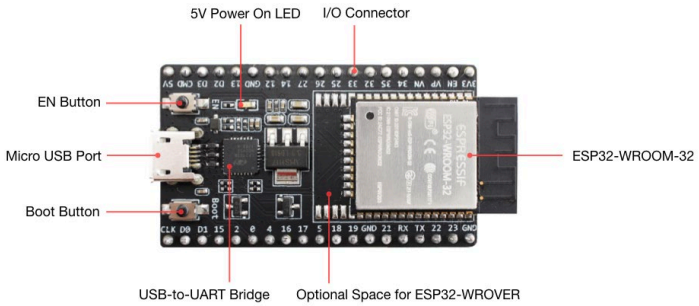


Figura 13: Kit de desarrollo ESP32-DevKitC

El chip FT232R de FTDI es un conocido **bridge USB** de uso muy extendido; su controlador para el reconocimiento del dispositivo en el ordenador no da problemas de compatibilidad y viene ya integrado en la mayoría de los equipos. Incorpora las salidas digitales (RTS y DTR) que nos servirán para activar la auto programación en el ESP32 mediante una función lógica a través de los pines IO0 y EN mencionados anteriormente.

La función principal de este bloque es realizar la conversión de las órdenes del compilador que se transmiten por USB al protocolo de comunicación serie UART; de esta forma se podrá programar el firmware en el ESP32 y, en caso de que sea necesario, tener comunicación en tiempo real con él para la depuración de código.

En un primer momento se pensó en realizar la programación y flasheo del ESP32 mediante un conversor USB-UART externo. Sin embargo, se decidió incorporar el **bridge USB** en la propia placa del sistema como se ha descrito anteriormente. De esta forma se facilitan las tareas de debug, programación, y futuras actualizaciones del firmware para el usuario final. No obstante, a modo de contingencia, se ha incluido un conector para que en caso de fallo de la etapa de conversión podamos programar el MCU con un conversor externo.

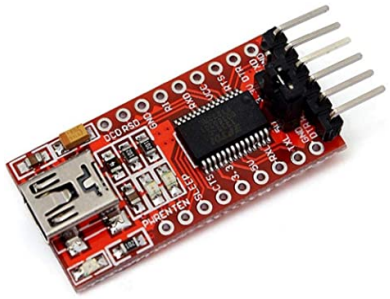


Figura 14: Conversor externo USB-UART TTL basado en FT232R.

La Figura 15 detalla el esquemático del circuito planteado para el bloque de programación.

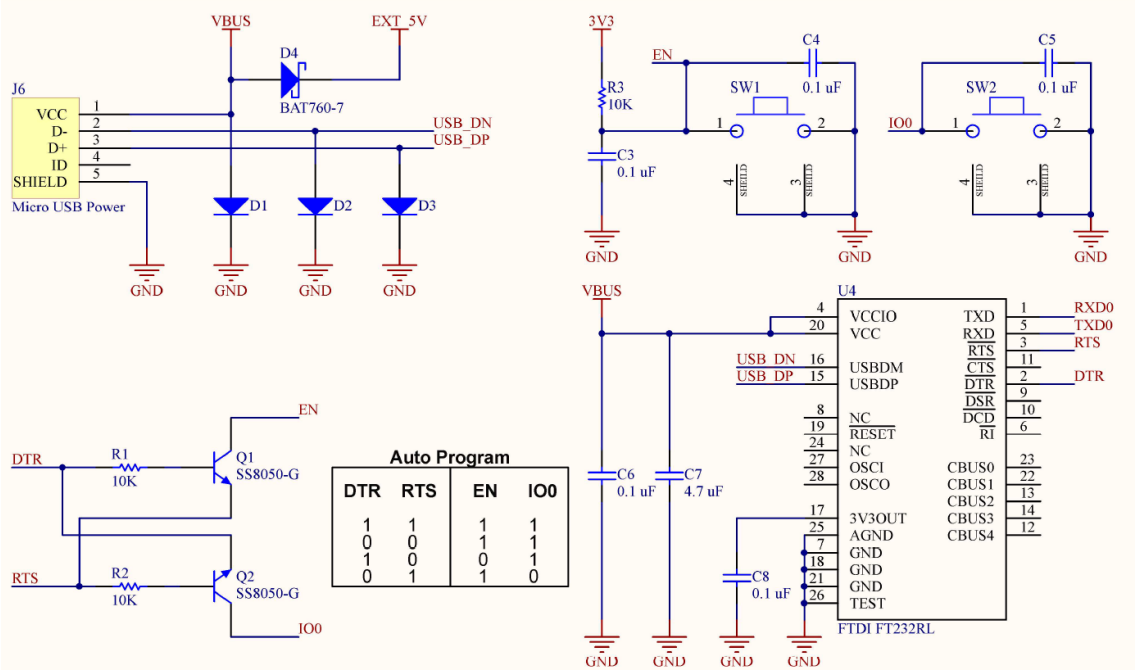


Figura 15: Esquemático del bloque de programación.

Como podemos observar, se incluyen 3 diodos de protección (D1, D2 y D3), a parte del ya mencionado D4, para proteger el bus USB ante la posibilidad de circulación de corriente en sentido inverso al bus.

El chip U4 es el FTDI232RL que se encarga de realizar la conversión de USB a UART. Para ello son necesarios ciertos componentes pasivos de polarización (C6, C7 y C8) que especifica el fabricante en el datasheet, cuya interconexión es la correspondiente a la configuración para alimentar el chip a través del propio bus USB, ya que dependiendo de cómo se quiera alimentar el chip y que uso se le va a dar, en el datasheet se nos indican diferentes topologías de polarización.

Los pulsadores SW1 y SW2 sirven para reiniciar el MCU e iniciar la secuencia de auto programación respectivamente. Si bien esto puede hacerse manualmente presionando dichos pulsadores, el bridge USB escogido presenta dos salidas digitales (RTS y DTR) que mediante un simple circuito con los transistores Q1 y Q2 podemos utilizar para generar las salidas función lógica que aparece detallada en la tabla de verdad “Auto Program” de la figura anterior.

3.1.4. Sensores y periféricos

Como se ha detallado en el apartado 1.2. de la memoria, el principal objetivo de este proyecto es crear un prototipo totalmente funcional que incluya todas las características de los dispositivos comerciales pero que, además, aporte funcionalidades extra que aseguren la seguridad del piloto y permitan seguir remotamente el trayecto de este. Por lo que se requerirán módulos y periféricos adicionales a los que equipan los aparatos comerciales.

En primer lugar, se diseña un **módulo de infrarrojos** externo para detectar el patrón impreso en el lateral del roadbook y poder contabilizar así el avance o retroceso de las viñetas. Podría considerarse un encoder lineal, en el cual hacen falta dos sensores para poder detectar las dos direcciones de desplazamiento. En el apartado 4.1.2 se detalla cómo se lleva a cabo dicha detección.

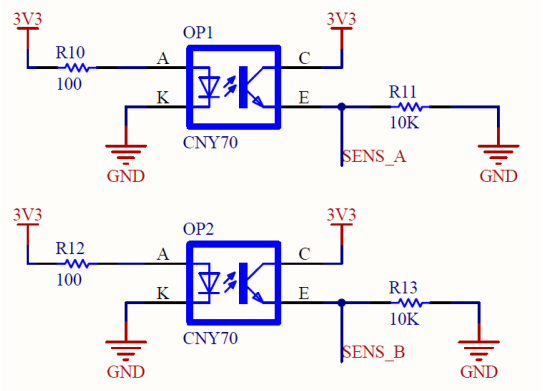
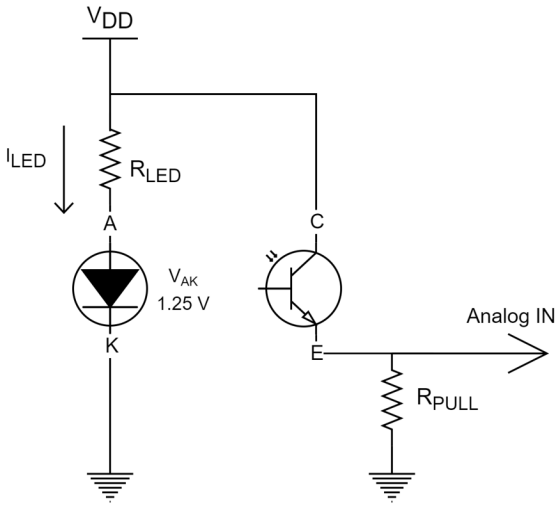


Figura 16: Esquemático del módulo IR.

Los componentes escogidos para este propósito se tratan de dos sensores de barrera infrarroja CNY-70 que constan de un LED emisor de infrarrojos y un fototransistor en el mismo encapsulado.

El principio de funcionamiento es sencillo: el LED emite radiación infrarroja que es reflejada por el obstáculo hacia el que esté dirigido el haz, el fototransistor recibe esa reflexión en mayor o menor medida en función de las características del obstáculo.

Para polarizar el fototransistor existen dos topologías posibles: pull-up o pull-down. En este caso escogemos la primera cuyo funcionamiento se traduce en que, a más luz captada por el fototransistor, mayor será el voltaje de salida en el punto de medida indicado. Dicha salida será leída mediante una entrada analógica con la que posteriormente se realizarán los cálculos pertinentes. Los cálculos realizados, así como el circuito propuesto para la polarización del CNY70 se detallan a continuación.



$$I_{LED} = \frac{V_{DD} - V_{AK}}{R_{LED}} < 50\ mA$$

$$R_{LED} > \frac{3.3 - 1.25}{50 \cdot 10^{-3}} = 41\ \Omega$$

$$R_{LED\ min} = 41\ \Omega$$

$$R_{LED|I_{LED}\,=\,20\ mA} = 102.5\ \Omega$$

$$\boxed{R_{LED} = 100\ \Omega \ \Rightarrow \ I_{LED} = 20.5\ mA}$$

Figura 17: Circuito y cálculos para la polarización de los CNY70.

Como se puede observar, se escoge un valor para R_{LED} de 100 Ω por ser este el valor comercial más cercano al resultado obtenido en los cálculos, el cual aseguraba una corriente a través del LED infrarrojo de 20 mA, valor más que suficiente para nuestro propósito y notablemente inferior al valor máximo indicado en el datasheet.

Para R_{PULL} se escogió un valor de 10 kΩ por ser este un valor comúnmente utilizado como resistencia pull-up para la lectura de sensores de este tipo a través de un GPIO.

Para poder detectar caídas y accidentes se decide incorporar un **acelerómetro**. Tras una búsqueda sobre los distintos tipos y las prestaciones de estos nos decantamos por un acelerómetro de salida digital, en concreto que integre la interfaz I2C para poder conectarlo a los GPIOs que implementan dicho bus en el ESP32.

Figura 18: Módulo GY-91

En un primer momento, se planteó usar el magnetómetro que suele venir integrado en el mismo chipset de los acelerómetros para obtener el rumbo que lleva el vehículo a través del campo magnético de la Tierra. Se realizaron pruebas con el módulo GY-91 que integra el chip MPU-9250 (acelerómetro y magnetómetro) y el barómetro BMP280 que podría resultar útil para obtener la altitud de manera más precisa. Sin embargo, dado que el magnetómetro no era muy preciso y no lograba obtenerse un rumbo válido estable, se decidió descartar esta opción.

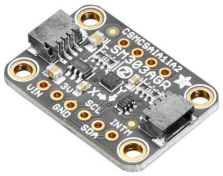


Figura 19: Módulo LSM303AGR

Por ello se buscó un chipset que tuviera mayor precisión y prestaciones; los denominados e-compass eran los candidatos perfectos, utilizados en smartphones, gafas de realidad aumentada, etc. por su precisión y sus 9 grados de libertad. Se decide utilizar el **LSM303AGR** de STMicroelectronics y se adquiere un módulo que lo integra del fabricante Adafruit para realizar pruebas de funcionamiento. Obtenemos unos resultados mejores que con el MPU-9250 a la hora de estimar el rumbo; sin embargo, no es suficiente para realizar un compás fiable debido a las interferencias magnéticas a las que está sometido el sensor por el resto de electrónica del sistema. A pesar de ello, se acabó utilizando este sensor pues es de mayor calidad y precisión que el anterior.

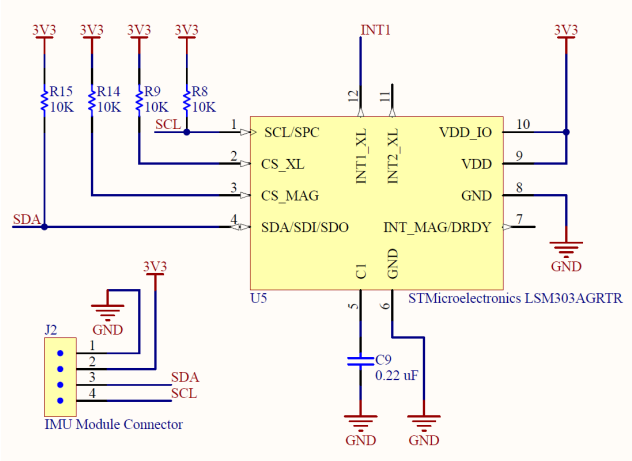


Figura 20: Esquemático de conexión del e-compass.

Atendiendo a las indicaciones del fabricante, se conectaron los pines CS del acelerómetro y del magnetómetro a los niveles lógicos correspondientes para poner en funcionamiento ambos sensores, y se conectaron los pines SDA y SCL al bus I2C con sus respectivas resistencias de pull-up. Se escogió un valor de 10 kΩ pues será el único dispositivo del bus I2C. Se conectó también el pin de interrupciones del acelerómetro por si pudiera ser de interés en algún momento.

Se debe destacar que el encapsulado que presentan este tipo de componentes no es soldable a mano, ya que los pads se encuentran en la cara inferior del chip y son muy pequeños para reducir al máximo los efectos parásitos. Por ello se decide plantear el circuito y la huella en la PCB por si en un futuro se llegara a soldar, pero a su vez dejar espacio para un conector al que poder conectar el módulo con el que se habían realizado las pruebas directamente. Esto, además, dota de una mayor versatilidad al

dispositivo, ya que podría funcionar con cualquier módulo que integrara un acelerómetro que funcione a través de I2C pues finalmente se descartó el uso del magnetómetro como compás.

También se incluyó un **transceptor CAN-BUS** que, aunque no se ha desarrollado para este proyecto, podrá utilizarse para conectarse a la centralita del vehículo y leer los datos como se ha comentado en apartados anteriores.

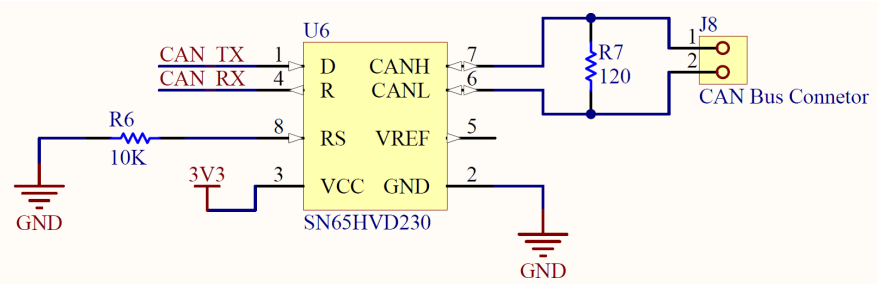


Figura 21: Esquemático de conexión transceptor CAN-BUS

Para poder guardar e introducir datos en el sistema se añade un conector para una tarjeta **microSD** y se conecta al bus SPI. No se requieren componentes adicionales pues los niveles lógicos del MCU y de la SD coinciden (3.3V).

Finalmente, para mostrar los datos y visualizar los diferentes menús del dispositivo, escogemos una **pantalla LCD** de matriz de puntos de 128 x 64 pixeles, en lugar de los displays de 7 segmentos que traen los odómetros convencionales. Esto nos da mucha más versatilidad a la hora de mostrar los menús, animaciones, textos, etc.

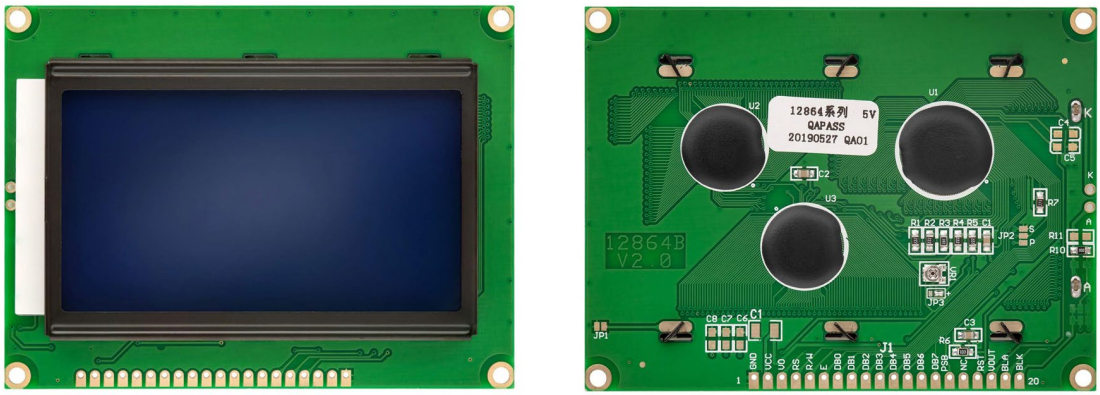


Figura 22: Módulo de pantalla LCD 128x64 de AZ-Delivery

En un primer momento se contempló la idea de adquirir la pantalla como componente aislado y diseñar el circuito de control según especificara el fabricante. Sin embargo, este proceso iba a resultar laborioso y económicamente no salía rentable pues se encontraron módulos de pantalla ya ensamblados sobre una PCB que incluía toda la electrónica necesaria para el manejo de la pantalla a través de puerto paralelo o el bus SPI a precios muy asequibles (10€).

3.1.5. Receptor GPS

Como se ha detallado en el apartado 2.1 uno de los periféricos básicos que debe integrar nuestro producto es el **receptor de señal GPS**. Se tuvo acceso a uno de los GPS comerciales que equipan actualmente los odómetros de la marca ICO Racing y se decidió realizar un trabajo de ingeniería inversa. Se contactó con el fabricante americano para solicitar el pinout de los cables de conexión (en este caso M8 de 4 pines) para asegurarse de los voltajes de trabajo y no dañar el receptor. Se alimentó y se conectó a un conversor de puerto serie a USB (como el de la Figura 14) para identificar el tipo de mensajes y la tasa de transmisión que empleaba.

Tras este análisis, se concluyó que el módulo GPS comercial se alimentaba a 3.3 V y se comunicaba a través de la interfaz serie (UART) a 9600 baudios; utilizaba una tasa de refresco de 5 Hz, y sólo reportaba una de las múltiples tramas NMEA[9] que reportan los receptores GPS, en concreto la trama GPRMC[10].

```
$GPRMC,220516,A,5133.82,N,00042.24,W,173.8,231.8,130694,004.2,W*70
        1    2    3    4    5     6    7    8      9     10  11 12


1   220516     Time Stamp
2   A          validity - A-ok, V-invalid
3   5133.82    current Latitude
4   N          North/South
5   00042.24   current Longitude
6   W          East/West
7   173.8      Speed in knots
8   231.8      True course
9   130694     Date Stamp
10  004.2      Variation
11  W          East/West
12  *70        checksum
```

Figura 23: Ejemplo y contenido de una trama GPRMC.

En vista de las especificaciones obtenidas, y descartando el uso del receptor que comercializa la propia marca (ICO Racing) por su elevado precio, se decide adquirir y adaptar un módulo GPS pensado para la navegación con drones.

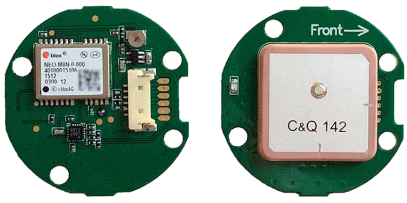


Figura 24: Vista interior del módulo GPS.

Este tipo de receptores son idóneos para tareas de geolocalización básicas ya que integran una antena cerámica en el propio módulo y tienen una precisión más que aceptable. Se buscaron módulos que integraran chips del fabricante uBlox debido a su buena reputación en soluciones de geolocalización, IoT, etc. En concreto nos decantamos por el modelo NEO-8M, ya que se trata del chip integrado más novedoso, versátil y con mejor calidad/precio de la marca.

[9] National Marine Electronics Association. Organización de comercio electrónico estadounidense que establece estándares de comunicación para electrónica marina, receptores GPS y otros instrumentos.
[10] GPRMC: Datos de tránsito /GPS específicos mínimos recomendados.

Finalmente se adquiere un módulo externo que incorpora toda la electrónica necesaria para el correcto funcionamiento del chip. Esta decisión se basa en ahorrar espacio en la PCB principal del prototipo y en maximizar la compatibilidad entre equipos, ya que, al ser externo, cualquier módulo GPS programable con comunicación serie serviría. Además, al estar encapsulado en una carcasa resistente como se puede ver en la Figura 25, resulta idóneo para su montaje en la moto.

Figura 25: Vista exterior del módulo GPS.

Una vez adquirido el módulo se reprogramó mediante la herramienta **"u-Center"**, la cual proporciona el propio fabricante y permite ajustar y definir las tasas y parámetros de configuración ya que los ajustes con los que viene programado por defecto no eran los adecuados. Finalmente, se cableó con el conector M8 anteriormente mencionado atendiendo al pinout facilitado por el fabricante y de esta manera tener además un módulo totalmente compatible con sus dispositivos, pero a un precio notablemente inferior (12€).

3.1.6. Mando de control

En cuanto al **mando** para controlar el dispositivo, en vista del elevado precio del que comercializa la marca líder, se decide construir uno propio buscando la compatibilidad con los dispositivos comerciales. Para ello se realizaron una vez más labores de ingeniería inversa (dado que los dispositivos y periféricos están sellados con silicona y resulta imposible acceder a la circuitería) para averiguar su funcionamiento interno, y fabricar uno propio totalmente compatible; y de esta forma garantizar que el prototipo podría funcionar con los mandos comerciales y viceversa.

Figura 26: Mando de control de la marca ICO Racing (65€).

Tras varias pruebas con el multímetro y el osciloscopio, y montar diferentes versiones del circuito candidato en una protoboard, conseguimos describir el comportamiento de los botones del mando con el circuito de la Figura 27.

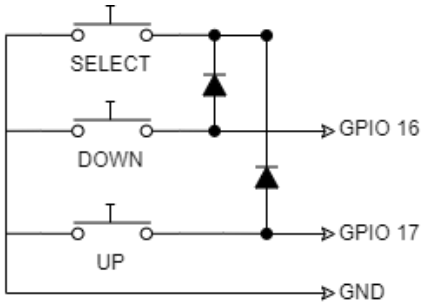


Figura 27: Circuito propuesto para el mando de control.

Como se puede observar, el mando consta de 3 botones: arriba, abajo y seleccionar. Sin embargo, el conector que emplean presenta tan solo 3 pines. Lo más sencillo y lógico hubiera sido conectar uno de los terminales de cada botón a un GPIO diferente y conectar a tierra el otro terminal de todos ellos, pero esta topología implicaría usar 4 cables.

Se observó que uno de los botones (seleccionar), cortocircuitaba los pines correspondientes al botón arriba y al botón abajo simultáneamente, cómo si se pulsaran a la vez. De esta manera, el circuito se realiza con tan solo 3 cables, pero requiere de un par de diodos para no conectar permanentemente entre sí los terminales del botón arriba y abajo. Dado que los GPIOs escogidos pueden configurarse como entrada pull-up, no se han incluido dichas resistencias en el circuito.

Se debe destacar que, en el mando diseñado, no se incluye el interruptor basculante que controla el avance del roadbook dado que el control del porta-roadbook no se contempla dentro de este proyecto; incluirlo no supondría mayor complicación que reservar espacio en la carcasa y realizar un sencillo circuito inversor de polaridad para controlar el motor.

Figura 28: Mando f2r con interruptor para control del roadbook (220 €).

3.2. Diseño de la PCB

Una vez estructurados todos los bloques y elementos por los cuales está compuesto el sistema y haber planteado los circuitos e interconexiones de los mismos, se procedió a diseñar la PCB. Para ello se utilizó la herramienta CircuitMaker, un programa gratuito basado en Altium cuyas prestaciones son más que suficientes para el propósito de este proyecto.

En primer lugar, se buscaron todos los componentes que integraban nuestro dispositivo dentro de la base de datos del programa asegurandose de que tuvieran asociada una huella correcta y el modelo 3D del componente, que serviría de referencia para el posterior diseño de la carcasa.

El siguiente paso, una vez escogidos todos los componentes y realizado el esquemático de interconexión, fue ubicar los componentes en la PCB y dimensionar la placa. Para ello se tomaron de referencia las dimensiones del módulo de la pantalla que se montaría mediante tornillos sobre la placa principal. Por lo tanto, las dimensiones se ajustaron al máximo a las de la pantalla para hacer el aparato lo más compacto posible.

Una vez se definieron la forma y límites de la placa, se procedió a la ubicación de componentes. Se siguió la topología de “soles y satélites” que se basa en primero ubicar los componentes principales (microprocesador, conversores, bridge USB…) y en torno a ellos ubicar el resto de elementos que se requieren para su funcionamiento (componentes pasivos, transistores, conectores, etc.). Se debe tener en cuenta que los conectores, LED, y pantalla, se tuvieron que ubicar en posiciones acordes con la carcasa que se iba a diseñar, teniendo en cuenta la ubicación del dispositivo en la moto, para que fueran visibles y accesibles.

También hay que destacar que al usar el ESP32 con una antena PCB para la conectividad WiFi y Bluetooth, debemos dejar libre de componentes y pistas la zona contigua a ella. Por tanto, como se puede observar en la Figura 30, en la parte superior de la placa la **zona de la antena** queda libre (en **morado**) y el plano de masa termina en ella tanto en la cara superior como la inferior.

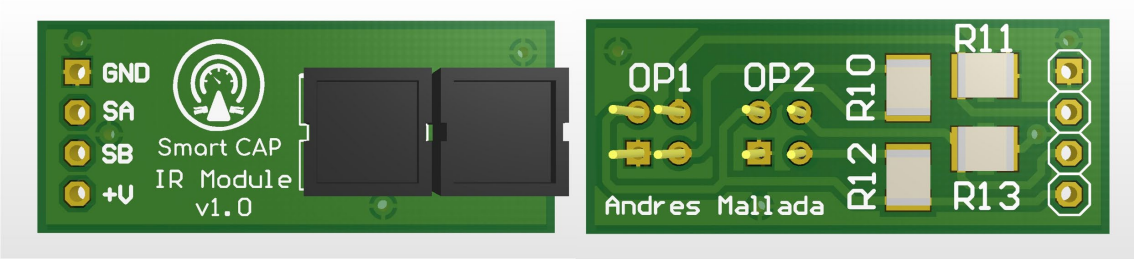


Figura 29: Vista superior e inferior del modelo 3D de la PCB del módulo IR.

Para la placa del módulo infrarrojo el proceso de diseño resultó más simple debido al reducido número de componentes como se puede apreciar en la Figura 29. Dado que el módulo iría ubicado encima del portaroadbook para poder alinearlo con el patrón impreso, se minimizó al máximo la superficie de la placa procurando que molestara lo menos posible al piloto cuando mirara las viñetas del roadbook. Se intentó routear solamente por la capa superior pues al tratarse de una placa muy pequeña, se pensó fabricarla en la fresadora que tiene la universidad que solo routea PCBs por una cara. Sin embargo, debido a los reducidos costes de fabricación en China, finalmente se decidió mandar a fabricar junto con la PCB principal.

La distribución de componentes en la PCB principal se realizó en función de los bloques descritos en el apartado 3.1. Se han resataldo con recuadros de colores dichos bloques para su mejor identificación.

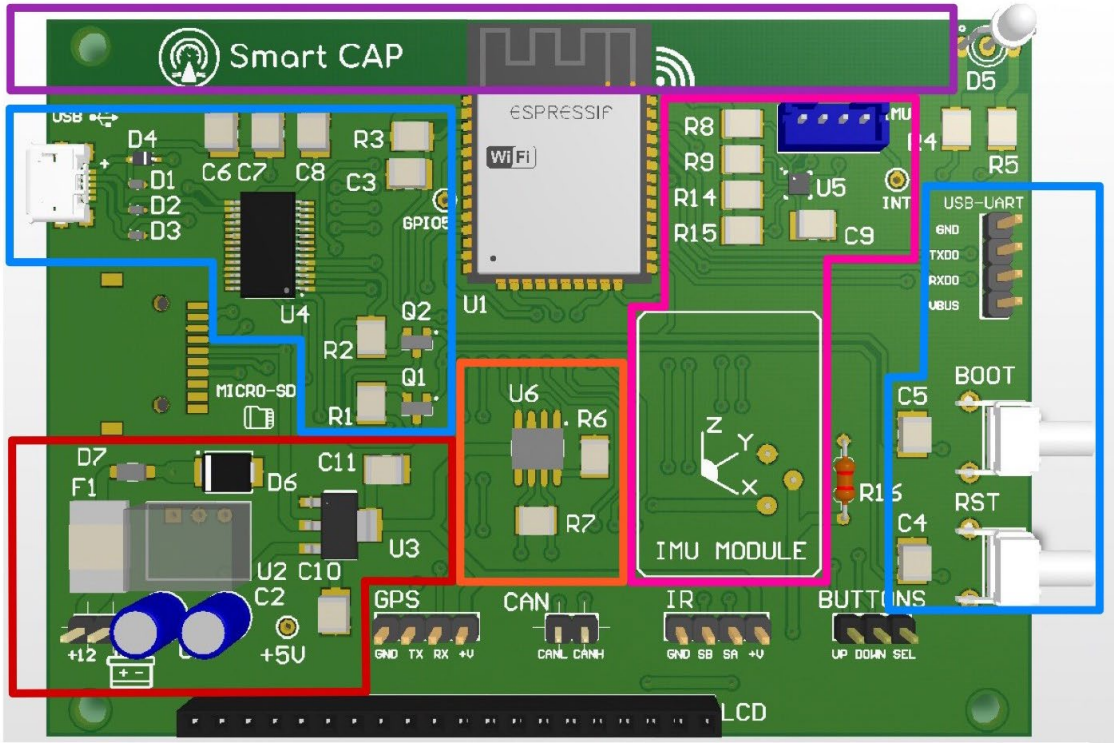


Figura 30: Distribución por bloques de la PCB principal.

Lo primero que se aprecia es la posición central superior del **microprocesador**. Se ubicó en el centro atendiendo a la topología de soles y satélites, siendo el microprocesador el sol principal de nuestro sistema, y se colocó en la parte superior para mejorar la propagación y minimizar interferencias de la señal del WiFi y Bluetooth.

En color **rojo** se encuentra el **bloque de alimentación** en la parte inferior izquierda. La ubicación estaba restringida debido a que los conectores de alimentación y de los diferentes periféricos debían ubicarse alienados y en la parte inferior para ser compatibles con los montajes y soportes estándar. Se puede apreciar como los soles del bloque son U2 y U3, que se corresponden con el regulador conmutado y el regulador lineal respectivamente.

Los recuadros de color **azul** se corresponden con el **bloque de programación.** En la parte izquierda se ubica el sol de este bloque, el bridge USB (U4). La ubicación viene restringida también debido al conector microUSB que requería estar situado en algún borde de la placa. Aprovechando su ubicación se situó el conector microSD contiguo a él, para dar así homogeneidad a la distribución de los conectores de entrada y salida.

En la parte derecha se aprecian recuadrados también en **azul** el conector para el conversor USB-UART externo (en caso de que fuera necesario), y los botones de reset y boot, que sirven para reiniciar y poner en modo programación el dispositivo respectivamente. Sin embargo, dado que la función lógica a través de los transistores Q1 y Q2 funcionó correctamente, y resetear el microprocesador no es una funcionalidad necesaria para el usuario, finalmente no se soldaron dichos botones pues se pusieron como plan de contingencia en caso de que diera problemas el bloque de programación.

Recuadrado en color **rosa** se encuentra el **bloque del acelerómetro;** el sol de este bloque es el integrado U5. No obstante, como ya se ha comentado antes, este chip y sus componentes satélites no se soldaron debido a la dificultad que entrañaba para los medios que se tenían, por lo que se soldó un conector y se reservó una zona libre para colocar el módulo externo adherido a la placa.

Finalmente, en color **naranja** se identifica el **bloque del transceptor CAN-BUS,** formado por el integrado U6 y sus respectivas resistencias.

El resto de los componentes no se han marcado por no tener especial relevancia. El LED bicolor D5 se situó en la parte superior de la placa por ser este el mejor lugar de visualización una vez colocadas la carcasa y la pantalla, cuyo conector se encuentra en la parte inferior por ser ahí donde se encontraban los pines de conexión en el módulo del display LCD.

En la Figura 31 se muestran los diferentes conectores y los respectivos pinouts para cada uno de los periféricos que no están integrados en la placa principal. Todos ellos utilizan el conector M8 mencionado en apartados anteriores. La asignación de los conectores macho o hembra se realizó atendiendo a la de los sistemas comerciales de

forma que no pudieran intercambiarse entre sí. De esta manera se asegura que cada periférico se puede conectar únicamente al conector para el que había sido diseñado. La única excepción es el conector CAN-BUS que utiliza el mismo conector que el mando, pero dado que en este proyecto no se implementará, no presenta mayor problema.

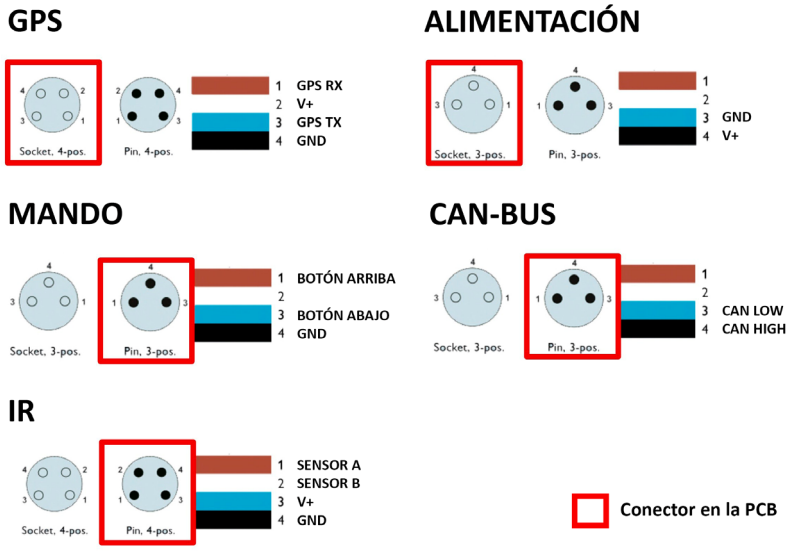


Figura 31: Resumen de conectores y pinout de cada periférico.

3.3. Fabricación, soldadura y ensamblaje

Una vez diseñada la PCB se exportaron los gerber de ambas placas y se mandaron fabricar a la empresa china JLCPCB. Se escogieron características estándar para la placa pues se trataba de un prototipo: dos capas, dielectrico FR4 de 1.6mm de espesor y pads estañados para facilitar la soldadura. Se especificó que el acabado de la superficie cumpliera el certificado RoHs Free (sin plomo).

Para soldar los componentes se utilizó tanto un cautín convencional como soldadura por aire caliente debido a que todos los componentes escogidos eran de montaje superficial (SMD). No obstante el encapsulado de algunos de ellos, así como la reducida distancia entre pines, hizo que se tuviera que pedir ayuda a los maestros de taller para asegurar su correcta soldura.

Para los conectores M8 se tomó la decisión de no soldarlos directamente a la PCB. Al ir conectados a diferentes elementos externos estarán sometidos a movimientos bruscos, esfuerzos de tracción, etc. por el tipo de actividad a la que está destinado su uso. Por ello, para no someter a un fuerte estrés mecánico las soldaduras y el resto del hardware, se adquirieron los diferentes conectores con un segmento de cable ya crimpado industrialmente como muestra la Figura 32.

Figura 32: Cables con conector M8.

Una vez soldados todos los componentes descritos con anterioriad, el resultado fue el que se muestra en la Figura 33. Destacar la ausecia de los botones y del acelerómetro como ya se ha indicado, así como la del potenciómetro para el control del contraste de la pantalla, del cual se prescindió y se utilizó en su lugar un valor fijo de resistencia par la cual se había reservado espacio (R16). A la interfaz CAN no se le cableó ningún conector pues no se llegó a implementar.

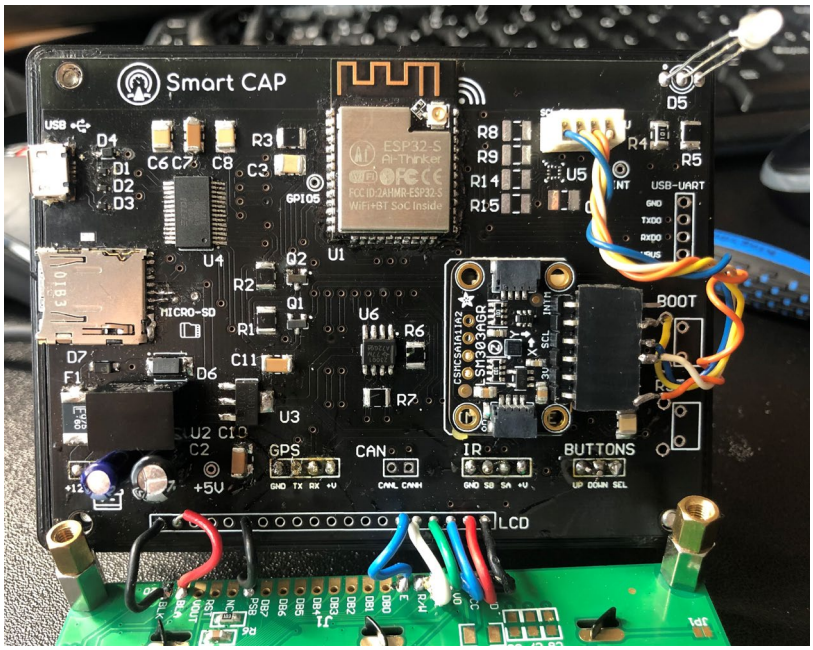


Figura 33: PCB principal del prototipo ya soldada.

En la Figura 34 podemos observar el resultado tras soldar los componentes y el cable del módulo infrarrojo.

Figura 34: PCB módulo IR ya soldada.

Para el mando de control no se empleó PCB alguna y se soldaron los componentes entre sí mediante cables. Se utilizaron botones de montaje de panel resistentes al polvo y agua debido al entorno en el que serán usados. Una vez soldados los diodos e interconectados los los botones siguiendo el esquemático de la Figura 27, se aisló todo el conjunto con silicona para evitar roturas y desconexiones, y asegurar así la estanqueidad del mando.

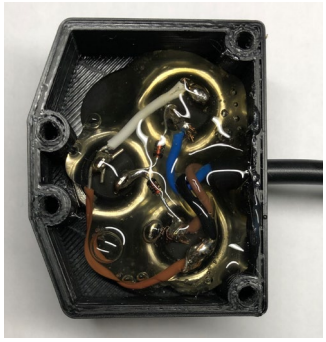


Figura 35: Cableado del mando de control.

A su vez, se adaptó el módulo GPS adquirido, soldándole el cable con el conector M8 correspondiente detallado en la Figura 31.

Una vez se tuvo todo el hardware implementado y se verificó su correcto funcionamiento, se fabricó una carcasa mediante impresión 3D para el odómetro, el mando de control y el módulo infrarrojo.

El diseño se realizó mediante el software CAD Shapr3D para iPadOS, sirviendo de referencia el modelo 3D de ambas PCB que se generó con la herramienta CircuitMaker.

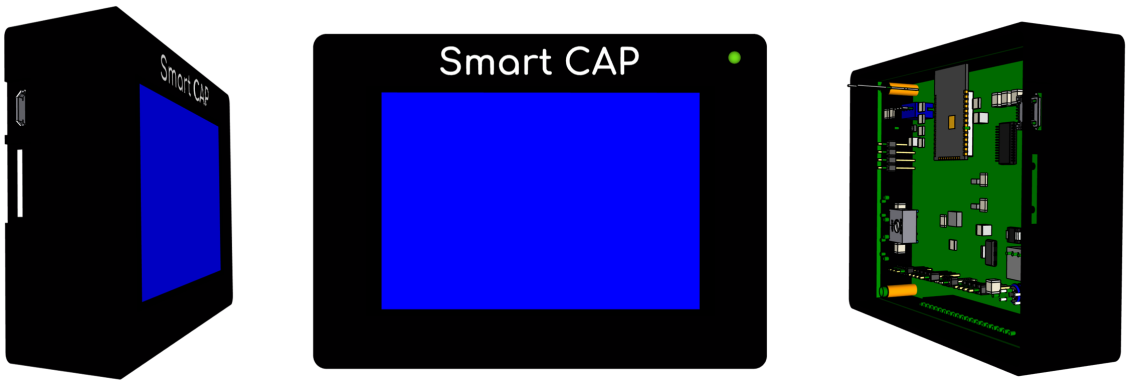


Figura 36: Vistas del modelo 3D de la carcasa para la PCB principal.

La carcasa consta de dos partes: carcasa superior (Figura 36) y tapa posterior (Figura 37). Se construyó de esta forma para poder introducir cómodamente la PCB ya soldada y en caso de avería poder acceder fácilmente a los componentes. Como se puede observar, se diseñó la carcasa entorno al modelo de la PCB, lo que sirvió de gran ayuda para cuadrar los agujeros de sujeción para la pantalla, así como los orificios y ranuras de acceso para el LED, el conector microUSB y la tarjeta microSD.

Figura 37: Modelo 3D tapa posterior.

En la tapa posterior se pueden apreciar tres agujeros en las esquinas que servirán para ensamblar el conjunto mediante unos tornillos roscados M3.
Los dos orificios centrales de mayor tamaño (M5) serán los que se usen para fijar el dispositivo a los soportes de la moto, siendo su separación y diámetro la misma que la de los sistemas comerciales.

En la Figura 38 se puede observar la carcasa propuesta para el del módulo infrarrojo. Se procedió de la misma forma que para la carcasa principal, con la particularidad de que el ensamblaje de la carcasa y la PCB en este caso se realizó mediante el ajuste por presión de la tapa y la aplicación de silicona para asegurar su estanqueidad. Las líneas rojas en los laterales de la carcasa se incluyeron para servir de referencia a la hora de situar el módulo sobre el patrón del roadbook.

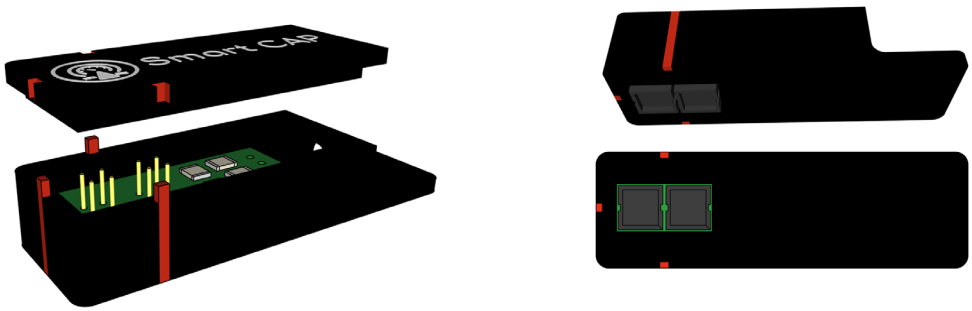


Figura 38: Modelo 3D del módulo IR.

Las carcasas se fabricaron en plástico PLA[11], material muy utilizado en el ámbito de la impresión 3D por su facilidad de tratamiento e impresión. Sin embargo, para futuras versiones, se ha estudiado el uso de plásticos como el ABS[12], mucho más resistente y con mejores propiedades mecánicas.

Se empleó una impresora 3D de un solo extrusor para la fabricación, por lo que hubo que imprimir por separado el logo y los textos en blanco y, sin llegar a retirar dichas piezas de la placa de impresión, imprimir el resto de la carcasa en negro. De esta manera quedaron integradas las letras y el logo en la misma pieza con un acabado profesional y uniforme. Para el laminado de los modelos 3D y la generación de los archivos de impresión se utilizó el programa de laminado CURA de Ultimaker.

En el diagrama de la Figura 39 se pueden observar todos los elementos por los que está compuesto físicamente el sistema, así como su interconexión y el tipo de conector asociado a cada uno de ellos. Se destaca a su vez la polaridad de los cables de alimentación de 12V, que para asegurar la compatibilidad con el resto de sistemas comerciales se tuvo que obviar el código de colores en los cables de conexión (negro positivo, azul negativo) debido a que los conectores se adquirieron ya crimpados.

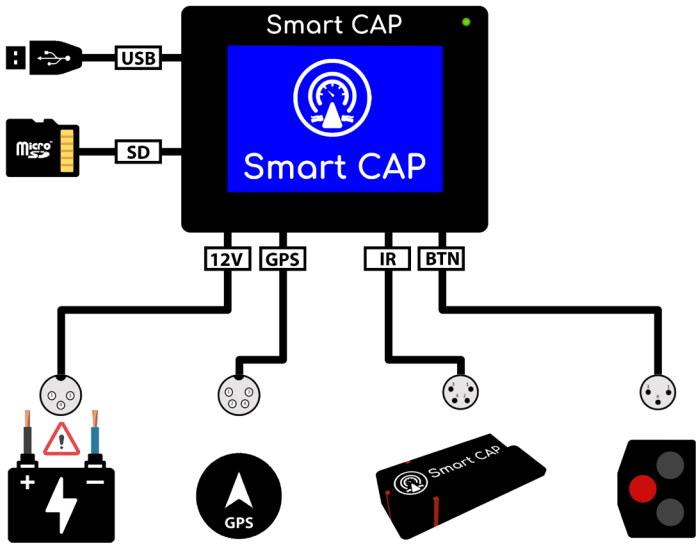


Figura 39: Diagrama de conexión de los elementos del sistema.

[11]El ácido poliláctico (PLA) es un termoplástico cuyos materiales de base se obtienen a partir de almidón de maíz, yuca o caña de azúcar. Es biodegradable en ciertas condiciones.

[12]El acrilonitrilo butadieno estireno (ABS) es un plástico muy resistente al impacto (golpes) muy utilizado en automoción y otros usos tanto industriales como domésticos. Se le llama plástico de ingeniería debido a que su cuya elaboración y procesamiento es más complejo que los plásticos comunes.

4. Diseño del firmware y del software

4.1. Costumer journey y pantallas

Para definir las características que debía tener el dispositivo y asegurar que se implementaban las funcionalidades que los pilotos echan en falta en los dispositivos comerciales, se realizó una entrevista al piloto profesional Joan Pedrero para definir el costumer journey del sistema. De esta manera se consiguieron reunir todas las funcionalidades demandadas y plantear una interfaz del sistema cuyo manejo fuera similar a la de los dispositivos comerciales, siendo esto una gran ventaja para los pilotos al poder familiarizarse con ella fácilmente.

La interacción con las diferentes pantallas y menús se realiza a través del mando que se ha detallado en apartados anteriores. El costumer journey principal que se definió es el siguiente:

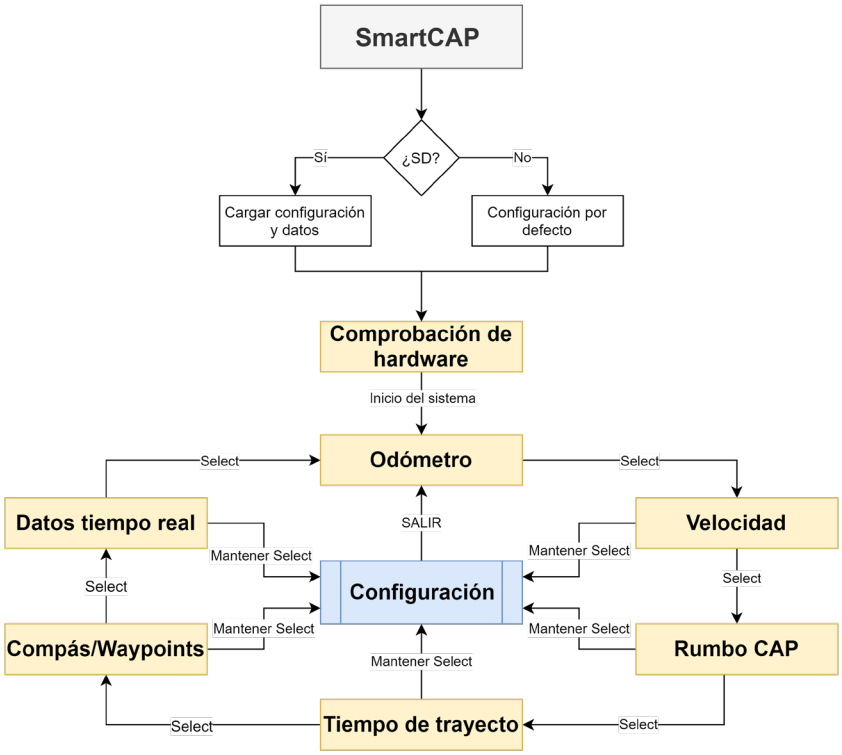


Figura 40: Costumer journey principal.

Nada más encender el sistema se muestra un mensaje de bienvenida con el logo diseñado para el dispositivo, tras ello, lo primero que se hace es comprobar si existen archivos de configuración y de datos de un trayecto anterior en la memoria SD. Dichos archivos podrían ser los guardados automáticamente en el dispositivo tras su último uso, o el usuario podría haberlos introducido en la tarjeta de memoria mediante un equipo externo y algún editor de texto. Esto resulta especialmente útil pues permite al piloto definir el punto de inicio de su trayecto, así como el resto de los datos de navegación; resulta idóneo para realizar etapas aisladas o tramos concretos de una carrera a modo de entrenamiento o incluso para la formación de nuevos pilotos. En

caso de que no se detecten dichos archivos de inicialización, el sistema se inicia con los valores por defecto.

Seguidamente se muestra una pantalla de comprobación de hardware, esta pantalla muestra información sobre el estado del GPS, el IMU, la tarjeta de memoria y la conexión WiFi; indicando si se han podido inicializar correctamente o si de lo contrario ha habido algún fallo de comunicación.

Concluye así el proceso de inicialización del sistema, y a continuación se muestra el menú principal. En el diagrama de la Figura 40 se puede observar como el menú gira en torno a seis pantallas principales en las que en cada una de ellas se muestra un tipo diferente de información. Para navegar entre ellas basta con pulsar el botón select, cambiando así a la siguiente pantalla. En cada una de ellas los botones tienen asignadas diferentes acciones que se detallan en el manual de usuario (Anexo V).

Se debe destacar la presencia de una séptima pantalla, la de configuración. Para acceder a ella se debe mantener pulsado el botón select en cualquiera de las pantallas que no sean la del odómetro. Esto es así pues esa acción ya tiene un uso asignado en dicha pantalla.

Al entrar en la pantalla de configuración se despliega un nuevo menú deslizable que permite al usuario configurar el sistema según sus necesidades. Las opciones que aparecen en el menú se pueden ver en el diagrama de la Figura 41.

Para navegar entre las diferentes opciones del menú se debe pulsa el botón abajo, y el botón arriba para seleccionar una de ellas. El desplazamiento es circular, al igual que en el menú principal, por lo que al llegar a la última opción se vuelve a la primera.

Aunque en la Figura 41 se pueden observar los diferentes ajustes que permite cada opción, en el Anexo V se detalla en profundidad cado uno de ellos.

Tras seleccionar un valor en algunas de las opciones, se regresa automáticamente al menú configuración. Para salir de él basta con navegar hasta la opción “Salir”. Tras seleccionarla, se guarda la configuración y los umbrales definidos (configurar caída y configurar IR) y se regresa a la pantalla desde la que se accedió a la configuración.

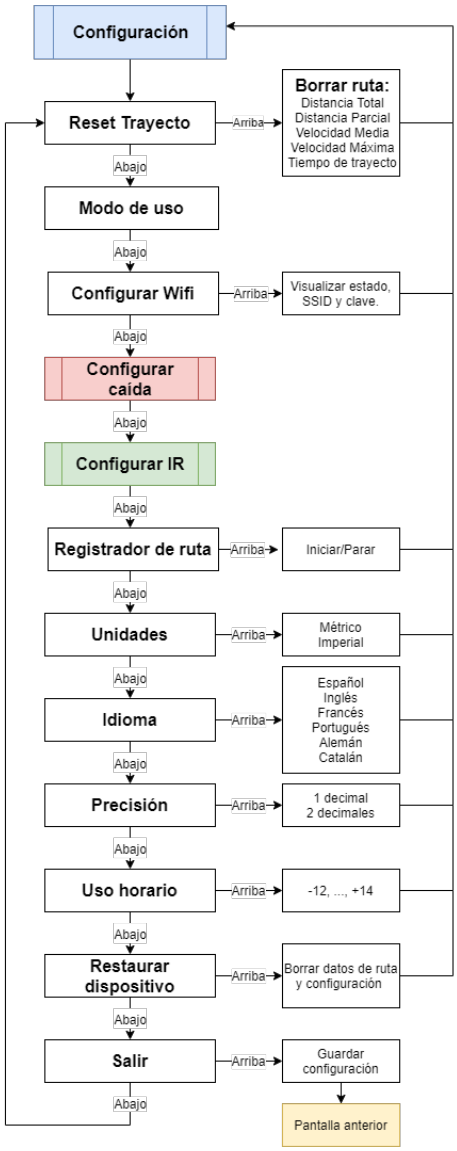


Figura 41: Costumer journey del menú de configuración.

En la Figura 42 se puede observar un ejemplo de cada una de las pantallas mencionadas anteriormente.

Figura 42: Ejemplo de las diferentes pantallas del sistema.

Al tratarse de una pantalla de matriz de puntos en lugar de una de 7 segmentos, podemos hacer un uso mucho más versátil de la misma, incluyendo animaciones, texto y números con diferente formato, formas geométricas, etc. Un ejemplo claro es la pantalla de bienvenida, en la que se muestra el logo del prototipo; para ello se convirtió la imagen a mapa de bits y se importó al proyecto en formato hexadecimal para su representación en pixeles.

Un aspecto importante a destacar es la posibilidad de ver al mismo tiempo la distancia recorrida y el rumbo actual en la pantalla odómetro. Esto es especialmente relevante pues en los dispositivos comerciales no se puede visualizar mas que un dato en la pantalla, por tanto los pilotos deben adquirir un segundo dispositivo para poder visualizar diferentes datos al mismo tiempo.

4.2. FreeRTOS

Para implementar el sistema descrito en el apartado anterior era evidente que el firmware debía ser multitareamultitarea, es decir, que se pudieran realizar diferentes tareas al mismo tiempo, de lo contrario sería imposible mantener los datos actualizados y asegurar las tasas de muestreo de los diferentes sensores y periféricos que componen el sistema; todo esto al mismo tiempo que dichos datos eran mostrados por pantalla.
Para dotar a nuestro sistema de dicha capacidad de multitarea se decidió utilizar un sistema operativo en tiempo real, en concreto FreeRTOS[13].

[13] https://www.freertos.org/

Se trata de un sistema operativo en tiempo real para dispositivos embebidos y está distribuido de forma gratuita bajo la licencia de código abierto MIT. Incluye un kernel y un conjunto de bibliotecas de software que proporcionan métodos para múltiples subprocesos o hilos, mutex, semáforos y temporizadores de software; además de admitir prioridades de hilos. Está diseñado para ser pequeño y simple. El núcleo en sí consta de tan solo tres archivos implementados en C, lo que hace que sea tanto simple como eficiente, con una huella de memoria pequeña, gastos indirectos bajos y ejecución rápida. Actualmente tiene soporte para más de 40 arquitecturas entre las que se incluye la del ESP32. (Wikipedia, 2020)

Figura 43: Logo de FreeRTOS.

Por ello, y dado que el ESP32 integra un microprocesador de doble núcleo Tensilica Xtensa LX6, tenemos la posibilidad de lograr una concurrencia real planificando las tareas en los núcleos adecuados a través de las funciones que nos proporciona FreeRTOS.

El proyecto se desarrolló con la herramienta PlatformIO y se estructuró en diferentes directorios y archivos como se puede observar en la Figura 44. De esta forma resultaba mucho más legible y facilitaba la programación. Dado que el firmware del sistema contenía numerosas líneas de código se decidió hacer el control de versiones mediante GIT y la plataforma GITLab, de esta manera el código siempre estaba actualizado y disponible en cualquier otro equipo a través de la plataforma.

```
src
    SmartCAP.cpp

    functions
        drawFunctions.h
        initFunctions.h
        sdFunctions.h

    globals
        config.h
        enum.h
        globals.h
        language.h
        logo.h

    tasks
        taskConfig.h
        taskGPS.h
        taskHandleData.h
        taskIMU.h
        taskIR.h
        taskLogData.h
        taskSendData.h
```

Figura 44: Estructura en árbol del proyecto.

4.2.1. Tareas principales

El firmware se estructuró en diferentes tareas y cada una de ellas se encarga de controlar alguna de las partes que componen el sistema. De esta forma logramos que confluyan todas a la vez y damos independencia a unas de otras.

Al tratarse de un sistema concurrente se debe controlar de manera exhaustiva el acceso a los recursos compartidos, sin embargo, al estar ante un sistema cuya temporalidad no es cercana a la de los tiempos del reloj, y en los casos en los que se hacen lecturas y escrituras de una misma variable global por diferentes tareas no se trata de variables críticas, dicha gestión de los recursos compartidos se dejó de lado salvo por el uso de algún semáforo cuando se consideró oportuno.

TAREAS SMART CAP				
Identificador	**Prioridad**	**Core**	**Descripción**	**Periodicidad (ms)**
updateDevice	7	0	Controla los estadados del sistema y la visualización por pantalla	Bucle continuo
showConfig	7	1	Gestiona el menú de configuración y la visualización por pantalla	Se inicia y muere
handleIMU	7	1	Lectura de los datos de acelerómetro y sensor magnético (I2C)	SAMPLING_PERIOD_IMU_MS = 200
handleGPS	6	0	Lectura de los datos de GPS (UART)	SAMPLING_PERIOD_GPS_MS = 200
readIR	7	1	Lectura de valores de sensores IR y control de viñetas (ANALOG)	SAMPLING_PERIOD_IR_MS = 200
handleData	4	0	Procesado y calculo de los datos. Creación del struct de datos	DATA_REFRESH_PERIOD_MS = 200
sendData	7	1	Envío de datos al servidor y gestión de la conexión WiFi	SEND_DATA_PERIOD_MS = 1000
logData	5	0	Escritura de datos, configuración y archivo de ruta en la microSD	LOG_TRACK_PERIOD_MS = 1000

Figura 45: Tabla resumen de las diferentes tareas del sistema.

En la Figura 45, se pueden observar las prioridades y núcleos asignados a las diferentes tareas que se definieron. El reparto entre núcleos se realizó teniendo en cuenta que ciertas tareas debían presentar una concurrencia real, es decir, que se ejecutaran simultáneamente en sendos núcleos. De lo contrario, aunque siguiera habiendo concurrencia a través del planificador del sistema operativo en función de las diferentes prioridades asignadas a cada tarea, se podría generar alguna condición de carrera crítica.

Las tareas pueden agruparse en tres boques según el tipo de función que desempeñan. En la Figura 45 se pueden apreciar dichos bloques por colores, siendo el **azul** el correspondiente a tareas de control del sistema; el **naranja** corresponde a aquellas que sirven para comunicarse y gestionar los diferentes periféricos; y el **verde** para aquellas tareas encargadas de procesar, almacenar y enviar los datos.

En todas ellas la periodicidad se controla con la función `vTaskDelay()` de FreeRTOS y se especifica con las constantes que se pueden observar en la última columna de la Figura 45, las cuales se encuentran definidas en el fichero "config.h". Los periodos escogidos para las **tareas de periféricos** se han basado en la frecuencia de refresco que utilizan los módulos GPS de los principales sistemas comerciales. Dicha frecuencia es de 5 Hz (200 ms) y por tanto todos los sensores se han definido con dicho periodo por ser un valor más que suficiente para asegurar una buena velocidad de respuesta y fluidez.

Dado que la tarea **`handleData()`** realiza los cálculos a partir de los datos obtenidos por las anteriores tareas, se estableció su tasa de refresco en 150 ms, de esta forma se actualizan los cálculos con mayor frecuencia de la que se reciben y se evita la pérdida de alguna trama de datos. Por último, para las **tareas de envío y escritura de datos** se estableció un periodo de 1 segundo por ser un valor adecuado para la actualización de datos en la plataforma y para definir los tracks en el archivo de ruta.

A continuación se procede a detallar cada una de las tareas de las que consta el firmware del sistema.

▪ Tarea principal: updateDevice()

Se corresponde con la primera tarea en ejecutarse en nuestro sistema. Esta tarea se trata de un bucle infinito que controla la máquina de estados principal del sistema e inicia y gestiona el resto de tareas. Se ejecuta al terminar la función `setup()` la cual usamos para inicializar el monitor serie para el debug del sistema, configurar los pines de entrada y salida, y algún otro recurso.

La máquina de estados se trata de una FSM[14] la cual consta un total de 9 estados diferentes. Cada uno de estos estados se corresponden con las distintas pantallas del

[14] FSM: Finite State Machine

sistema que ya se han descrito en la Figura 42. Dicha FSM se rige principalmente por el costumer journey de la Figura 40.
Los diferentes estados controlan dos aspectos principalmente: el buffer de la pantalla y los botones. Deben detectar que tipo de pulsación y en que botón se ha producido, para de esta forma ejecutar la acción asignada y calcular el siguiente estado; una vez se ha terminado de ejecutar las instrucciones asignadas a dicho estado, se envía a la pantalla el nuevo buffer actualizado con la información a mostrar por pantalla mediante la función `display.sendbuffer()`, se actualiza el nuevo estado del sistema y se ejecuta un pequeño delay de 10 ms para asegurar que la tarea no ocupa todos los recursos del planificador al tratarse de un bucle infinito. Los estados son los siguientes: "MENU_ODOMETER", "MENU_SPEED", "MENU_CAP", "MENU_TIME", "MENU_WP", "MENU_RAWDATA", "MENU_CONFIG".

Los otros dos estados, "WELCOME" y "CHECK_HARDWARE" solo se ejecutan una vez al iniciarse el sistema. El estado en el que entra la FSM por defecto es el de "WELCOME", en él se llaman a las diferentes funciones de inicialización que inician el resto de tareas, configuran los periféricos, etc. durante esta inicialización se muestra por pantalla el logo del dispositivo (Figura 7). Seguidamente se actualiza el estado del sistema (variable `deviceState_next`) a "CHECK_HARDWARE" en el cual se muestra el estado de los principales periféricos y conexiones e informa si ha ocurrido algún fallo en su inicialización en el estado de bienvenida. Por último, el estado "MENU_CONFIG" tan solo crea la tarea `showConfig()` que se describe en el siguiente punto y bloquea mediante un semáforo a la tarea `updateDevice()` hasta que termina de ejecutarse `showConfig()`.

La información que se muestra en cada pantalla, así como las acciones asociadas a cada uno de los botones se detallan en el manual de usuario (Anexo V).

▪ Tarea de configuración: showConfig()

Se trata de la segunda tarea de control del sistema, encargada de mostrar y gestionar el menú de configuración. El comportamiento de esta tarea es diferente al resto ya que las sentencias no están dentro de un bucle infinito y por tanto, una vez se crea la tarea, se ejecuta y al terminar la ejecución del código correspondiente se auto elimina.

Se trata también de máquina de estados que controla las diferentes pantallas de configuración y su navegación. Para implementar los menús desplegables se han utilizado unas funciones que incluye la librería "u8g2Lib" que se utilizó para el control de la pantalla. Estas funciones son `display.userInterfaceSelectionList()` y `display.userInterfaceMessage()`, a las cuales se les especifica una lista de los diferentes elementos a mostrar mediante los argumentos de la función, e implementa de forma sencilla un menú de selección navegable con las opciones especificadas.

Ambas funciones son funciones bloqueantes, es por ello por lo que se decidió crear una tarea externa a la principal para el control de la configuración; una vez se selecciona el submenú configuración, la FEM calcula el siguiente estado en base a

dicha selección y actualiza la pantalla mostrando el submenú correspondiente. La tarea finaliza cuando se selecciona la opción “Salir”, que libera el semáforo en el cual espera la tarea `updateDevice()` y ejecuta `vTaskDelete()` para auto eliminarse.

▪ Tarea de control del sensor infrarrojo: readIR()

Cada uno de los sensores se controla mediante tareas independientes, se tomó esta decisión para poder controlar las frecuencias de muestreo de cada uno de ellos de manera aislada. La tarea `readIR()` se encarga tanto de la lectura de los CNY70 a través de las entradas analógicas como del cálculo de la viñeta actual en base a dichas entradas. El sistema de detección es el equivalente a un encoder circular solo que a lo largo de todo el roadbook. El patrón que se diseñó para incorporar en las viñetas del roadbook se puede observar en la Figura 47.

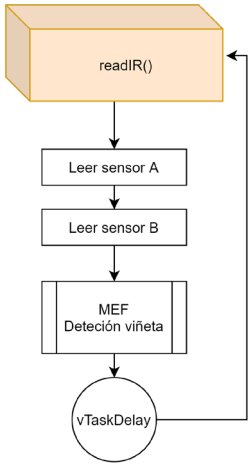


Figura 46: Diagrama de la tarea readIR().

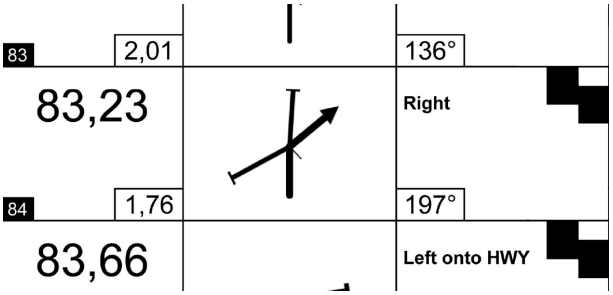


Figura 47: Patrón de detección de viñetas.

Este patrón es el que mejores resultados arrojó tras realizar varias pruebas con patrones de diferentes tamaños, distancias y simetrías. El uso de dos filas de símbolos es necesario para poder detectar tanto el avance como el retroceso de las viñetas.

Para la detección de dicho patrón se implemento una FSM para controlar los posibles estados en función del valor que entregaban los sensores al reflejar el haz infrarrojo en el patrón impreso en la superficie del roadbook. La FSM controla un contador que se incrementa o decrementa en función del estado en el que se encuentra y el resultado que obtiene. En el siguiente diagrama se puede observar el funcionamiento de la misma.

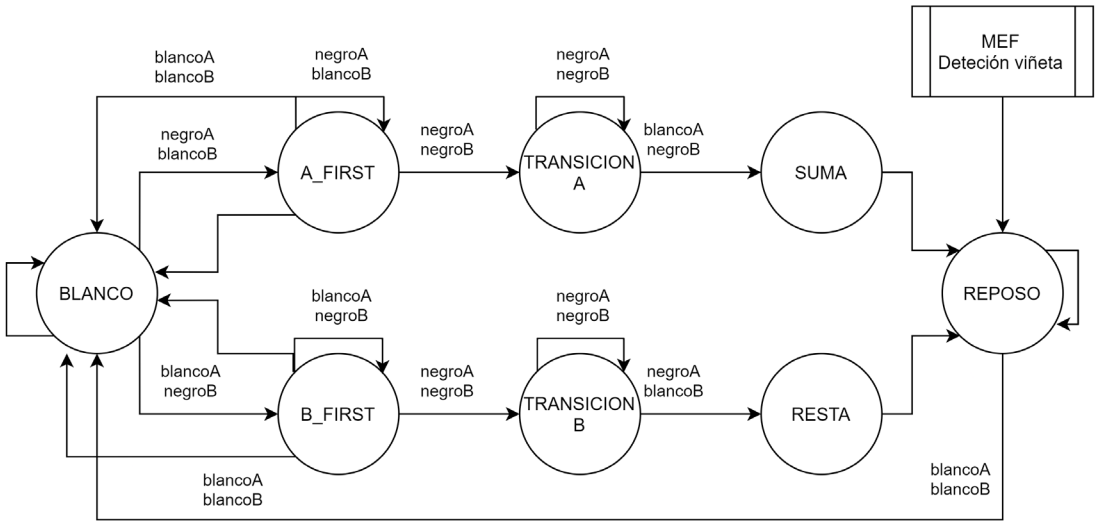


Figura 48: Máquina de estados finita para la detección y control de viñetas.

Se implementó a su vez una función para la configuración de los umbrales de detección, es decir, establecer que valores analógicos de señal extraídos de los sensores se tienen que interpretar como blanco y como negro. Este proceso de configuración es importante pues el escenario de uso del dispositvo es al aire libre, y por tanto la condiciones de iluminación así como la radiación infrarroja del sol podrían interferir en la detección, de esta forma aseguramos que para cada uso se han adaptado los umbrales a dicho contexto.

El proceso de configuración es sencillo y se puede acceder a él desde el menú de configuración. Los valores de referencia se toman mediante una viñeta de prueba que se añadirá al principio de cada roadbook.

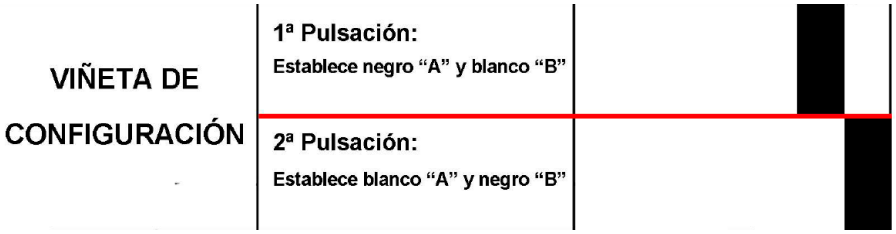


Figura 49: Viñeta de configuración de umbrales IR.

▪ Tarea de control del módulo GPS: handleGPS()

Se trata de la tarea que procesa los datos provenientes del módulo GPS a través del bus UART. Esta tarea se crea al ejecutarse la función `initGPS()` en la tarea principal, la cual configura la comunicación serie con el receptor GPS a 9600 baudios. Se escogió este valor pues aparte de ser el valor que usan los módulos comerciales y de esta manera asegurábamos la compatibilidad, es más que suficiente para la cantidad de datos que se transmiten ya que previamente se configuró el módulo desactivando las tramas NMEA que no eran necesarias para el sistema.

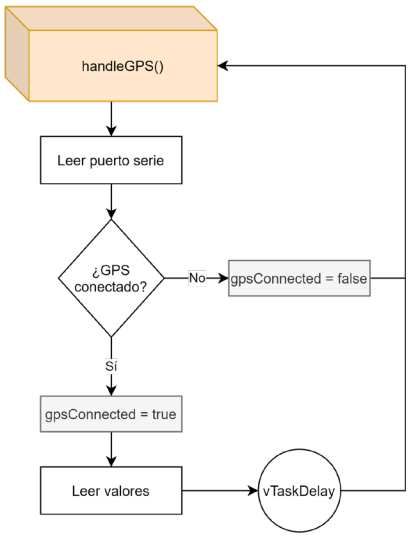


Figura 50: Diagrama de la tarea handleGPS().

El comportamiento de esta tarea es el descrito en la Figura 50. Como se puede apreciar, en cada iteración se verifica si el GPS sigue conectado al sistema, esto se logra comprobando que el número de caracteres procesados es superior al de la iteración anterior.

De estar conectado el módulo, se extraen los valores necesarios de la trama NMEA correspondiente y se guardan en memoria mediante un struct, el cual se sobrescribe en cada iteración. Dado que la hora se recibe en formato UTC, en esta tarea también se ajusta el valor al huso horario correspondiente que se haya especificado desde el menú de configuración.

▪ Tarea de lectura del sensor IMU: handleIMU()

Esta tarea tiene un comportamiento similar a la anterior con la particularidad que la comunicación con el acelerómetro y el magnetómetro se realiza a través del bus I2C. Si bien este proceso se realiza en niveles inferiores por los HAL [15] de las librerías utilizadas, para comprobar que el módulo está conectado y funcionando correctamente fue necesario realizar una lectura del registro WHO_AM_I.
Este tipo de registro es muy común en integrados de este estilo y se trata de una dirección de memoria de solo lectura especificada por el fabricante en la cual se encuentra un byte que identifica al dispositivo y cuyo valor se proporciona en el datasheet.

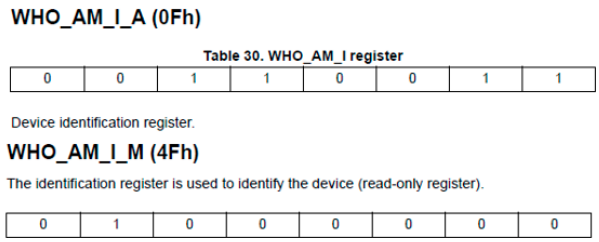


Figura 51: Registros WHO_AM_I del acelerómetro y magnetómetro.

Comparando el valor leído con el conocido por el datasheet, podemos comprobar si el dispositivo funciona correctamente o de lo contrario no ha sido posible comunicarse con él. Como se puede observar en el extracto del datasheet de la Figura 51, el valor que se debería obtener al pasarlo a decimal es “51” para el acelerómetro y “64” para el magnetómetro.

Si la validación es correcta, se leen los valores de los tres ejes tanto del acelerómetro como del magnetómetro y se almacenan en un struct. Esta lectura se lleva a cabo mediante funciones de librería y facilitan la obtención de los datos al realizarse a un nivel más alto de las capas del sistema, como no ocurre con la lectura del registro WHO_AM_I.

▪ Tarea de procesado de datos: handleData()

Esta tarea es la encargada de procesar todos los datos obtenidos por las anteriores tareas y crear la trama final (struct) para su posterior envío.

En la Figura 52 se puede observar el diagrama de flujo de la tarea `handleData()`, así como todos los cálculos que realiza.

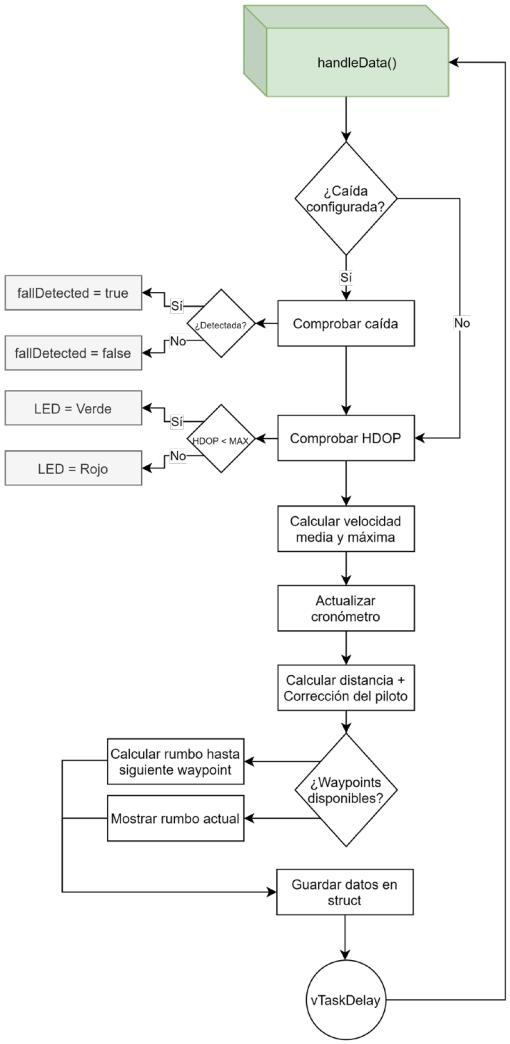


Figura 52: Diagrama de la tarea handleData().

[15] HAL: Hardware Abstraction Layer

La **detección de caídas** se lleva a cabo de una forma sencilla comparando los valores actuales del acelerómetro con los que se han definido como caída (teniendo en cuenta un cierto margen). La configuración de dichos umbrales se lleva a cabo en el menú de configuración a través de un asistente. Para ello se debe tumbar la moto en el suelo de ambos costados y poder guardar los valores de los tres ejes de ambas posiciones que se tomarán como referencia de caída. Si al cabo de un tiempo se obtiene un número determinado de coincidencias, se activa el flag de caída y se muestra una alerta por pantalla, así como en la plataforma de seguimiento.

Seguidamente se comprueba el valor de **HDOP**[16] que reporta el módulo GPS. Esto es necesario pues la precisión del cálculo de distancias depende exclusivamente del sistema GPS. Es por ello por lo que mediante el LED bicolor se le indica al piloto cuando las coordenadas que se obtienen son lo suficientemente precisas como para poder iniciar el trayecto y obtener unos resultados válidos. El valor de HDOP mínimo especificado es de 3.5. Se debe recordar que mediante los botones de recalado, el piloto puede ajustar la distancia recorrida si se produjera algún desajuste significativo con respecto a las viñetas del roadbook. La comprobación del HDOP es solo por motivos informativos ya que el sistema sigue funcionando a pesar de tener un HDOP elevado; por tanto, es el piloto el responsable de resetear e iniciar el recorrido cuando considere oportuno.

A continuación se realizan los cálculos relacionados con la **velocidad**, en concreto de la velocidad máxima y media. Para actualizar la velocidad máxima se llama a una sencilla función que comprueba si el valor actual de velocidad es mayor que el último máximo detectado, de ser así, se actualiza dicho valor. Para el cálculocálculo de la velocidad media se implementó una función de cálculo de media de ventana deslizante, pues es el método más eficiente en términos de memoria para calcular el valor medio de un conjunto de datos. El tamaño de ventana se estableció en 10 muestras y su funcionamiento se basa en el cálculo del valor medio de los elementos de un buffer circular implementado con punteros.

Como se ha podido observar en la Figura 42, el sistema integra un **cronómetro** totalmente funcional basado en el reloj interno del microprocesador. Para su implementación se programó una librería propia que integra las funciones básicas de control de un cronómetro. A pesar de no utilizar un RTC[17] físico, se logró una precisión idéntica a la de los cronómetros que integran los smartphones actuales utilizando la función `millis()`, la cual devuelve el tiempo transcurrido desde el encendido del microprocesador hasta el instante en que se le invoca y, con ciertos cálculos, es posible controlar y medir el tiempo transcurrido. El resultado es un cronómetro el cual se puede parar, reanudar y reiniciar. El hecho de actualizarse en una tarea independiente hace que no se bloquee en ningún momento logrando así la concurrencia necesaria.

Esta tarea es también la encargada de calcular la **distancia recorrida** en el intervalo de tiempo impuesto por la periodicidad de la tarea, en este caso de 150 ms como se

[16] HDOP: Horizontal Dilution Of Precision
[17] RTC: Real Time Clock

detalla en la tabla de la Figura 45. El cálculo se realiza a través de una función de la librería utilizada para la conversión de datos de las tramas NMEA, la cual pasándole dos coordenadas geográficas devuelve la distancia en metros entre ellas. En cada iteración se acumula la distancia entre la ubicación actual y la del instante anterior, y a continuación se efectúan las correcciones de recalado. De esta manera obtenemos los valores de distancia total recorrida (sin recalado) y distancia parcial (con recalado y reiniciable durante el recorrido).

Como se ha detallado en el apartado 2.1, se quiso implementar un sistema que sirviera como alternativa al Iritrack de las carreras profesionales; para ello se añadió una pantalla de **guiado de waypoints**. El funcionamiento de dicha pantalla reside en la lectura de una serie de waypoints almacenados en un fichero con extensión GPX[18] (si existe dicho fichero en la memoria SD) y el posterior cálculo de la distancia y rumbo desde la ubicación actual hasta la del waypoint correspondiente. Una vez se da por válido dicho waypoint (distancia inferior a un umbral especificado), se muestran las indicaciones hasta el siguiente. Las indicaciones constan de un compás que muestra la dirección que el piloto debe seguir, implementado mediante reglas trigonométricas para el trazado de la aguja; la visualización numérica del rumbo a seguir y la distancia en metros a la que se encuentra del siguiente waypoint.

Por último, una vez calculados todos los datos anteriormente descritos, se guardan todos ellos en un struct el cual conforma la **trama de datos** que es usada por las funciones de visualización del display y que posteriormente se enviará a la plataforma web.

▪ Tarea de envío de datos y gestión del WiFi: sendData()

La tarea `sendData()` integra tanto la gestión de la conexión WiFi como el envío de telemetría al servidor Thingsboard. En un principio las dos funcionalidades se implementaron en tareas separadas, pero dado que ambas estaban estrechamente relacionadas, y el envío de datos requería una comprobación de conexión con el servidor, así como validar el estado del WiFi, se decidió unificarlas en una misma tarea.

El diagrama de flujo es el que se muestra en la Figura 53. En primer lugar, al crearse la tarea se inicializa la interfaz WiFi y se configura para su conexión al punto de acceso especificado en el archivo de configuración WiFi de la tarjeta SD.

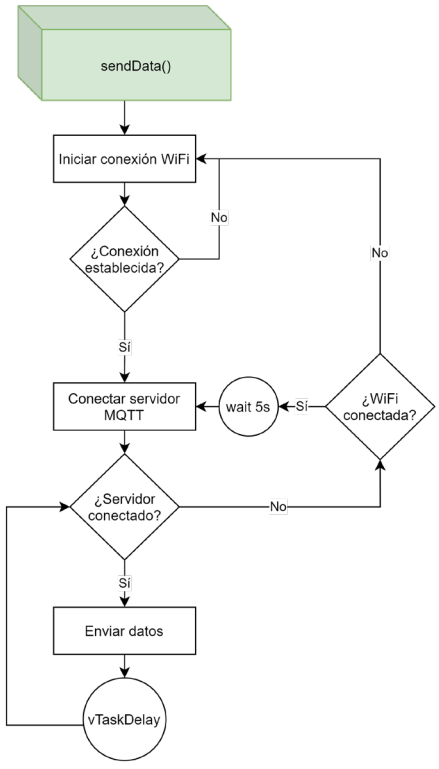


Figura 53: Diagrama de la tarea sendData().

[18] GPX: GPS eXchange Format (Formato de Intercambio GPS) es un esquema XML pensado para transferir datos GPS entre aplicaciones.

La tarea envía datos al servidor cada segundo, y en cada iteración se comprueba la conexión con el servidor, en caso de que se haya perdido, se llama a la función de reconexión que comprueba si sigue activo el punto de acceso WiFi y lo reconecta en caso de desconexión; si por el contrario sigue activa la conexión WiFi pero no la del servidor, se intenta reconectar con la página de Thingsboard pasados 5 segundos de espera. Una vez se haya restablecido la conexión se envía la trama de datos generada en la tarea `handleData()`.

▪ Tarea de escritura de datos en memoria microSD: logData()

La tarea `logData()` se encarga únicamente de la escritura de datos en la memoria SD. Hay dos tipos de datos que se almacenan en la tarjeta: los datos de trayecto y, en caso de que esté habilitada la opción, las coordenadas de la ruta que está siguiendo el piloto.

Los datos de trayecto se guardan en un fichero que se lee al iniciarse el sistema, de esta forma conseguimos que ante algún apagado repentino del vehículo o una parada programada por el piloto, no se pierdan los datos del recorrido actual y solo puedan ser borrados si el piloto reinicia el track manualmente desde el menú de configuración. Como no es necesario almacenar cada poco tiempo dichos valores, se implementó un contador el cual contabiliza iteraciones del bucle principal y cuando llega al valor definido guarda los datos. De esta manera la misma tarea controla ambas escrituras pero a frecuencias diferentes pues creamos un subperiodo dentro de la propia tarea, el cual se estableció en 5 segundos.

Por otro lado, el recorrido se almacena en un archivo GPX para poder ser visualizado en cualquier aplicación de terceros una vez el piloto termina la ruta. Las coordenadas se guardan cada segundo (periodo de repetición de la tarea) siempre y cuando esté habilitada la opción de guardar el recorrido. En la Figura 54 se puede observar el comportamiento descrito anteriormente.

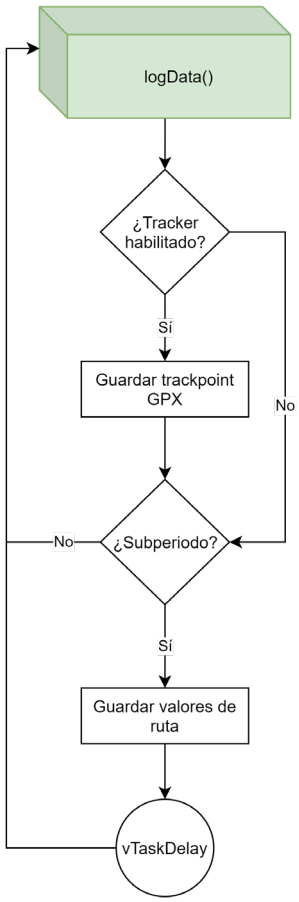


Figura 54: Diagrama de la tarea logData().

4.2.2. Almacenamiento y gestión de los datos

Como se ha indicado en el apartado anterior, varias tareas leen y escriben ficheros en la memoria microSD. Por tanto, fue necesario crear una jerarquía de directorios y archivos para de esta forma facilitar la navegación y visualización cuando se inserte la memoria en un equipo externo.

Cuando arranca el sistema, se monta el sistema de archivos y se inicializa la SD con la función `initSD()`. Si se trata de la primera puesta en marcha del dispositivo o se

utiliza una tarjeta SD nueva, se crean todos los directorios y archivos necesarios, sin embargo, en caso de que ya existan no se sobrescriben.

Los diferentes directorios y ficheros, así como su jerarquía pueden observarse en la Figura 55. Como se puede apreciar consta de 3 directorios: config, data y waypoints.

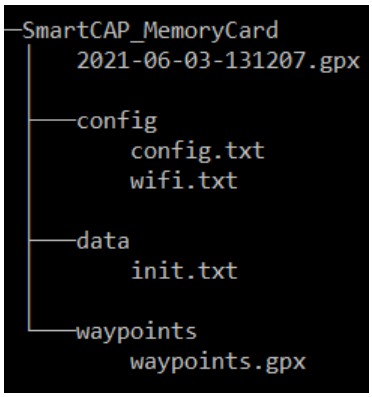


Figura 55: Archivos de la memoria SD.

En el directorio "config" encontramos el archivo de configuración del dispositivo **"config.txt"**, el cual contiene los diferentes parámetros, umbrales, etc. que pueden establecerse desde el menú de configuración del dispositivo, y que se guardan una vez se sale de dicho menú. También encontramos el archivo **"wifi.txt"** en el cual se puede definir el SSID y la contraseña del punto de acceso al que deberá conectarse el dispositivo.

En la carpeta "data" se guarda el archivo **"init.txt"**, que contiene todos los datos del último trayecto llevado a cabo. Este es el fichero sobre el cual escribe datos la tarea `logData()` cada 5 segundos. Cuando se inicia el dispositivo, el sistema lee el archivo e inicializa las variables de distancia, velocidad máxima, viñeta actual, etc. a los valores almacenados en el fichero.

El tercer directorio es la carpeta "waypoints", en ella se guarda el archivo **"waypoints.gpx"** el cual contiene los puntos de control hacia los que el dispositivo guiará al piloto. Este fichero no es generado por el sistema y debe introducirse manualmente en la tarjeta microSD.

Como puede observarse en la Figura 55, los archivos que registran la ruta que ha seguido el piloto se guardan en la raíz de la tarjeta de memoria. El nombre del archivo contiene la fecha y hora en la que se inició el track y el formato es .GPX, para de esta forma poder visualizar el recorrido en cualquier aplicación de terceros.

Salvo los archivos .GPX, el resto de ficheros que son generados por el sistema utilizan el formato JSON[19]. Se decidió usar este formato en lugar de XML ya que es másmás sencillo de leer y trabajar con él. Además, estos archivos están pensados para que puedan ser leídos y modificados por el usuario y permitir así configurar el sistema y establecer valores de inicio de forma manual a través de un editor de texto.

Este fue uno de los motivos por los que se decidió utilizar la tarjeta SD como medio para guardar datos en lugar de la memoria flash que integra el propio ESP32 mediante el SPIFFS[20]; además, el número de escrituras en los ficheros es muy grande a lo largo de un recorrido completo y este tipo de memorias tienen un número máximo de escrituras.

[19] JSON: JavaScript Object Notation. Es un formato de texto sencillo para el intercambio de datos.
[20] SPIFFS : SPI Flash File System

4.3. Comunicación y protocolo MQTT

En el anterior apartado se ha descrito el funcionamiento de las tareas que conforman el sistema, así como la estructura de datos que usa la tarea `sendData()` para enviar la telemetría al servidor web. El protocolo que se utiliza para llevar a cabo dicha comunicación es el conocido protocolo MQTT.

MQTT son las siglas Message Queing Telemetry Transport y es un protocolo de comunicación de cola de mensajes machine-to-machine.

Se trata de un servicio de mensajería push que sigue un esquema publicador/suscriptor (pub-sub). En este tipo de infraestructuras los clientes se conectan con un servidor central denominado broker. Los mensajes que son enviados por los clientes se filtran de forma que se disponen en topics organizados jerárquicamente. Un cliente puede publicar un mensaje en un determinado topic, mientras que otros clientes pueden suscribirse a este topic, y el broker le hará llegar los mensajes suscritos. La siguiente figura muestra el esquema básico del protocolo.

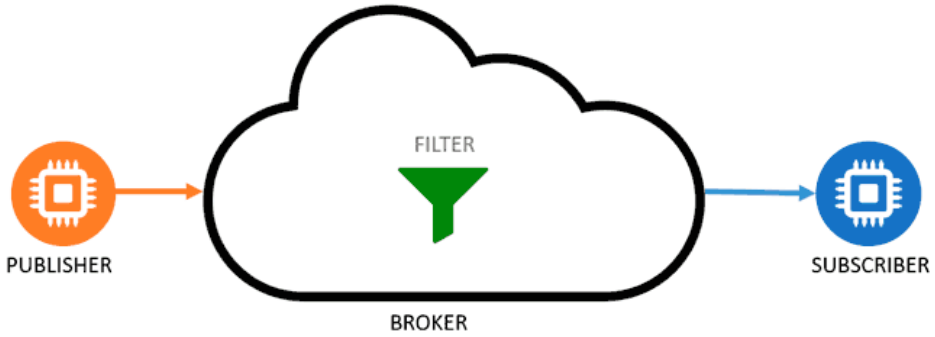


Figura 56: Esquema básico del protocolo MQTT.

Para iniciar una conexión el cliente envía un mensaje CONNECT que contiene información necesaria (nombre de usuario, contraseña, client-id…) y el broker responde con un ACK que indica si la conexión ha sido aceptada, rechazada, etc.

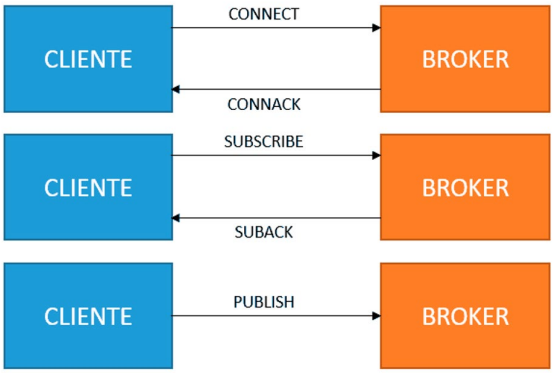


Figura 57: Comunicación básica por MQTT.

Para suscribirse y desuscribirse de un topic se emplean mensajes SUBSCRIBE y UNSUSCRIBE, a los cuales el servidor responde con ACK para cada tipo de mensaje. EL cliente envía los datos mediante mensajes PUBLISH, que contienen el topic y el payload. Finalmente, el cliente se desconecta enviando un mensaje de DISCONNECT (Llamas, 2019).

La Figura 57 muestra de forma simplificada los mensajes que son intercambiados por un cliente y el broker para iniciar la comunicación.

En nuestro sistema todo este proceso se implementa bajo la API que proporciona Thingsboard, de esta forma se simplifican las tareas de conexión y publicación de datos. Para la conexión del dispositivo con el servidor tan solo es necesario especificar tres parámetros a la función `connect()` de la API. El primero de ellos es el servidor, en este caso "tfg.howlab.es"; el segundo es el token de acceso, que es un código que se establece al agregar un dispositivo en la plataforma Thingsboard y sirve como mecanismo de autenticación; por último, el tercer parámetro hace referencia al puerto utilizado para la conexión, en este caso el 41883, que no corresponde a los puertos reservados del protocolo MQTT, pero es con el que está configurado el servidor.

Una vez se ha completado el proceso de autenticación y suscripción al broker, el envío de datos resulta trivial a través de una serie de funciones que varían en función del tipo de dato que se quiere enviar (float, bool, int…) y en las que tan solo hay que especificar el topic al que se va a mandar el dato y su respectivo valor.

4.4. Software. Plataforma web

El parte correspondiente al software del sistema reside en una plataforma web accesible desde cualquier dispositivo, de esta forma se ofrece una gran versatilidad a los usuarios para visualizar la telemetría que reporta el aparato.

Como se ha indicado anteriormente, de todas las plataformas gratuitas que existen actualmente, se escogió Thingsboard por las posibilidades y prestaciones que ofrece. Además, permite una gestión de usuarios para acceder a los datos pudiendo ofrecer así una plataforma de visualización con un carácter más profesional de cara al futuro.

Lo primero que se hizo fue diseñar un boceto de la interfaz de usuario con la herramienta Photoshop. El objetivo que se buscó fue replicar el aspecto del cuadro de instrumentos que llevan los pilotos en sus vehículos, así como distribuir coherentemente el resto de datos que se consideraron relevantes para el seguimiento del piloto. Se puede observar el boceto que se planteó en las siguientes figuras.

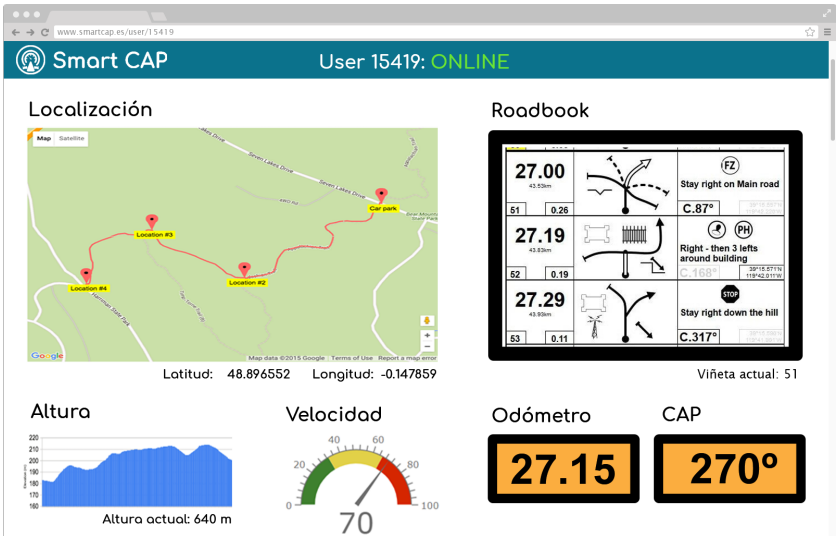


Figura 58: Boceto de la interfaz de usuario (página 1).

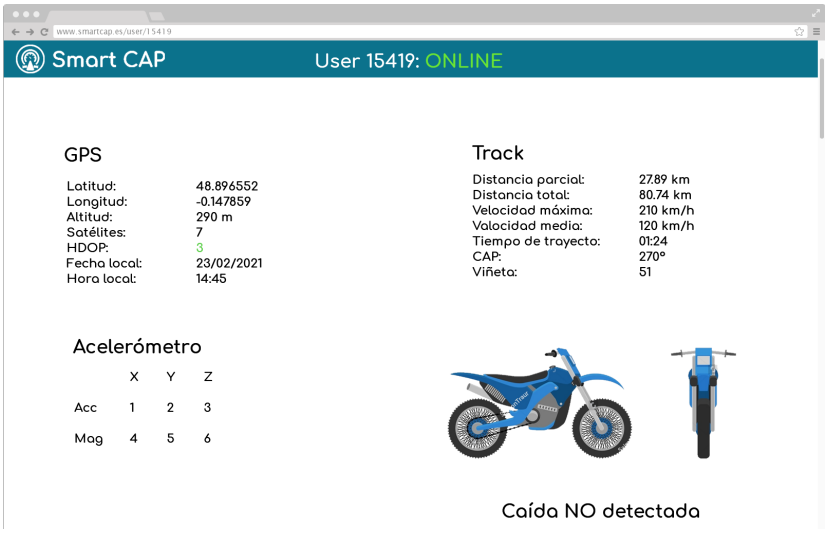


Figura 59: Boceto de la interfaz de usuario (página 2).

Se plateó una interfaz de dos páginas, la primera correspondiente a la pantalla principal que contiene los widgets con los datos más detacables y relevantes; y la segunda, una pantalla de texto en la cual se visualizarían todos los datos en bruto para tener una visión global de todo lo que está reportando el dispositivo.

Para implementar paneles de visualización, Thingsboard permite diseñar dashboards en los que representar la información de los dispositivos mediante la inclusión de diversos widgets que incorpora la plataforma, pudiendo configurar el aspecto y la funcionalidad de los mismos desde la propia plataforma. En la Figura 60 se pude apreciar los diferentes tipos de widgets que permite seleccionar.

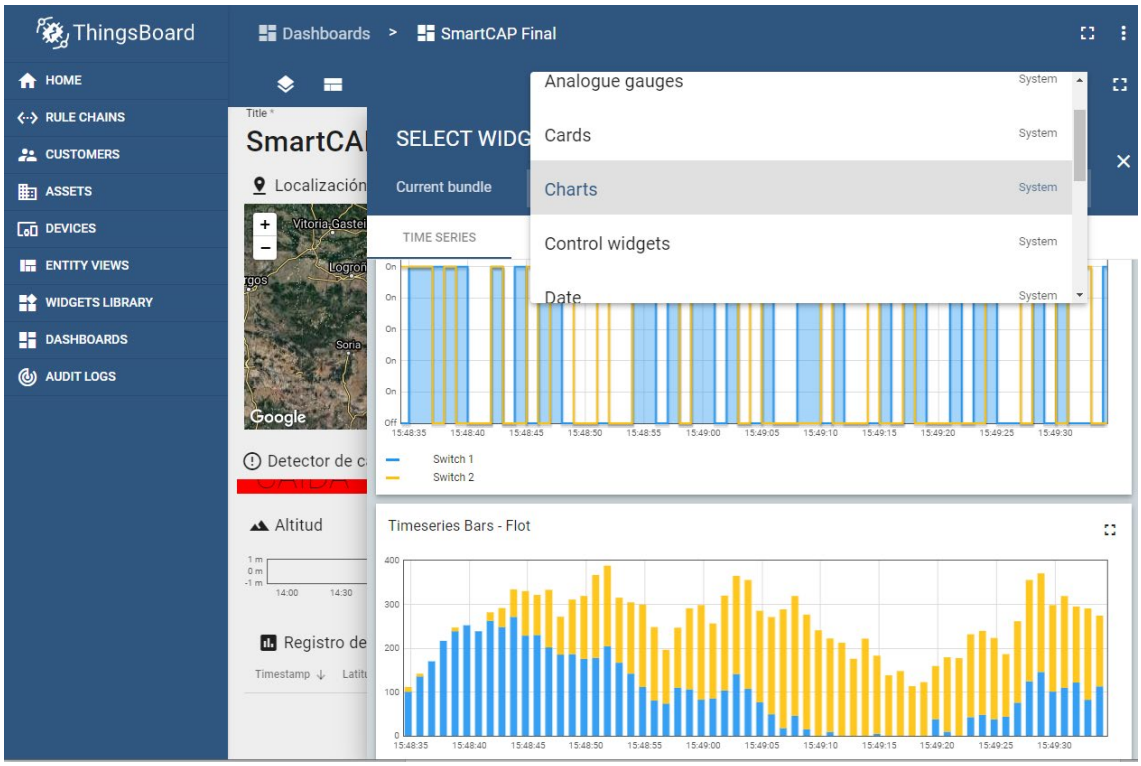


Figura 60: Menú de importación de widgets de Thingsboard.

La mayoría de dichos widgets se encuentran ya preconfigurados y tan solo requieren que se les indique el dato (topic) que deben procesar y establecer algunos parametros de visualización como color, tipografía, etc. Sin embargo, existe la posibilidad de editar el contenido y comportamiento de algunos de ellos a través de los diferentes menús de edición de cada widget como se puede observar en la Figura 61.

Figura 61: Menú de configuración del widget HTML Value Card.

Teniendo claro este planteamiento y una vez estudiadas todas las opciones de personalización y representación de datos que ofrecía la plataforma, se diseñó un dashboard que fue retocandose hasta obtener un resultado funcional similar al boceto que se hizo.

Para probar el correcto funcionamiento de la plataforma se programó un sencillo código de prueba, el cual generaba una serie de datos aleatorios de manera periódica para cada uno de los parámetros, y los mandaba a la plataforma a través de la API de Thingsboard. Dicho proyecto se cargó en el módulo de desarrollo mostrado en la Figura 13.

Como se puede apreciar en la Figura 62, el resultado final del panel de visualización guarda cierta similitud con el boceto planteado inicialmente, salvo por algunas particularidades que se proceden a comentar.

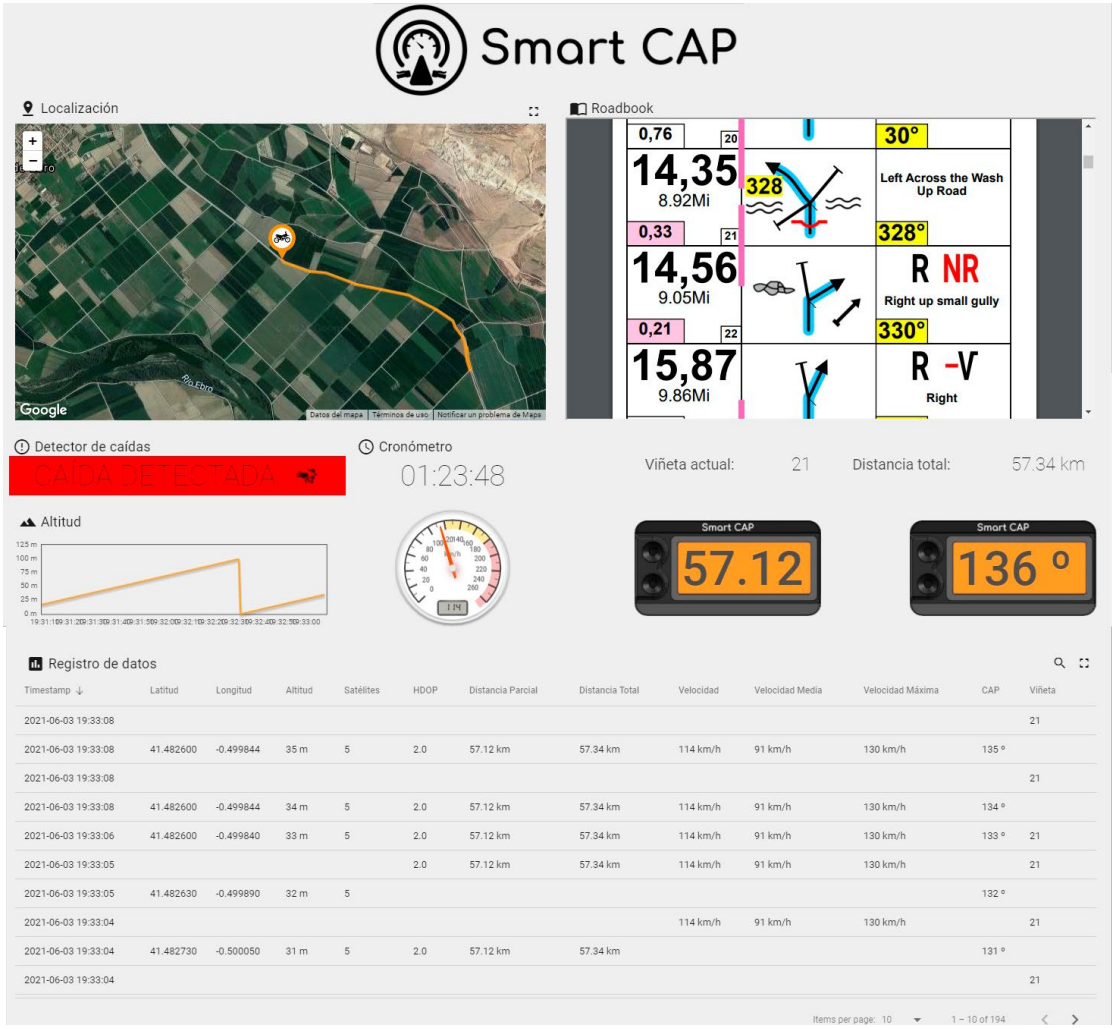


Figura 62: Captura de la plataforma online.

En primer lugar, se desechó la idea de una segunda pantalla para mostrar los datos en bruto pues el widget **“Timeseries table”** permitía mostrar todos los datos que se especificaran con su timestamp correspondiente. De esta forma obtenemos una tabla ordenada y actualizada con todos los parámetros, con la posibilidad añadida de poder buscar los valores para un instante de tiempo concreto. Este widget se encuentra en la Figura 62 bajo el nombre de “Registro de datos”.

Para la visualización de datos como pueden ser el valor del cronómetro, la viñeta actual y la distancia total, el widget **“HTML Value Card”** fue el escogido. Este widget muestra el contenido de la variable que se especifique mediante código HTML y el formato de estilo CSS especificado, lo cual resultaba idóneo para mostrar simples valores.

Un widget que también se usó fue el **"HTML Card"**, estrechamente relacionado con el anterior con la particularidad de que este widget muestra el contenido completo que se especifique en HTML como si de una página web se tratara. Esto proporciona mucha versatilidad a la hora de crear widgets concretos, como es el caso del visor del roadbook, para lo cual se implementó un visor embebido de un PDF específico alojado en Google Drive. La función de este widget es mostrar el roadbook que lleva impreso el piloto y de esta forma, al conocer la viñeta actual en la que se encuentra gracias al módulo IR, el usuario que este visualizando remotamente los datos puede anticipar sus movimientos, validar su recorrido y detectar fallos de navegación para darle feedback al piloto.

Para mostrar la altitud del terreno y la velocidad se escogieron unos widgets ya optimizados para mostrar dicho tipo de datos, **"Timeseries Flot"** y **"Speed Gauge"** respectivamente, en los cuales solo se personalizó la apariencia y ciertos parámetros secundarios.

Por último, de todos los widgets de mapas disponibles, se escogió el **"Route Map"** para representar el recorrido del piloto por ser este el que más se adecuaba al tipo de ruta a mostrar. Se escogió como fuente del mapa la de Google Maps con carácter provisional, ya que es la que más detalles ofrece y para uso no comercial es gratuita. También se diseñó una chincheta de ubicación con una moto en su interior en varios colores y se configuró para que se mostrarán datos del recorrido al posicionar el ratón sobre ella.

Figura 63: Detalle de la chincheta de ubicación en el mapa.

5. Resultados y prototipo final

Una vez finalizado el proceso de desarrollo se comenzó la fase de pruebas. Se ensamblaron e interconectaron todos los módulos y periféricos, así como el dispositivo principal y se procedió a realizar tests concretos para verificar el correcto funcionamiento de todas las funcionalidades y opciones que ofrece el sistema.

En primer lugar se comprobó la correcta visualización por pantalla de todos los menús, textos y símbolos, así como la verificación de una correcta traducción de los textos en cada uno de los 6 idiomas en los que se puede configurar el sistema. Se testeó el costumer journey definido, realizando diferentes recorridos entre menús y opciones para comprobar si alguna combinación podía generar algún tipo fallo y descartar así cualquier condición de carrera que pudiera comprometer la integridad del sistema. Una vez se verificó que el sistema era estable y su manejabilidad era la adecuada, se comenzaron a probar cada una de las funcionalidades por separado antes de realizar una prueba final de campo.

Figura 64: Menú de selección de idioma.

Lo primero fue comprobar la correcta recepción de señal GPS, así como una visualización fluida de los datos de distancia, velocidad, ubicación, etc. Dichas pruebas se realizaron en coche en zonas no urbanas pues la presencia de edificios dificultaba en algún caso la recepción de señal con un buen HDOP. Se detectaron algunos fallos en el método de cálculo de distancias y se corrigieron. Al mismo tiempo se verificó que el sistema de transmisión de telemetría funcionaba correctamente y se recibían los datos en la plataforma web.

Dado que se llevaron a cabo varios recorridos, se comprobó también el correcto funcionamiento del registrador de ruta. Se detectó un fallo relacionado con la prioridad asignada a la tarea `logData()`, encargada de la escritura de datos en la tarjeta SD. El problema era que se le había asignado una prioridad muy baja y el planificador de FreeRTOS no planificaba las instrucciones de dicha tarea. Se elevó dicho valor y el problema se solucionó. Se obtuvieron así diferentes archivos de rutas en formato GPX y se comprobó la precisión de los recorridos con un visualizador de rutas GPX.

A su vez se probó la función de guiado hacia waypoints. Para ello se generó un archivo con diferentes localizaciones mediante la herramienta GoogleEarth, y se introdujo en la tarjeta de memoria del dispositivo. El resultado fue el esperado, el compás indicaba correctamente la dirección en la que se encontraba el siguiente waypoint, y los cálculos de distancia restante eran precisos, así como el avance automático al acercarse a los diferentes waypoints.

La última función que se comprobó fue la de reconocimiento de viñetas. Debido a la falta de recursos no fue posible imprimir un roadbook completo con el patrón de detección propuesto. Para realizar dichas pruebas se diseñó un roadbook de tamaño A4, al cual se le añadió el patrón de detección en la esquina superior de cada viñeta mediante la herramienta Photoshop. Como se puede apreciar en la Figura 65, el sistema era capaz de reconocer tanto el avance como el retroceso de las viñetas. El efecto de avance se simuló mediante el movimiento manual del módulo IR sobre el folio impreso.

Figura 65: Prueba de detección de viñetas.

Una vez se comprobó el correcto funcionamiento de todas las funcionalidades y se realizaron los arreglos pertinentes de los pequeños fallos detectados, se contactó con el piloto profesional Joan Pedrero para poder realizar una prueba final de campo instalando el sistema SmartCAP en la moto oficial que usó en el Dakar 2021.

En la siguiente imagen se puede ver como quedó instalado el dispositivo junto con el portaroadbook y otros dos odómetros comerciales. Como se puede apreciar, el sistema fue alimentado mediante la conexión USB y una batería portátil en lugar de utilizar la toma de 12 voltios de la moto. Esto fue debido a que la instalación eléctrica de la moto dakariana lleva un sistema de alimentación diferente para los odómetros propios que se les exige equipar en dicha competición, por ello, para no tener que realizar ningún cambio en la moto, se decidió utilizar la alimentación por USB y demostrar así la versatilidad que presenta dicha opción.

Figura 66: Instalación en la moto dakariana.

Como se puede observar, la instalación del sistema es compatible con los soportes estándar del mercado incluso con los usados por profesionales. Además, el uso de una pantalla LCD retroiluminada permite una visualización cómoda de los datos y no presentó problemas en cuanto a su visibilidad como se temía en un principio.

Se llevaron a cabo diferentes recorridos y se sometió el dispositivo a las condiciones reales de funcionamiento: alta velocidad, terreno irregular, saltos, polvo, etc. y el dispositivo, a pesar de tratarse de un prototipo con una carcasa no definitiva, resistió perfectamente el estrés mecánico de la prueba.

Tras terminar el recorrido de prueba se le preguntó al piloto acerca de la manejabilidad del SmartCAP, a lo que respondió que era muy intuitivo de usar y que los controles se correspondían con los estándar en este tipo de aparatos. También se le preguntó acerca de la precisión que ofrecía el sistema, y si era comparable a la de los otros dispositivos que llevaba a bordo durante la prueba; el piloto aseguró que presentaba mayor precisión que el odómetro comercial de sensor Hall, y que, con respecto al odómetro basado en GPS, la precisión en las mediciones era idéntica.

Figura 67: El piloto Joan Pedrero manejando el sistema SmartCAP.

6. Conclusiones

6.1. Objetivos logrados

El objetivo de este proyecto era realizar un sistema integral de navegación rally para pilotos, que contara con tecnología IoT para la monitorización remota y buscando además la máxima compatibilidad con los sistemas ya existentes. Tras las pruebas realizadas descritas en el apartado anterior, y en base al feedback recibido por parte del piloto Joan Pedrero, así como de gente del entorno aficionada a esta disciplina, se puede concluir que se ha cumplido el objetivo establecido.

No solo se ha conseguido crear un prototipo comercial desde cero, sino que se ha logrado realizar un prototipo de producto equiparable a los que se comercializan actualmente a precios muy elevados y que ofrecen muchas menos funcionalidades que el SmartCAP.

A su vez, el hecho de incorporar un sistema de monitorización remota abre las puertas a su uso como medida de control y seguridad en multitud de eventos, así como a nivel particular para que los familiares y amigos puedan seguir de cerca el recorrido de los pilotos y estar tranquilos por si ocurriera algún incidente durante el trayecto.

En cuanto a los contenidos y conceptos abordados en este proyecto, se han combinado diferentes disciplinas de ingeniería de telecomunicaciones, así como de diseño del producto.

En el diseño de las placas de circuito impreso se han aplicado los conocimientos de las asignaturas de **electrónica** cursadas, tanto analógica para los bloques de alimentación, módulo infrarrojo, etc. como digital para la comunicación entre periféricos a través de los diferentes protocolos y la programación del microprocesador. A su vez, resulta evidente la presencia de conceptos propios de asignaturas de programación y **arquitectura de sistemas** por el mero hecho de usar un sistema operativo en tiempo real como base del firmware. También han sido útiles los conceptos de las diferentes asignaturas de **telemática** a la hora de comunicar el sistema con la plataforma web mediante el protocolo MQTT y plantear la arquitectura global del sistema.

Este proyecto además ha dado la posibilidad de familiarizarse y coger experiencia con múltiples herramientas del ámbito del IoT, así como de desarrollo electrónico. Se destaca también el manejo de software CAD para el diseño de las carcasas de los diferentes elementos que componen el sistema.

Los lenguajes de programación con los que se ha trabajado y sobre los que se ha profundizado más han sido C++ para la programación del firmware, XML para la creación y lectura de ficheros del sistema, y tanto JavaScript como HTML para el desarrollo del dashboard de la plataforma web.

6.2. Plan de futuro y mejoras

En base a los gratificantes resultados obtenidos en las pruebas del prototipo, y el interés que ha despertado en pilotos y aficionados con los que se contactó para documentarse sobre las necesidades que se debían cubrir, realizar pruebas de funcionamiento, y recibir feedback sobre el acabado final del prototipo; se ha decidido seguir adelante con el proyecto e ir mejorándolo y puliendo hasta conseguir un producto comercial con prestaciones profesionales.

Al tratarse de la primera versión de un producto, se detectaron algunos errores y carencias durante su desarrollo que, aunque se han conseguido solventar para lograr un prototipo totalmente funcional, se van a tener en cuenta para futuras versiones del dispositivo. A su vez existen numerosas ideas que no se han podido implementar por falta de tiempo o recursos, sin embargo, se lleva idea de incluirlas y llevarlas a cabo conforme se siga desarrollando.

Dichas modificaciones y mejoras se han agrupado según sus características en alguno de los tres bloques generales que componen el sistema: hardware, firmware y software. A continuación se describen cada una de ellas.

HARDWARE

▪ Componentes SMD de mayor tamaño, de tal forma que las labores de soldado y ensamblaje se simplifiquen.

▪ Cambiar huella de conectores microUSB y microSD por otras que soporten mejor el esfuerzo mecánico.

▪ Eliminar botones boot y reset, así como su circuito asociado dado que se incluyeron como medida de contingencia en caso de no funcionar la función de autoprogramación.

▪ Eliminar circuito y huella del acelerómetro debido a su gran dificultad de soldadura que presenta el chipset. En su lugar se utilizará el conector reservado para la conexión de un módulo externo con interfaz I2C, lo que ofrece mayor versatilidad y ha dado buenos resultados en el prototipo.

▪ Redistribuir la disposición de los conectores externos para que no interfieran en los anclajes del vehículo.

▪ Reubicación de potenciómetro para el control del contraste de la pantalla o control digital del mismo mediante PWM.

▪ Carcasa y uniones estancas, ya que las actuales son una prueba de concepto y las condiciones meteorológicas a las que está expuesto el aparato requieren robustez e impermeabilidad.

FIRMWARE

▪ Mejorar el algoritmo de detección de caída, debido a la sencillez y poca robustez que presenta el actual. No se profundizó en él por alejarse de los objetivos de este trabajo.

▪ Configurar la conexión WiFi vía servidor web para poder introducir las credenciales del punto de acceso de manera sencilla sin tener que modificar el archivo de configuración directamente de la microSD.

▪ Implementar conectividad CAN tanto por cable, como a través de bluetooth (con adaptadores OBD de terceros) y así evitar el uso del cable de conexión.

▪ Uso de librerías con licencia no vírica o de creación propia. Ver Anexo II.

▪ Cambio del algoritmo de guiado hasta waypoint adaptándolo a las normas y especificaciones de las competiciones oficiales.

▪ Actualización de firmware mediante tarjeta SD.

SOFTWARE

▪ Acceso a la plataforma mediante inicio de sesión de usuarios para garantizar la privacidad.

▪ Posibilidad de crear carreras y competiciones en la propia plataforma mediante el registro y agrupación de varios dispositivos.

▪ Posibilidad de importar y superponer el recorrido original en la plataforma para verificar si el piloto se desvía de la ruta original.

▪ Implementar un sistema de selección de archivos que permita importar el roadbook al widget de visualización de forma sencilla.

7. Bibliografía

f2r. (19 de Junio de 2021). *Kit de navegación RB850 de la marca f2r*. Obtenido de f2r.pt: https://www.f2r.pt/es/Bundle002RB850

f2r. (19 de Junio de 2021). *Mando de control para ICO Rallye MAX e ICO Rallye MAX G.* Obtenido de f2r.pt: https://www.f2r.pt/ico008

f2r. (19 de Junio de 2021). *Mando de control para roadbook y tripmaster.* Obtenido de www.f2r.pt: https://www.f2r.pt/CR002

f2r. (19 de Junio de 2021). *Odómetro ICO Rallye MAX 2 de la marca ICO Racing*. Obtenido de f2r.pt: https://www.f2r.pt/es/ICO001

f2r. (19 de Junio de 2021). *Odómetro ICO Rallye MAX G de la marca ICO Racing*. Obtenido de f2r.pt: https://www.f2r.pt/es/ICO002

Garmin. (19 de Junio de 2021). *Navegador portátil Garmin Montana 680.* Obtenido de buy.garmin.com: https://buy.garmin.com/es-ES/ES/p/523643

LLamas, L. (10 de Abril de 2018). *Comparativa ESP8266 frente a ESP32, los SoC de Espressif para IoT.* Obtenido de www.luisllamas.es: www.luisllamas.es/comparativa-esp8266-esp32/

Llamas, L. (17 de Abril de 2019). *Qué es y como funciona el protocolo MQTT.* Obtenido de www.luisllamas.es: https://www.luisllamas.es/que-es-mqtt-su-importancia-como-protocolo-iot/

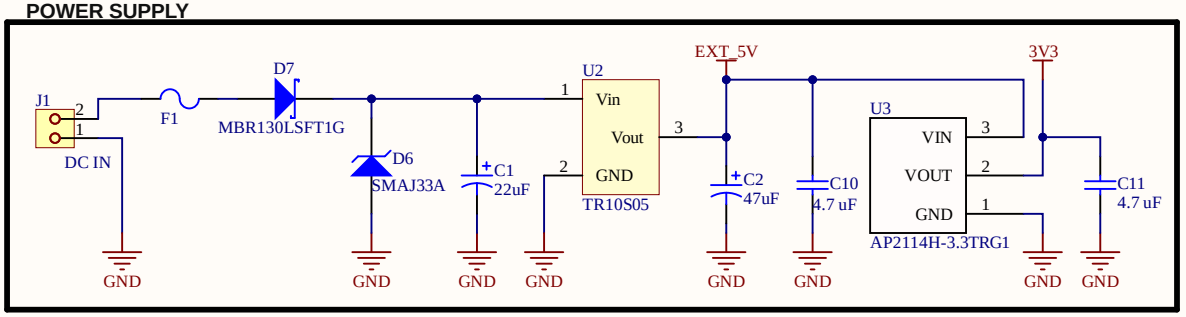
Marlink. (19 de Junio de 2021). *Alquiler de Iritrack V3.* Obtenido de eshop.marlink.com: https://eshop.marlink.com/sp/newiritrack-rental-on-dk.html

Wikipedia. (25 de Diciembre de 2020). *Características de FreeRTOS.* Obtenido de es.wikipedia.org: https://es.wikipedia.org/wiki/FreeRTOS

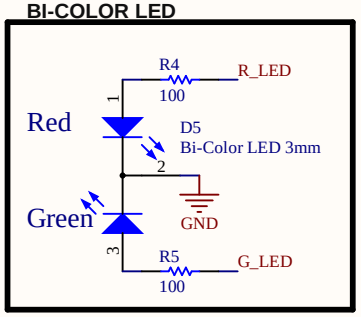
Wikipedia. (23 de Junio de 2020). *Funcionamiento de un fusible rearmable PPTC.* Obtenido de es.wikipedia.org: https://es.wikipedia.org/wiki/Fusible_rearmable

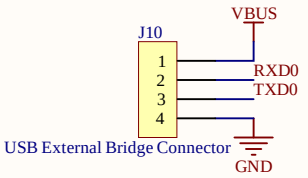
Anexo I: Esquemático, PCB y BOM.

POWER SUPPLY

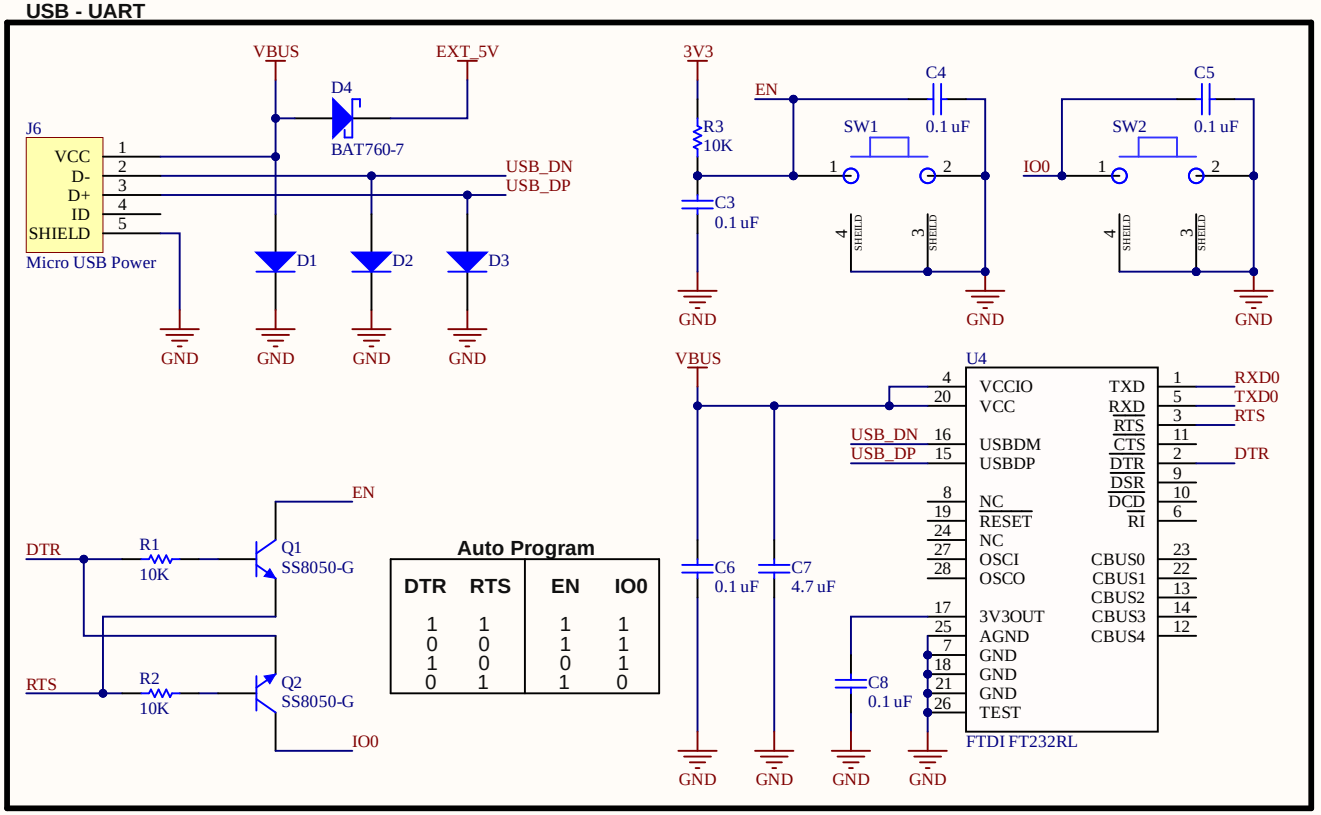


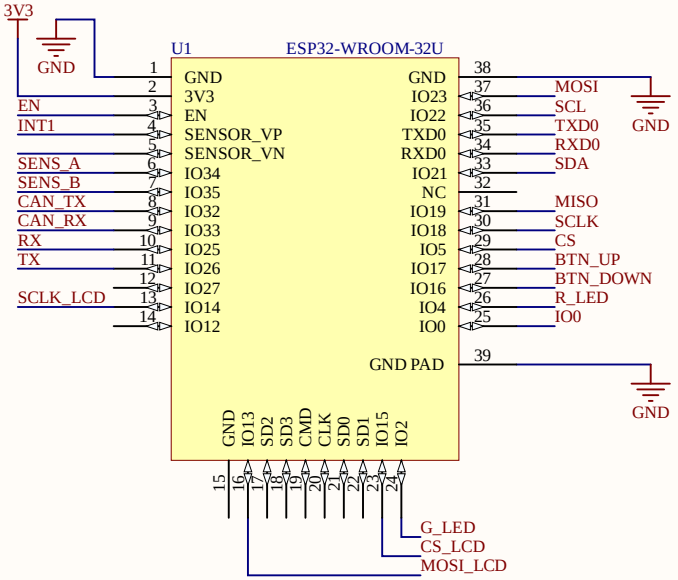
BI-COLOR LED





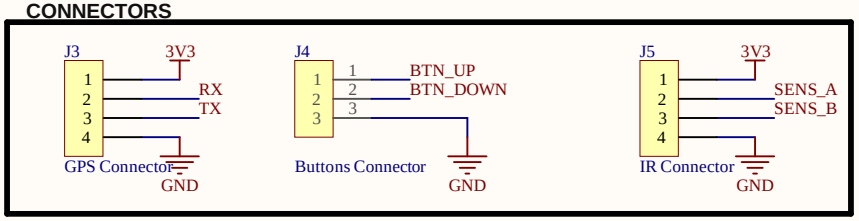
USB - UART



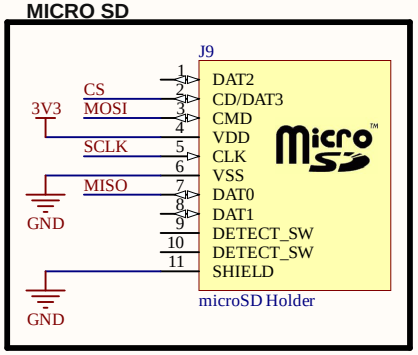


Title	SmartCAP Rally Computer: Main Circuit		
Size	Number		Revision
A4			v1.0
Date:	6/15/2021	Sheet 1 of 3	
File:	Main_ESP32.SchDoc	Drawn By: Andrés Mallada	

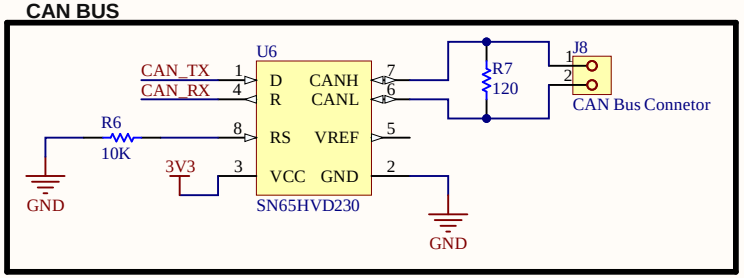
CONNECTORS



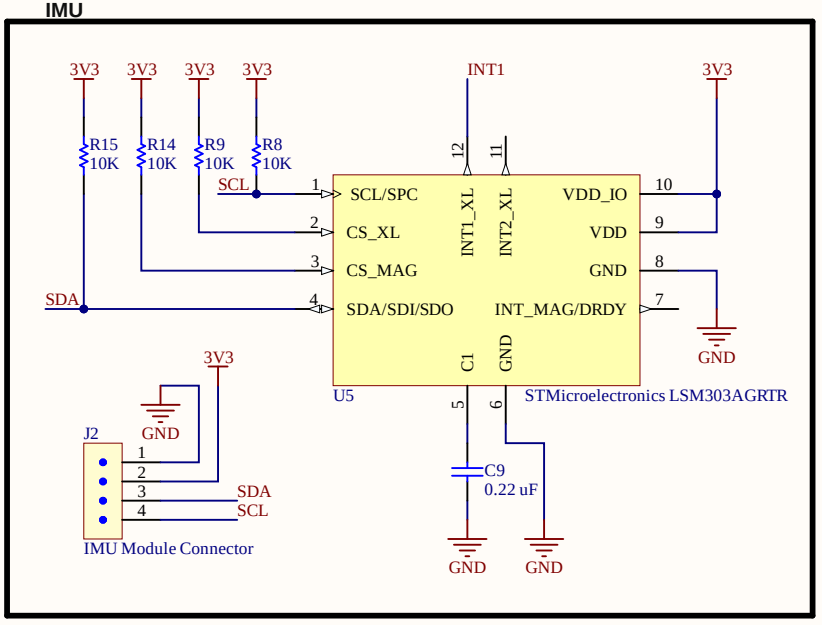
MICRO SD

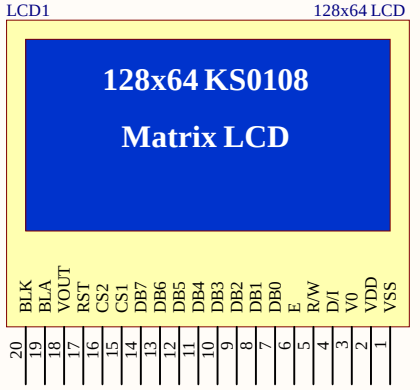


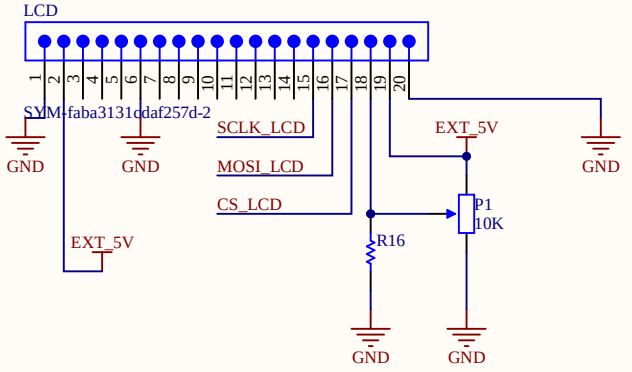
CAN BUS



IMU

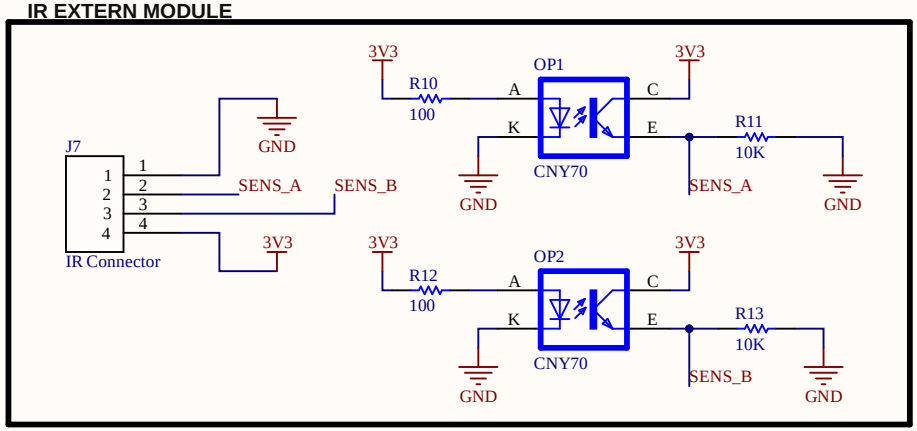




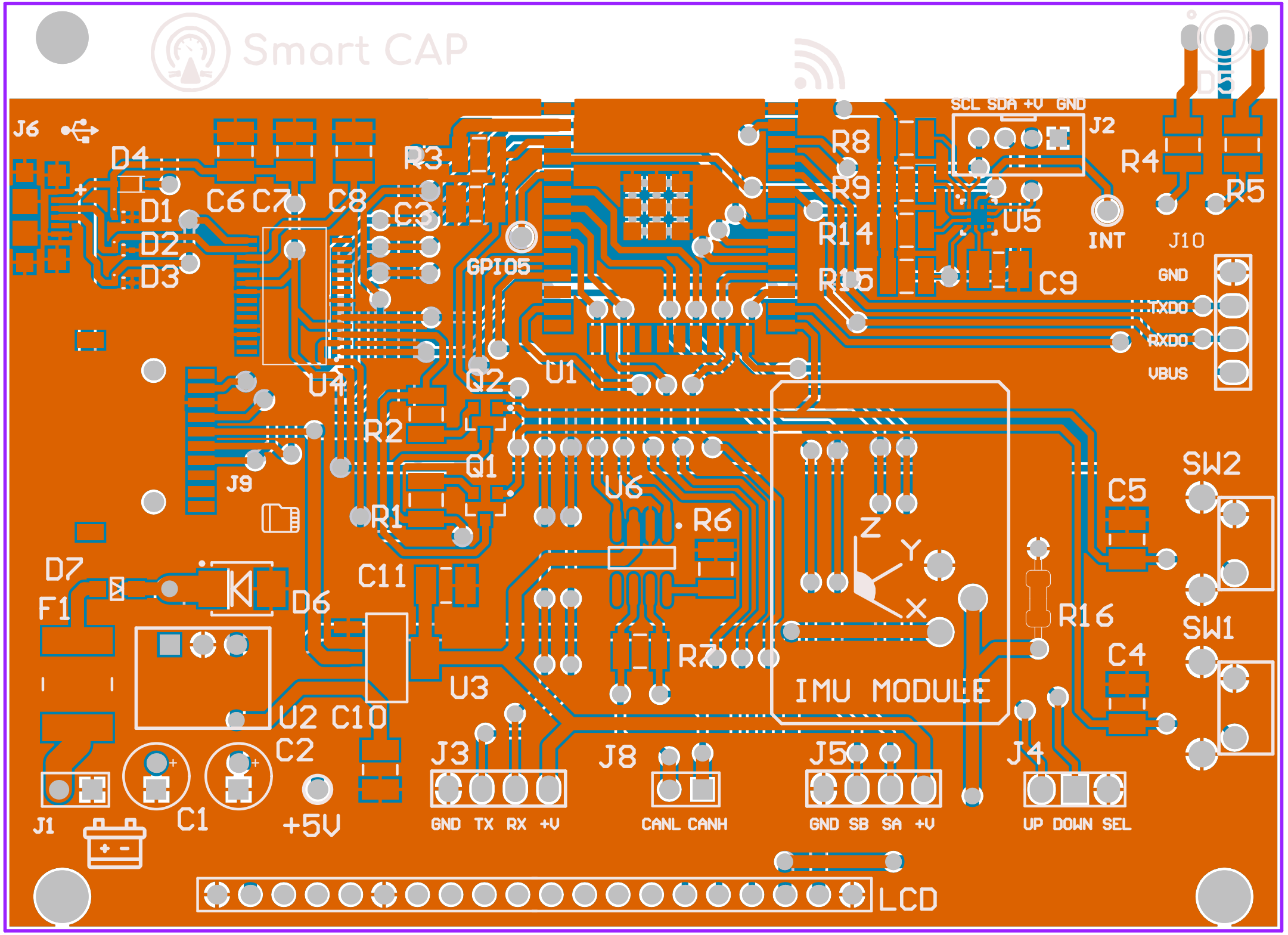


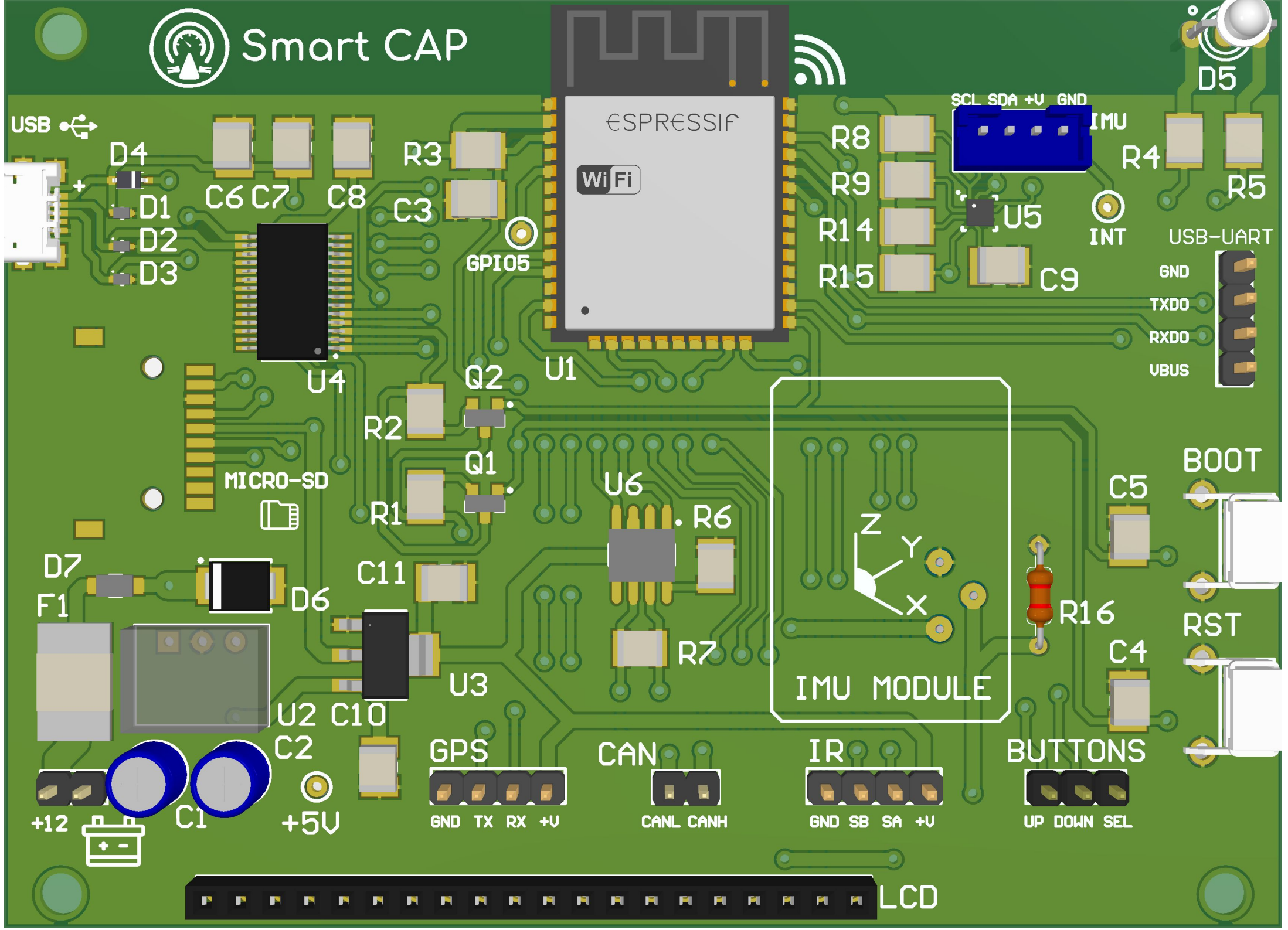
Title: SmartCAP Rally Computer: Peripherals		
Size: A4	Number:	Revision: v1.0
Date: 6/15/2021		Sheet 2 of 3
File: Sensors.SchDoc		Drawn By: Andrés Mallada

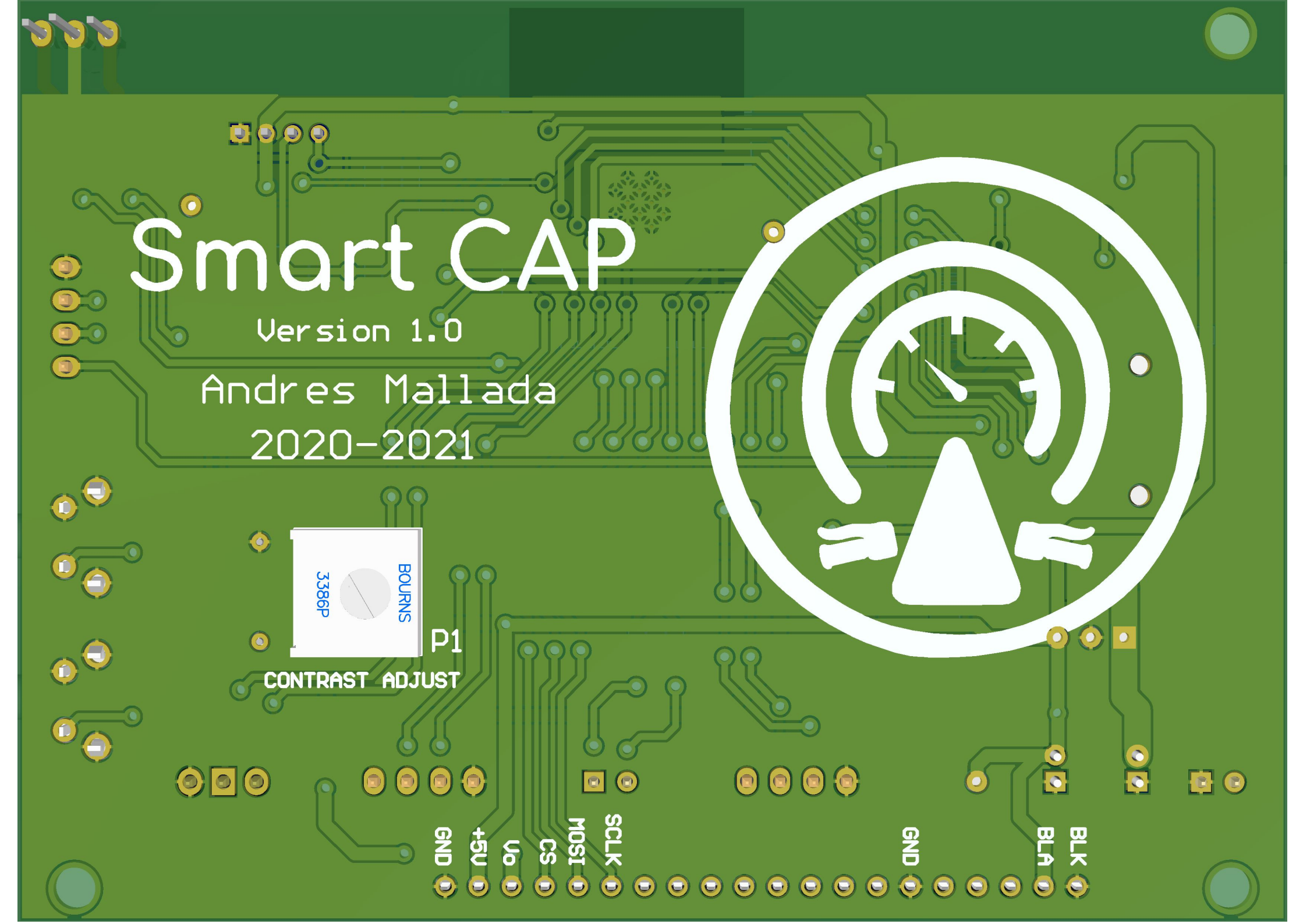
IR EXTERN MODULE

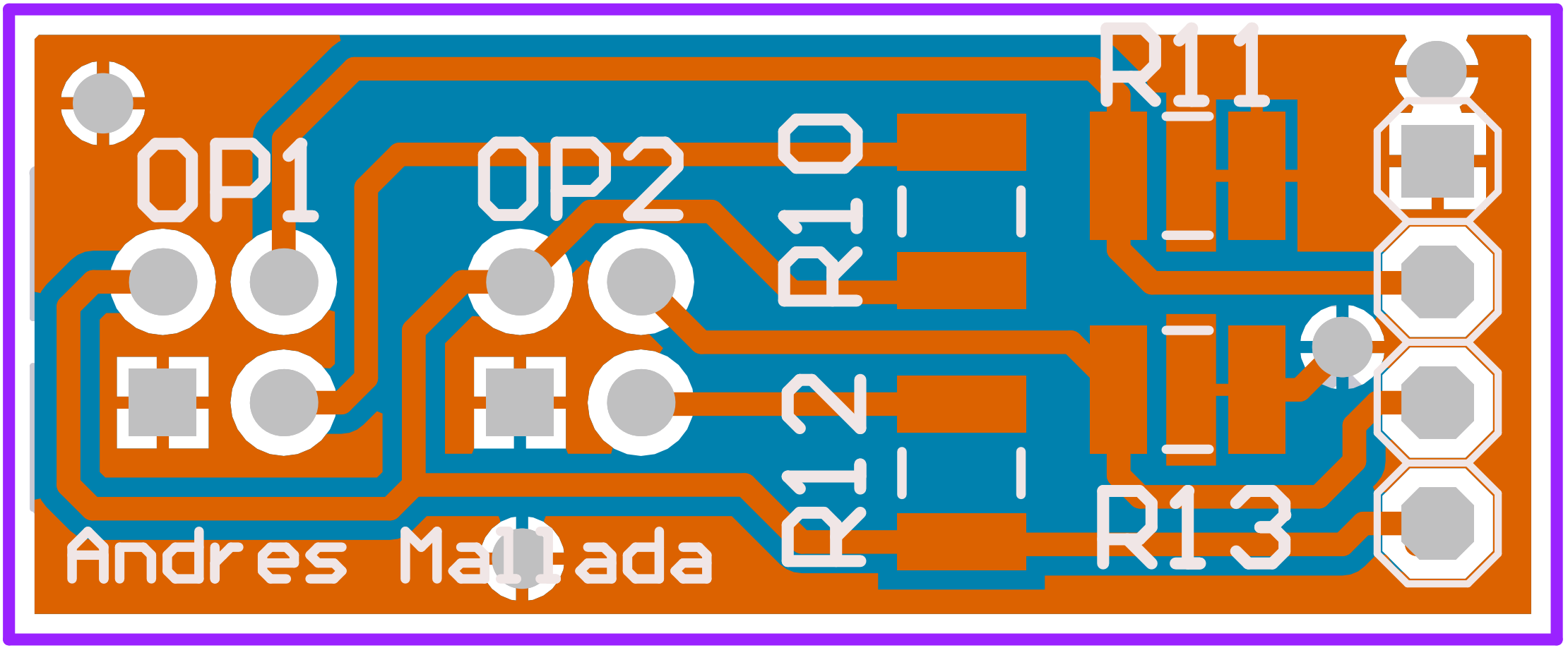


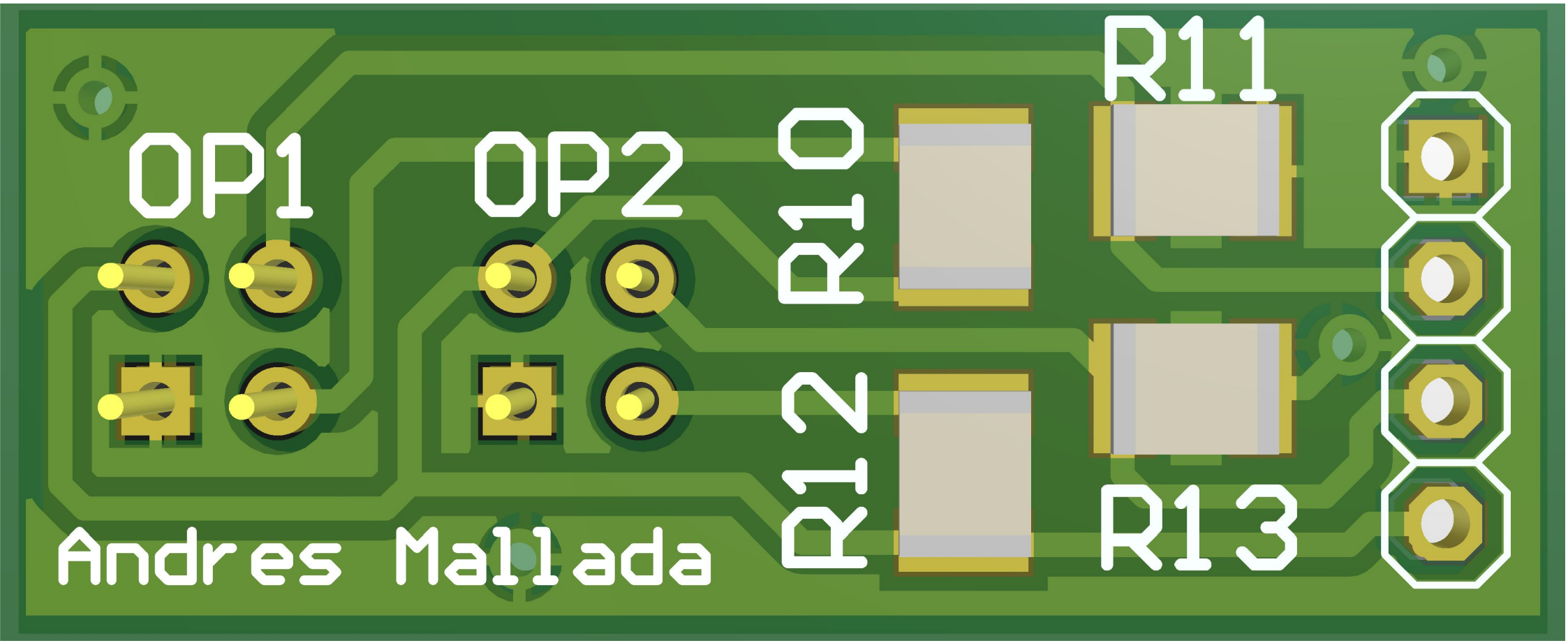
Title: SmartCAP Rally Computer: IR Module		
Size: A4	Number:	Revision: v1.0
Date: 6/15/2021		Sheet 3 of 3
File: IR_Module.SchDoc		Drawn By: Andrés Mallada

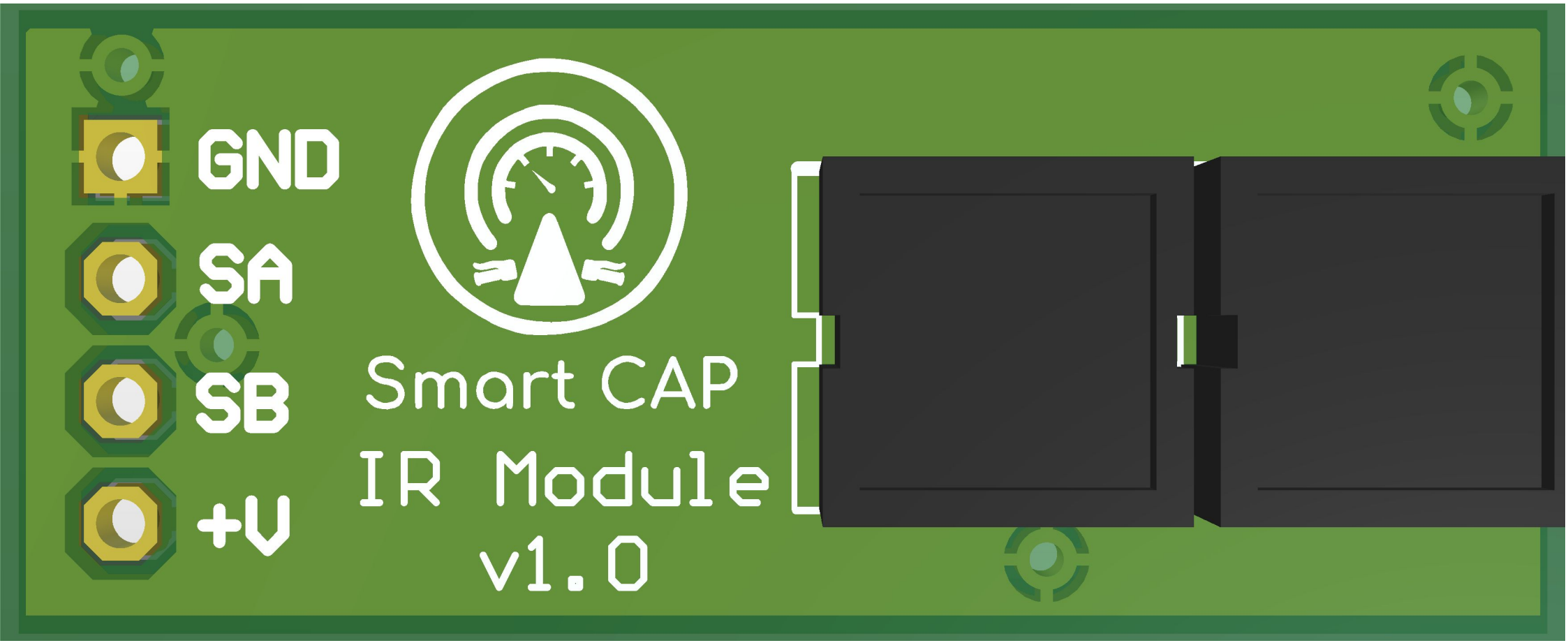

Smart CAP
J6
D4
D1
D2
D3
C6
C7
C8
R3
C3
GPIO5
U4
Q2
U1
R2
Q1
J9
R1
U6
R6
D7
F1
D6
C11
R7
U3
U2
C10
C2
C1
+5V
J1
J3
GND TX RX +V
J8
CANL CANH
J5
GND SB SA +V
J4
UP DOWN SEL
LCD
IMU MODULE
Z
Y
X
R8
R9
R14
R15
SCL SDA +V GND
J2
U5
INT
C9
R4
R5
J10
GND
TXD0
RXD0
VBUS
C5
SW2
R16
C4
SW1


Smart CAP
ESPRESSIF
WiFi
USB
D4
D1
D2
D3
C6
C7
C8
R3
C3
GPIO5
U4
U1
Q2
R2
Q1
R1
MICRO-SD
U6
R6
R7
C11
U3
D7
F1
D6
U2
C10
C2
C1
+12
+5V
GPS
GND TX RX +V
CAN
CANL CANH
IR
GND SB SA +V
BUTTONS
UP DOWN SEL
LCD
IMU MODULE
Z
Y
X
SCL SDA +V GND
IMU
U5
INT
C9
R8
R9
R14
R15
D5
R4
R5
USB-UART
GND
TXD0
RXD0
VBUS
BOOT
RST
C5
C4
R16


Smart CAP
Version 1.0
Andres Mallada
2020-2021
BOURNS
3386P
P1
CONTRAST ADJUST
GND
+5V
Vo
CS
MOSI
SCLK
GND
BLA
BLK


OP1
OP2
R10
R11
R12
R13
Andres Maldonada


OP1
OP2
R10
R11
R12
R13
Andres Mallada


GND
SA
SB
+V
Smart CAP
IR Module
v1.0

Bill Of Materials (BOM)

Designator	Quantity	Value	Manufacturer	Manufacturer Part Number	Vendor	Vendor Link	Price	Total Price	Price +1000	Total Price +1000
C1	1	22uF	Panasonic	ECA-1JM220	Farnell	https://es.farnell.com/panasonic/eca1jm220/conden-22-f-6	0,25 €	0,25 €	0,06 €	59,50 €
C2	1	47uF	Panasonic	EEUEB1H470S	Farnell	https://es.farnell.com/panasonic/eeueb1h470s/conden-47-	0,39 €	0,39 €	0,13 €	130,00 €
C3, C4, C5, C6, C8	5	0.1 uF	AVX	12101C104KAT2A	Farnell	https://es.farnell.com/avx/12101c104kat2a/conden-0-1-f-1	0,23 €	1,13 €	0,09 €	93,20 €
C7	1	4.7 uF	AVX	12101C475K4T2A	Farnell	https://es.farnell.com/avx/12101c475k4t2a/conden-4-7-f-1	0,90 €	0,90 €	0,24 €	240,00 €
C9	1	0.22 uF	Vishay	VJ1210Y224KXBAT	Farnell	https://es.farnell.com/vishay/vj1210y224kxbat/conden-0-2	0,44 €	0,44 €	0,26 €	261,00 €
D1, D2, D3	3		On Semiconductor	ESD5Z5.0T1G	Farnell	https://es.farnell.com/on-semiconductor/esd5z5-0t1g/esd-	0,15 €	0,44 €	0,15 €	146,00 €
D4	1		Diodes Incorporated	BAT760-7	Farnell	https://es.farnell.com/diodes-inc/bat760-7/diodo-rectifica	0,44 €	0,44 €	0,24 €	238,00 €
F1	1		Littlefuse	2920L075/60MR	Farnell	https://es.farnell.com/littelfuse/2920l075-60mr/resettable	1,10 €	1,10 €	0,34 €	335,00 €
J6	1		Amphenol Icc	10103594-0001LF	Farnell	https://es.farnell.com/amphenol-icc-fci/10103594-0001lf/m	0,68 €	0,68 €	0,36 €	358,00 €
J9	1		Hirose Connector	DM3AT-SF-PEJM5(11)	Farnell	https://es.farnell.com/hirose-hrs/dm3at-sf-pejm5/connecto	1,88 €	1,88 €	1,29 €	1.290,00 €
LCD1	1		AZDelivery	LCD Display 128 x 64	Amazon	https://www.amazon.es/AZDelivery-Display-Pantalla-Carac	10,99 €	10,99 €	8,99 €	8.990,00 €
OP1, OP2	2		Vishay	CNY70	Farnell	https://es.farnell.com/vishay/cny70/sensor-ptico-salida-tra	0,80 €	1,60 €	0,72 €	724,00 €
Q1, Q2	2		Comchip Technology	BC818-40LT1G	Farnell	https://es.farnell.com/on-semiconductor/bc818-40lt1g/tra	0,13 €	0,27 €	0,06 €	58,80 €
R1, R2, R3, R6, R8, R9, R11, R13, R14, R15	10	10K	Vishay	CRCW121010K0FKEA	Farnell	https://es.farnell.com/vishay/crcw121010k0fkea/res-10k-1	0,24 €	2,38 €	0,03 €	34,40 €
R4, R5, R10, R12	4	100	Vishay	CRCW1210100RFKEA	Farnell	https://es.farnell.com/vishay/crcw1210100rfkea/res-aec-q2	0,16 €	0,65 €	0,06 €	57,10 €
R7	1	120	Vishay	CRCW1210120RFKEA	Farnell	https://es.farnell.com/vishay/crcw1210120rfkea/res-120r-1	0,24 €	0,24 €	0,04 €	37,10 €
SW1, SW2	2		TE Connectivity	1-1825027-1	Farnell	https://es.farnell.com/te-connectivity/1-1825027-4/interr-p	0,16 €	0,31 €	0,15 €	153,00 €
U1	1		Espressif Systems	ESP32-WROOM-32U	DigiKey	https://www.digikey.es/product-detail/es/espressif-system	3,33 €	3,33 €	3,33 €	3.330,00 €
U2	1		XP Power	TR10S05	Farnell	https://es.farnell.com/xp-power/tr10s05/convertidor-dc-dc	4,22 €	4,22 €	3,54 €	3.540,00 €
U3	1		Diodes Incorporated	AP2114H-3.3TRG1	Farnell	https://es.farnell.com/diodes-inc/ap2114h-3-3trg1/ldo-fijo	0,28 €	0,28 €	0,11 €	107,00 €
U4	1		FTDI	FT232RL	Farnell	https://es.farnell.com/ftdi/ft232rl-reel/ic-usb-a-uart-smd-2	3,92 €	3,92 €	2,56 €	2.560,00 €
U5	1		STMicroelectronics	LSM303AGRTR	Farnell	https://es.farnell.com/stmicroelectronics/lsm303agrtr/mem	2,24 €	2,24 €	2,29 €	2.290,00 €
U6	1		Texas Instruments	SN65HVD230QDG4Q1	Farnell	https://es.farnell.com/texas-instruments/sn65hvd230qdg4q	4,51 €	4,51 €	1,90 €	1.900,00 €
D1	1		Kingbright	L-3VEGW	Farnell	https://es.farnell.com/kingbright/l-3vegw/led-red-grn-30-40	0,22 €	0,22 €	0,22 €	219,00 €
Condensadores Botones Mando	2	1uF	Vishay	K105K20X7RF5TH5	Farnell	https://es.farnell.com/vishay/k105k20x7rf5th5/conden-1-f	0,66 €	1,32 €	0,27 €	273,00 €
Diodos Botones Mando	2	1N4148	Multicomp Pro	1N4148	Farnell	https://es.farnell.com/multicomp/1n4148-do-35/diodo-peq	0,04 €	0,07 €	0,01 €	12,90 €
3 Pin Female M8 Cable Connector	2				Aliexpress	https://es.aliexpress.com/item/32841370102.html	1,49 €	2,98 €	1,49 €	1.490,00 €
3 Pin Male M8 Cable Connector	3				Aliexpress	https://es.aliexpress.com/item/32841370102.html	1,49 €	4,47 €	1,49 €	1.490,00 €
4 Pin Female M8 Cable Connector	2				Aliexpress	https://es.aliexpress.com/item/32841370102.html	1,55 €	3,10 €	1,55 €	1.550,00 €
4 Pin Male M8 Cable Connector	2				Aliexpress	https://es.aliexpress.com/item/32841370102.html	1,55 €	3,10 €	1,55 €	1.550,00 €
Water Resistant Button	3				Aliexpress	https://es.aliexpress.com/item/681794061.html	0,26 €	0,78 €	0,26 €	260,00 €
GPS Module	1		uBlox	NEO-M8N	Aliexpress	https://es.aliexpress.com/item/4000253063711.html	13,95 €	13,95 €	13,95 €	13.950,00 €
Adafruit LSM303AGR Module	1		Adafruit	4413	Amazon	https://www.amazon.es/Adafruit-LSM303AGR-Aceler%C3%	7,58 €	7,58 €	7,58 €	7.580,00 €
IR Module PCB	1		JLCPCB		JLCPCB	https://jlcpcb.com/	0,50 €	0,50 €	0,50 €	504,00 €
Main PCB	1		JLCPCB		JLCPCB	https://jlcpcb.com/	0,50 €	0,50 €	0,50 €	504,00 €
Carcasa y sujecciones	1						5,00 €	5,00 €	5,00 €	5.000,00 €
							TOTAL	**86,16 €**	**TOTAL**	**61.315,00 €**

Anexo II: Librerías utilizadas y licencias.

Para facilitar las tareas de desarrollo del firmware sin tener que profundizar a programación de bajo nivel se utilizaron diferentes licencias que para algunas de las principales funcionalidades del sistema.

El hecho de usar librerías de terceros implica que se debe prestar especial atención al tipo de licencia de uso sobre el que está recogida dicha librería, pues en ciertos casos podría obligar a hacer libre el código implementado para nuestro sistema. Este tipo de licencias reciben coloquialmente el nombre de víricas pues "infectan" el código implementado teniendo que liberarlo como código libre.

Todas las librerías utilizadas en este proyecto han sido obtenidas a través de la publicación original de cada autor en la plataforma GitHub. A final de este anexo se describen las principales características de las licencias bajo las que están las librerías utilizadas en este proyecto, así como un listado en el que se detallan todas las librerías utilizadas y el uso que se les ha dado.

Como se puede comprobar, tan solo hay una librería que supone un problema en cuanto a la liberación de código, en concreto aquella que está sujeta a la licencia GNU-GPL. Es el caso de la librería "ArduinoHelpers", utilizada para simplificar las tareas de control de las pulsaciones de los botones.

Para futuras versiones del producto y si procede, su comercialización, esto no presentará ningún inconveniente pues se implementará una librería propia para el control de los botones, de forma análoga a como se ha hecho con la librería "Stopwatch".

Librería: Adafruit LIS2MDL
Autor: Adafruit Industries
Versión: 2.1.3
Uso: Obtención de datos del magnetómetro
Licencia: BSD License
Header: Adafruit_LIS2MDL.h
Enlace: https://github.com/adafruit/Adafruit_LIS2MDL

Librería: Adafruit LSM303 Accel
Autor: Adafruit Industries
Versión: 1.1.4
Uso: Obtención de datos del acelerómetro
Licencia: BSD License
Header: Adafruit_LSM303_Accel.h
Enlace: https://github.com/adafruit/Adafruit_LSM303_Accel

Librería: Arduino
Autor: Espressif Systems
Versión: 1.1.4
Uso: Núcleo Arduino para el ESP32
Licencia: LGPL-2.1 License
Header: Arduino.h
Enlace: https://github.com/espressif/arduino-esp32/blob/master/cores/esp32/Arduino.h

Librería: Arduino Helpers
Autor: Arduino Helpers
Versión: 2.0.0
Uso: Control de pulsaciones de los botones
Licencia: GNU License
Header: Arduino_Helpers.h
Enlace: https://github.com/tttapa/Arduino-Helpers

Librería: ArduinoJson
Autor: Benoit Blanchon
Versión: 6.17.3
Uso: Serialización mensajes JSON
Licencia: MIT License
Header: ArduinoJson.h
Enlace: https://github.com/bblanchon/ArduinoJson

Librería: FS
Autor: Espressif Systems
Versión: 1.0.0
Uso: Sistema de ficheros para la SD
Licencia: LGPL-2.1 License
Header: FS.h
Enlace: https://github.com/espressif/arduino-esp32/tree/master/libraries/FS

Librería: Arduino GPX
Autor: Ryan Sutton
Versión: 0.0.1
Uso: Creación de fichero GPX
Licencia:
Header: GPX.h
Enlace: https://github.com/suttonr/Arduino-GPX-Library

Librería: ListLib
Autor: Luis Llamas
Versión: 1.0.0
Uso: Creación y gestión de arrays dinámicos
Licencia: Apache License
Header: ListLib.h
Enlace: https://github.com/luisllamasbinaburo/Arduino-List

Librería: STM32duino LSM303AGR
Autor: AST
Versión: 2.0.0
Uso: Lectura de registros del acelerómetro y del magnetómetro.
Licencia:
Header: LSM303AGR_ACC_Sensor.h / LSM303AGR_MAG_Sensor.h
Enlace: https://github.com/stm32duino/LSM303AGR

Librería: SD
Autor: Espressif Systems
Versión: 1.0.5
Uso: Control de la SD mediante SPI
Licencia: LGPL-2.1 License
Header: SD.h
Enlace: https://github.com/espressif/arduino-esp32/tree/master/libraries/SD

Librería: SPI
Autor: Espressif Systems
Versión: 1.0.0
Uso: Control del bus SPI
Licencia: LGPL-2.1 License
Header: SPI.h
Enlace: https://github.com/espressif/arduino-esp32/tree/master/libraries/SPI

Librería: Stopwatch
Autor: Andrés Mallada
Versión:
Uso: Cronómetro basado en reloj interno
Licencia:
Header: Stopwatch.h
Enlace:

Librería: ThingsBoard
Autor: ThingsBoard Team
Versión: 0.5.0
Uso: Acceso a la plataforma Thingsboard por MQTT
Licencia: MIT License
Header: ThingsBoard.h
Enlace: https://github.com/thingsboard/ThingsBoard-Arduino-MQTT-SDK

Librería: TinyGPS++
Autor: Mikal Hart
Versión: 1.0.2
Uso: Conversión y lectura de tramas NMEA (GPS)
Licencia: LGPL-2.1 License
Header: TinyGPS++.h
Enlace: https://github.com/mikalhart/TinyGPSPlus

Librería: tinyxml2
Autor: Lee Thomanson
Versión:
Uso: Conversión y lectura de archivos XML
Licencia: Zlib License
Header: tinyxml2.h
Enlace: https://github.com/leethomason/tinyxml2

Librería: U8g2lib
Autor: Olikraus
Versión: 2.29.3
Uso: Control del display LCD
Licencia: BSD License
Header: U8g2lib.h
Enlace: https://github.com/olikraus/u8g2

Librería: WiFi
Autor: Espressif Systems
Versión: 1.0.0
Uso: Librería WiFi para ESP32
Licencia: LGPL-2.1 License
Header: WiFi.h
Enlace: https://github.com/espressif/arduino-esp32/tree/master/libraries/WiFi

Librería: Wire
Autor: Espressif Systems
Versión: 1.0.1
Uso: Bus I2C para ESP32
Licencia: LGPL-2.1 License
Header: Wire.h
Enlace: https://github.com/espressif/arduino-esp32/tree/master/libraries/Wire

Licencia de software BSD

La licencia BSD cubre las distribuciones de software de Berkeley Software Distribution, además de otros programas. Ésta es una licencia considerada 'permisiva', ya que impone pocas restricciones sobre la forma de uso, alteraciones y redistribución del software. El software puede ser vendido y no hay obligaciones de incluir el código fuente. Esta licencia garantiza el crédito a los autores del software, pero no intenta garantizar que las modificaciones futuras permanezcan siendo software libre.
https://es.wikipedia.org/wiki/Licencia_BSD

Licencia de software LGPL 2.1

Estos programas se pueden distribuir bajo cualquier condición elegida si no se tratan de trabajos derivados. Un programa que no contiene derivado de ninguna porción de la biblioteca, pero está diseñado para trabajar con la biblioteca al ser compilado o enlazado con ella se denomina un "trabajo que usa la biblioteca". Dicho trabajo, por separado, no es un trabajo derivado de la biblioteca, y por tanto cae fuera del ámbito de esta Licencia.
https://es.wikipedia.org/wiki/GNU_Lesser_General_Public_License

Licencia de software Apache

La licencia Apache (Apache License o Apache Software License para versiones anteriores a 2.0) es una licencia de software libre permisiva creada por la Apache Software Foundation (ASF). La licencia Apache (con versiones 1.0, 1.1 y 2.0) requiere la conservación del aviso de derecho de autor y el descargo de responsabilidad, pero no es una licencia copyleft, ya que no requiere la redistribución del código fuente cuando se distribuyen versiones modificadas.
https://es.wikipedia.org/wiki/Apache_License

Licencia de software Zlib

Se trata de una licencia de software libre permisiva aprobada por Fundación de Software Libre como licencia de software libre y compatible con la Licencia Pública General GNU. De los términos de esta licencia se extrae el siguiente texto:
“Se concede permiso a cualquier persona para utilizar este software para cualquier propósito, incluidas aplicaciones comerciales, y para modificarlo y redistribuirlo libremente, sujeto a las siguientes restricciones: 1. No se debe tergiversar el origen de este software; no debe afirmar que ha escrito el software original. Si utiliza este software en un producto, se agradecería un reconocimiento en la documentación del producto, pero no es necesario. Las versiones de la fuente alterada deben estar claramente marcadas como tales y no deben tergiversarse como si fueran el software original. Este aviso no puede eliminarse ni modificarse de ninguna distribución de origen.”
https://es.vvikipedla.com/wiki/Zlib_License

Licencia de software MIT

La licencia MIT es una licencia de software que fue creada en el Instituto Tecnológico de Massachusetts y es posible usarla tanto para licenciar software libre como software no libre. Se encuentra entre las licencias compatibles con GNU-GPL y es muy parecida a la Licencia BSD. Es una Licencia de software libre permisiva lo que significa que impone muy pocas limitaciones en la reutilización y por tanto posee una excelente Compatibilidad de licencia. La licencia MIT permite reutilizar software dentro de Software propietario. Por otro lado, la licencia MIT es compatible con muchas licencias copyleft, como la GNU General Public License1 (software con licencia MIT puede integrarse en software con licencia GPL, pero no al contrario).

https://es.wikipedia.org/wiki/Licencia_MIT

Licencia de software GNU-GPL

Este tipo de licencia asegura que todo el material licenciado bajo la misma, esté disponible de forma libre para todos los usuarios de la aplicación o del software que cuente con este tipo de licencia.

Todos los programas informáticos que cuentan con licencia GNU, pueden ser libremente copiados, distribuidos, vendidos y hasta modificados por cualquier usuario, siempre y cuando mantengan el material informático bajo los mismos términos de la licencia GNU.

Por lo tanto, GPL es una licencia de tipo vírica, en cuanto usas algo con GPL, se propaga a todo el proyecto, que tiene que ser GPL también.

https://opensource.stackexchange.com/questions/4718/hardware-with-gpl-firmware

https://es.wikipedia.org/wiki/GNU_General_Public_License

Anexo III: Problemas encontrados y soluciones.

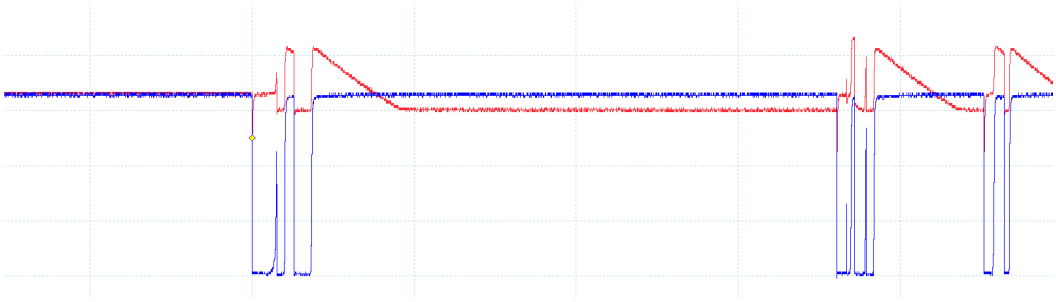
Durante el desarrollo del proyecto han surgido numerosos problemas relacionados tanto con el software, como con el firmware del dispositivo, sin embargo, se han conseguido solucionar todos ellos. A continuación se detallan aquellos que se han considerado más relevantes junto con la solución aportada.

▪ El comportamiento del sistema al pulsar los diferentes botones del mando no era el asignado o no se detectaban las pulsaciones.

Cuando se comenzó a implementar la navegación entre pantallas del dispositivo se comenzó creando un subproyecto más sencillo en el cual programar las funcionalidades de cada botón. Se empezó creando unas funciones de detección que se basaban en interrupciones, de esta manera se evitaba espera activa a la espera de que se produjera alguna pulsación. Sin embargo, el resultado no era el esperado y no se comportaba como se había descrito.

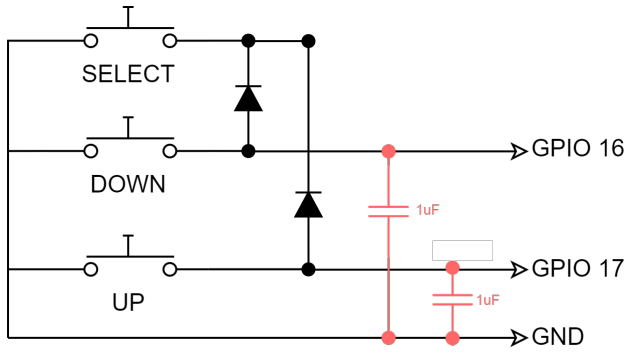
Fue por ello por lo que se decidió incorporar una librería de terceros para realizar dicho control. Se escogió la ya mencionada “Arduino Helpers” pues ofrecía distintos tipos de detección de pulsaciones tales como pulsación corta, pulsación larga, mantener pulsado, etc. además de integrar funciones para contrarrestar el debounce de los botones. No obstante, el sistema seguía sin funcionar correctamente.

Fue entonces cuando se planteó como un problema de hardware y no de software. Se conecto a los pines de entrada del mando un osciloscopio USB para poder realizar capturas de imagen, y al pulsar un botón se obtuvieron formas de onda como la siguiente.



Se debe recordar el circuito que integra el mando para controlar 3 botones con tan solo 3 cables, tal y como se describe en el apartado 3.1.4. lo que provocaba, como se puede ver en la anterior captura, que al pulsar el botón “DOWN” (azul) se producián una serie de rebotes tanto en el pin del botón “DOWN” como en el del botón “UP”. Y sucedía de igual forma al contrario. El problema residía en que dichos rebotes, oscilaban entre de los valores que los GPIO del ESP32 toman como referencia para establecer un nivel lógico alto o bajo.

La solución que se implantó fue introducir dos condensadores en paralelo en los pines de conexión del mando tal y como se indica en el siguiente esquemático:

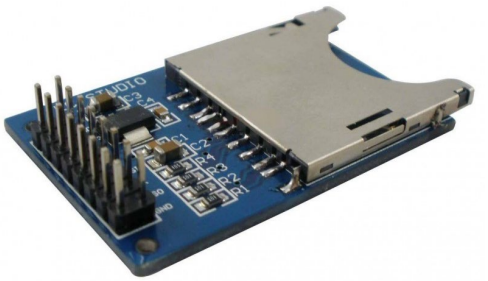


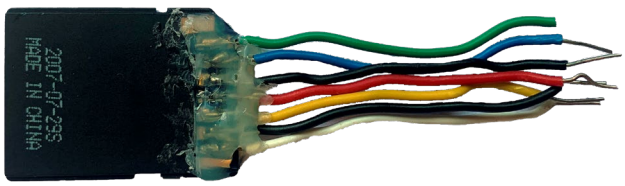
Se escogió el valor de 1uF de forma empírica, pues se realizaron pruebas con diferentes valores y este fue el que mejores resultados dio tanto en la visualización de la onda como en el funcionamiento del sistema.

▪ El sistema no era capaz de comunicarse con la tarjeta microSD.

Tras soldar correctamente el conector de la tarjeta microSD a la PCB, al introducir la tarjeta de memoria el sistema no era capaz de iniciar la comunicación SPI. Al principio se pensó que se trataba de un problema de programación, sin embargo, estaba todo correctamente definido.

También se planteó que se hubiera diseñado mal el circuito de comunicación, pues se había conectado directamente los GPIOs que implementan el bus SPI a los pines de la tarjeta, sin resistencias ni convertidores de nivel lógico pues tanto nuestro sistema como las memorias SD trabajan a 3.3V pero se tenía la inseguridad de que esto estuviera siendo la causa. Por ello se conectó un módulo SD externo directamente a los pines del ESP32 correspondientes, el cual funcionó correctamente con la memoria SD y se pudo seguir implementado el firmware del sistema hasta que se diera con la solución definitiva.



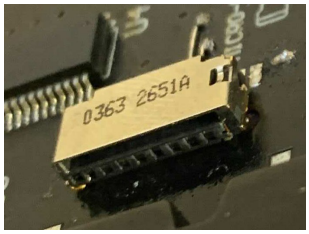


Dado que con el módulo externo que contenía componentes de regulación de nivel funcionaba, y el sistema debería funcionar sin ellos también, se fabricó un adaptador casero mediante un lecto microSD a SD, el cual se soldó directamente a los pines del ESP32 de igual forma que para el módulo externo anterior. De esta forma se establecía una conexión directa entre microSD y MCU. El resultado fue sorprendente pues la tarjeta SD funcionaba correctamente. De esta forma quedó descartado que el fallo fuera la falta de componentes de polarización y el circuito planteado desde el principio era correcto.

Por tanto, el fallo debía ser provocado por el conector soldado a la PCB, pues podía haberse dañado en la labores de soldadura. Se adquirió uno nuevo y se soldó, sin embargo, los resultados fueron los mismos y no se estableció comunicación.

Se conectó directamente mediante cables el conector microSD nuevo a los pines del ESP32, y la tarjeta volvía a dar error. Tras esta prueba se analizó rigurosamente el conector adquirido, y se detectó un fallo de fabricación que hacía que uno de los pines no entrara en contacto con la tarjeta de memoria y sucedía de igual forma con el primero que se adquirió.

Tras esto se decidió comprar un conector de otro tipo, en este caso que no fuera de extracción por muelle y que tuviera la separación entre pines compatible con la huella con la que se diseñó la PCB. Se introdujo la tarjeta en dicho conector y el sistema respondió correctamente pudiendo establecer comunicación con la memoria mediante el protocolo SPI.



Todas estas pruebas se realizaron en paralelo al desarrollo del firmware, pues mediante el módulo externo que se conectó directamente a los GPIOS se pudo seguir programando. Para la localización del fallo se necesitaron equipos de precisión y la inestimable ayuda de los maestros de taller, en concreto de Álvaro Graguera, debido a las reducidas dimensiones de los componentes y pistas, y la precisión necesaria para las diferentes soldaduras que se llevaron a cabo.

▪ El módulo GPS no se configuraba con las tasas y mensajes especificados.

Como ya se ha descrito en el apartado 3.1.5. se adquirió y adaptó un módulo GPS pensado para drones debido al elevado precio que presentaban los del principal fabricante. Se escogió uno que integraba el chip NEO M8N del fabricante uBlox por las prestaciones que ofrecía.

Se descargó del fabricante el software de diagnóstico uCenter y se conectó el módulo mediante un conversor USB-UART externo al ordenador. Mediante la interfaz del programa, se enviaron los mensajes correspondientes al módulo para desactivar las tramas que no eran necesarias y fijar la frecuencia a 5 Hz. El resultado fue el esperado al enviar las tramas de configuración, sin embargo, una vez se desconectaba el módulo, la configuración se perdía pues el módulo GPS volvía a enviar todas las tramas NMEA y a una frecuencia de 1 Hz.
Se investigó en foros y se consultó la página web de soporte del fabricante, y fue entonces cuando se descubrió un documento de uBlox (https://www.u-blox.com/en/counterfeit-products-and-u-blox-brand-misuse) en el que advertía de que había en el mercado copias piratas de sus módulos los cuales para reducir costes habían prescindido de algunos periféricos internos del encapsulado, como es el caso de una pequeña memoria flash que integran los originales para poder almacenar la configuración que se especifica al receptor.

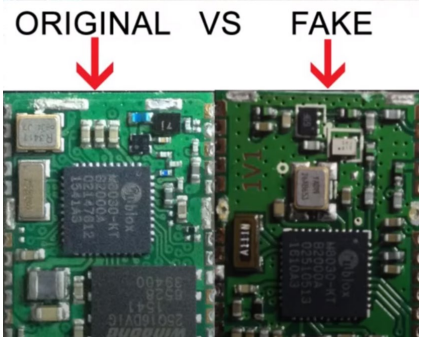
Tras esto se realizó un diagnóstico del receptor mediante la herramienta uCenter y se comprobó que no existía dicha memoria flash, como se puede observar en la siguiente captura.

UBX - LOG (Data Logger) - INFO (Log Info)

Param	Value	Units
Filestore capacity	0 (0 kB)	bytes
Maximum log size	0 (0 kB)	bytes
Current log size	0 (0 kB)	bytes
Entry count	0	
Oldest time	00:00:00 00/00/0	hh:mm:ss dd/mm/yyyy (UTC)
Newest time	00:00:00 00/00/0	hh:mm:ss dd/mm/yyyy (UTC)
Log status	10	hex
Status: log created	no	
Status: log recording	no	
Status: log is circular	no	

Como solución se pensó en configurar el módulo transmitiendo los comandos específicos a nivel de byte por el puerto serie cada vez que se iniciara el sistema. De esta forma al estar alimentado posteriormente a la configuración solucionábamos el problema de perder la configuración. Sin embargo, no se consiguió pues el receptor no interpretaba los comandos de esta forma.

Por ello, se decidió adquirir un nuevo módulo asegurándose de que integrara la memoria flash de la que prescindía el anterior. Se procedió a su configuración de igual forma que con el anterior módulo, con la particularidad de que en este sí que quedo almacenada la configuración en la memoria interna del módulo.



Tras diversas pruebas se comprobó que el receptor no se desprograma con el tiempo, y las tasas y los mensajes NMEA eran los especificados.

▪ El sistema no funcionaba correctamente al alimentar el dispositivo a través de la entrada de 12V.

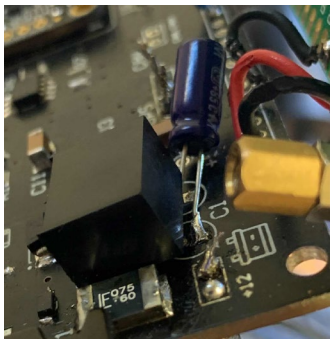
Durante el desarrollo del proyecto se utilizó la interfaz USB para programar y debuggear el sistema, y por tanto la alimentación del dispositivo provenía a su vez del bus USB del ordenador.

Para realizar las pruebas de alimentación externa se decidió conectar el dispositivo a una fuente de alimentación variable e ir incrementando paulatinamente el voltaje desde los 5V hasta los 12V para evitar dañar el sistema en caso de fallo. Sin embargo, conforme se aumentaba el voltaje de entrada lo hacía también en consecuencia la salida del regulador conmutado que debía estar fija a 5V. Esto provocaba que la pantalla LCD saturara los pixeles porque está alimentada a 5V, pero el resto del sistema funcionaba con normalidad pues el LDO presentaba cierto margen en sus voltajes de entrada (2.5V-6V).

Utilizando un multímetro se consiguió detectar que el componente que no funcionaba correctamente era el regulador conmutado que reduce el voltaje de alimentación a 5V. Se comprobaron los componentes contiguos, que protegen al circuito de sobretensiones y polarización inversa y todos funcionaban correctamente.

Fue entonces cuando se detectó que la huella de la PCB de este componente estaba invertida, y por tanto se había soldado al revés, la entrada en la salida y viceversa.

Debido a la disposición de componentes no se podía soldar al revés el regulador, por lo que se decidió eliminar el condensador C2 pues no era necesario controlar el transitorio de arranque para nuestro propósito, y de esta manera liberar espacio. Se desoldó a su vez el condensador C1 y se desplazó para dejar hueco al regulador, y finalmente se soldó el regulador con un ángulo de 90º. El resultado fue el que se muestra en la imagen, y no interfirió en el ensamblaje final del dispositivo.



▪ El dispositivo funcionaba con normalidad hasta que se intentaba conectar al punto de acceso, tras lo cual esporádicamente saltaba un error en tiempo de ejecución que hacía que el sistema se reiniciara.

La búsqueda de solución a este problema se prolongó hasta el final del desarrollo del firmware pues no se daba con la causa exacta. Se había comprobado que solo saltaba dicho error cuando el sistema trataba de conectarse al punto de acceso mediante la función `WiFi.begin()` de la librería WiFi del propio fabricante Espressif.

El mensaje de error que se mostraba cuando fallaba el sistema era el siguiente:

```
-> Guru Meditation Error: Core  1 panic'ed (Cache disabled but cached memory region accessed)
-> Core 1 register dump:
-> PC      : 0x4014b908  PS      : 0x00060034  A0      : 0x80081298  A1      : 0x3ffbed40
-> A2      : 0x3ffc187c  A3      : 0x00000001  A4      : 0x03bc7827  A5      : 0x00000000
-> A6      : 0x3ffc2a3c  A7      : 0xffffffff  A8      : 0x800811ba  A9      : 0x00000001
-> A10     : 0x3ffbac00  A11     : 0x3ffbed4c  A12     : 0x3ffba264  A13     : 0x0000abab
-> A14     : 0x3ffc2674  A15     : 0x3ffba264  SAR     : 0x0000001a  EXCCAUSE: 0x00000007
-> EXCVADDR: 0x00000000  LBEG    : 0x00000000  LEND    : 0x00000000  LCOUNT  : 0x00000000
-> Core 1 was running in ISR context:
-> EPC1    : 0x4008446c  EPC2    : 0x00000000  EPC3    : 0x00000000  EPC4    : 0x4014b908
->
-> ELF file SHA256: ad0bd0baad0bd0baad0bd0baad0bd0baad0bd0baad0bd0baad0bd0baad0bd0ba
->
-> Backtrace: 0x4014b908:0x3ffbed40 0x40081295:0x3ffbed70 0x4008215d:0x3ffbed90 0x40084469:0x3ffba190 0x400839b3:0x3ffba1b0 0x4008ead1
->
-> Rebooting...
```

Al buscar información sobre dicho error se averiguó que una ISR estaba ejecutando una función que no había sido puesta en la IRAM (Instruction RAM), que se usa para almacenar partes de la aplicación que deben ejecutarse desde la RAM. En el manual de referencia del ESP32 se especifica que los manejadores de interrupciones (handlers) deben colocarse en IRAM si se usa ESP_INTR_FLAG_IRAM al registrar el handler de interrupciones. Alternativamente, es posible especificar que cierta función se ubique en la IRAM en el código fuente usando la macro IRAM_ATTR.

Se pensaba que este error era causado por la librería WiFi, sin embargo, cuando se deshabilitaban ciertas tareas del sistema había casos en los que no saltaba dicho error. Tras numerosas pruebas con diferentes combinaciones de tareas desactivadas y activadas, se consiguió aislar el error, y resultó que estaba siendo provocado por la librería EspSoftwareSerial, la cual se utilizaba en la implementación del bus serie para la recepción de los datos GPS en los GPIOs especificados.

Se profundizó en el código de dicha librería y no se encontró la o las funciones internas que podía estar causando el problema. En consecuencia, se decidió no utilizar dicha librería y utilizar las funciones de puerto serie por hardware que implementa el propio ESP32 para la lectura de buses serie (Serial2). Se cambió el código para que funcionara con dicha clase y sus funciones, y el sistema funcionó con normalidad sin dar error alguno durante la ejecución.

Anexo IV: Referencias de consulta. Linkografía.

En este anexo se detallan las referencias y links consultados a lo largo del desarrollo del proyecto, así como el uso que se le han dado. Cabe destacar que en su mayoría se han utilizado como ejemplos de uso para familiarizarse con cada uno de los elementos que componen el sistema. De forman complementaria se han utilizado las API y webs oficiales de los componentes y librerías.

Se incluyen también los enlaces al código fuente del proyecto y a la plataforma web de visualización de datos (Thingsboard).

▪ Código fuente del proyecto:
https://gitlab.com/760331/smartcap_tfg

▪ Plataforma de visualización de datos:
http://tfg.howlab.es:8080/dashboard/199f84c0-a29a-11eb-87bf-9d02f11ee3eb?publicId=ae875a90-9958-11eb-87bf-9d02f11ee3eb

▪ API y tareas básicas con FreeRTOS:
https://www.freertos.org/a00106.html
https://savjee.be/2020/01/multitasking-esp32-arduino-freertos/

▪ Conexiones y ejemplos de uso del hardware utilizado con el ESP32:
ESP32:
https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/index.html
https://tecnotizate.es/esp32-mapeo-de-pines-y-sensores-internos/
https://desire.giesecke.tk/index.php/2018/07/06/reserved-gpios/
SD:
https://alexlubbock.com/micro-sd-adapter-esp8266-esp32
Display LCD:
https://www.instructables.com/ST7920-128X64-LCD-Display-to-ESP32/
IMU:
https://www.luisllamas.es/usar-arduino-con-los-imu-de-9dof-mpu-9150-y-mpu-9250/
https://learn.adafruit.com/lsm303-accelerometer-slash-compass-breakout/coding

▪ Programación del GPS y lectura de datos:
https://www.handheldgroup.com/knowledge-base/enable-or-disable-nmea-datasets-in-u-blox-gps-module/
https://andrea-toscano.com/u-blox-neo-m8n-u-center-configuration-and-arduino-parser-sketch/
http://aprs.gids.nl/nmea/

Anexo V: Manual de usuario

MANUAL DE USO

Smart CAP

Contenido de este manual

1. Introducción

Este es el manual de uso para el SmartCAP v1, un completo ordenador a bordo que sirve de asistente y guía para la navegación rally por roadbook. Aquí encontrará todo lo necesario para instalar y poner en funcionamiento su nuevo dispositivo.

Ante cualquier duda o sugerencia no dude en contactar con nosotros.

2. Contenido de la caja

La versión completa del kit SmartCAP incluye:

- Dispositivo SmartCAP
- Mando de control
- Cable de alimentación
- Módulo GPS
- Módulo detector de viñetas infrarrojo

NOTA

Tanto el **conector de alimentación** como los del mando de control y el módulo GPS, son **compatibles e intercambiables** con la mayoría de los sistemas del mercado como por ejemplo **ICO Racing**. Por lo tanto, si dispone de un mando y/o módulo GPS de otra marca que no sea SmartCAP **puede conectarlo a este sistema con total seguridad**.

3. Instalación del sistema

3.1. Montaje y alimentación

Smart CAP es un sistema basado en GPS por lo que no será necesario realizar ningún cableado al vehículo salvo para la alimentación.

Los tornillos de sujeción del dispositivo Smart CAP son de rosca M5 y tienen una separación de 38 mm entre ellos, lo cual le permite su instalación en la mayoría de soportes estándar pensados para otros dispositivos.

¡ATENCIÓN!

Preste especial atención a la hora de instalar el cable de alimentación a la batería del vehículo. La polaridad es la siguiente:

Negro: POSITIVO (+)

Azul: NEGATIVO (-)

En la siguiente imagen se muestra el diagrama general de conexión del SmartCAP y sus periféricos.

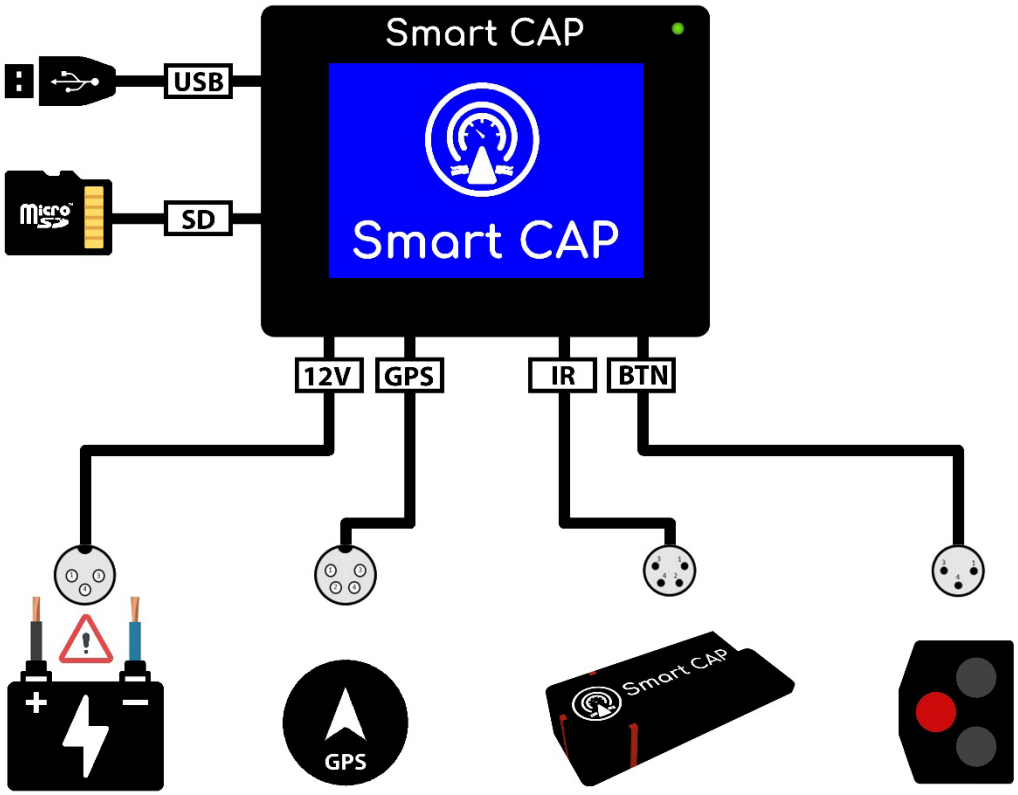


Fig. 1. Diagrama general de conexión.

3.2. Mando

Para instalar el mando de control debe seguir los siguientes pasos:

1. Desatornille los tornillos que cierran el agarre.
2. Coloque ambas partes que conforman el mando de manera que rodeen el manillar.
3. Apriete los tornillos hasta que el mando quede firme y sin holgura.
4. Conecte el cable a la entrada correspondiente del SmartCAP como se indica en Fig. 1.(conector de **3 pines macho**).

Fig. 2. Mando incluido en la caja.

NOTA
También puede aprovechar el mando que **ya tenga instalado** en su vehículo. Tan solo desconecte el mando de su antiguo equipo y siga las instrucciones del **paso 4**.

3.3. Módulo detector de viñetas infrarrojo

Para instalar el módulo detector de viñetas debe seguir los siguientes pasos:

1. Avance el roadbook hasta que sea visible la viñeta de configuración de umbrales en la parte superior del porta-roadbook.
2. Situé el módulo centrado sobre el patrón de la viñeta con ayuda de las líneas de guía rojas.
3. Conecte el cable a la entrada correspondiente del SmartCAP como se indica en Fig. 1.(conector de **4 pines macho**).
4. Configure los umbrales desde el menú de configuración y siga las instrucciones del asistente.

Fig. 3. Módulo infrarrojo detector de viñetas.

3.4. Módulo GPS

Para instalar el módulo GPS debe seguir los siguientes pasos:

1. Localice una ubicación en su vehículo que no obstaculice la recepción de la señal satélite.
2. Coloque el módulo receptor y asegúrelo con el adhesivo proporcionado y/o bridas.
3. Conecte el cable a la entrada correspondiente del SmartCAP como se indica en Fig. 1.(conector de **4 pines hembra**).
4. Encienda el sistema y espere a obtener una señal válida (**LED verde**).

Fig. 4. Receptor GPS.

NOTA
También puede aprovechar el recetor que **ya tenga instalado** en su vehículo. Tan solo desconecte el GPS de su antiguo equipo y siga las instrucciones a partir del **paso 3**.

4. Funcionamiento del sistema

Una vez realizada correctamente la instalación del dispositivo, así como la conexión de sus módulos y periféricos se puede empezar a usar el sistema con total normalidad.

4.1. Puesta en marcha

Para iniciar el SmartCAP tan solo debe conectar el dispositivo a una fuente de alimentación. Puede hacerlo de dos formas:

▪ Alimentando el sistema a 12 voltios a través del conector estándar de alimentación.

▪ Alimentando el sistema a 5 voltios mediante el conector microUSB situado en el lateral.

Una vez haya alimentado el sistema por alguna de las dos vías anteriormente descritas, se iniciará automáticamente.

En primer lugar, se muestra el logo del sistema y se inicializan todos los recursos internos. A los 3 segundos aparece la pantalla de **"Comprobación de hardware"**, la cual muestra al usuario información sobre el estado de los diferentes periféricos que componen el sistema para verificar su correcta conexión.

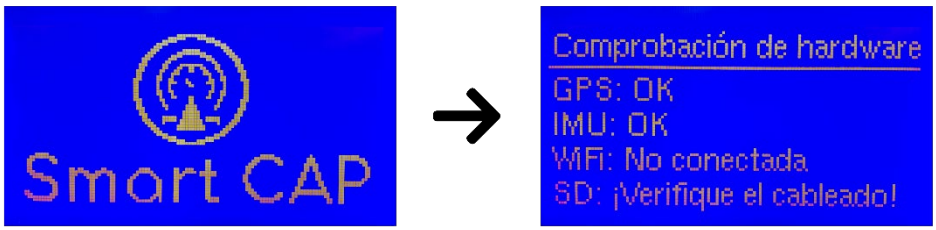


Fig. 5. Pantallas de inicialización del sistema.

Una vez se completa la inicialización del dispositivo y terminada la comprobación de hardware, el sistema se inicia y muestra la pantalla **"Odómetro"** del menú principal.

4.2. Pantallas principales

La navegación por las diferentes pantallas del menú principal es circular y se lleva a cabo mediante el botón **"Select"** del mando de control. Cada pulsación corta sobre dicho botón mostrará la siguiente pantalla. Una vez que se llegue a la última (Datos en tiempo real) se volverá a la primera de ellas (Odómetro).

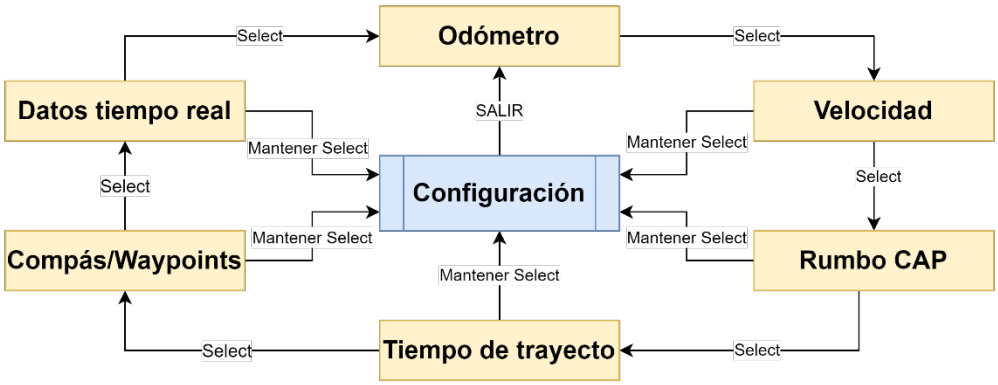


Fig. 6. Diagrama de navegación entre las pantallas principales.

A continuación se detalla la información y los controles en cada una de ellas:

• Odómetro

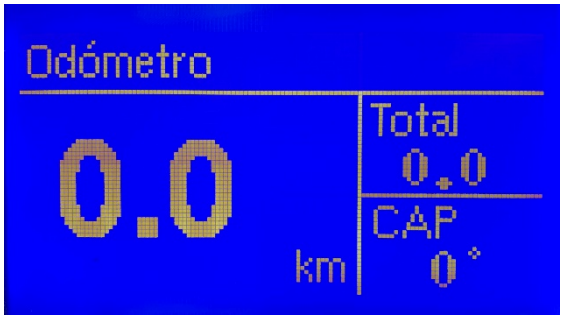


Fig. 7. Pantalla odómetro.

CONTENIDO:

▪ **Distancia parcial:** Ubicada en el centro. Contabiliza la distancia recorrida desde el ultimo reset de distancia parcial. Puede recalarse.

▪ **Distancia total:** Ubicada en el primer recuadro. Contabiliza la distancia recorrida desde el último reset de recorrido (desde el menú de configuración). **NO** puede recalarse. Se trata de la distancia total real recorrida durante la etapa.

▪ **Rumbo CAP:** Ubicado en el segundo recuadro. Indica el rumbo en grados sexagesimales.

CONTROLES:

▪ **Arriba:** Recala la distancia parcial incrementando su valor. Los incrementos se producen en función de la precisión establecida, 10 metros para dos decimales y 100 metros para un decimal. Si se mantiene pulsado incrementa rápidamente la distancia.

▪ **Abajo:** Recala la distancia parcial decrementando su valor. Los decrementos se producen en función de la precisión establecida, 10 metros para dos decimales y 100 metros para un decimal. Si se mantiene pulsado decrementa rápidamente la distancia.

▪ **Selección:** Si se pulsa brevemente, cambia a la siguiente pantalla. Si se mantiene presionado resetea la distancia parcial a 0. Este reset **NO** afecta a la distancia total.

• Velocidad

Fig. 8. Pantalla velocidad.

CONTENIDO:

▪ **Velocidad actual:** Ubicada en el centro. Muestra la velocidad a la que se circula en las unidades configuradas.

▪ **Velocidad media:** Ubicada en el primer recuadro. Muestra la velocidad media desde el último reset de recorrido (desde el menú de configuración).

NOTA
Tiene en cuenta el tiempo que no se está en movimiento, por lo que si se permanece mucho tiempo parado mostrará valor 0. Es útil para su uso en etapas de regularidad.

▪ **Velocidad máxima:** Ubicado en el segundo recuadro. Muestra la velocidad máxima desde el último reset de recorrido (desde el menú de configuración).

CONTROLES:

▪ **Arriba:** No asignado.

▪ **Abajo:** No asignado.

▪ **Selección:** Si se pulsa brevemente cambia, a la siguiente pantalla. Si se mantiene presionado muestra el menú de configuración.

• Rumbo CAP

Fig. 9. Pantalla rumbo CAP.

NOTA
El rumbo se calcula mediante vectores entre puntos de ubicación GPS, por lo que en caso de estar detenidos el valor mostrado no es válido.

CONTENIDO:

▪ **Rumbo CAP:** Ubicado en el centro. Indica el rumbo actual en grados sexagesimales.

CONTROLES:

▪ **Arriba:** No asignado.

▪ **Abajo:** No asignado.

▪ **Selección:** Si se pulsa brevemente, cambia a la siguiente pantalla. Si se mantiene presionado muestra el menú de configuración.

• Tiempo de trayecto

Fig. 10. Pantalla tiempo de trayecto.

CONTENIDO:

▪ **Cronómetro:** Ubicado en el centro. Indica el tiempo transcurrido en horas, minutos y segundos desde que se inició el temporizador.

CONTROLES:

▪ **Arriba:** Inicia/pausa el cronómetro.

▪ **Abajo:** Reinicia el cronómetro a 0.

▪ **Selección:** Si se pulsa brevemente, cambia a la siguiente pantalla. Si se mantiene presionado muestra el menú de configuración.

• Compás y guiado de waypoints

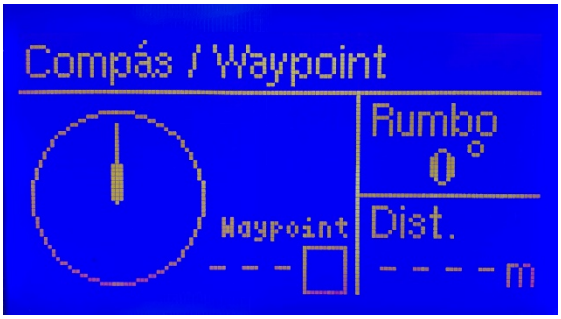


Fig. 11. Pantalla rumbo CAP.

Para que el sistema realice un guiado hacia waypoints, se debe cargar en la memoria del dispositivo un archivo GPX que contenga los waypoints hasta los que se desea realizar la navegación.

NOTA
El archivo debe tener el nombre de “waypoints” y ubicarse en la carpeta “waypoints” de la tarjeta microSD. En caso de que el sistema no detecte dicho fichero, la pantalla indicará el rumbo actual tanto en el primer recuadro como en el compás central.

CONTENIDO:

▪ **Rumbo hasta siguiente waypoint:** Ubicado en el primer recuadro. Indica el rumbo a seguir para llegar hasta el waypoint correspondiente.

▪ **Distancia hasta siguiente waypoint:** Ubicada en el segundo recuadro. Indica la distancia restante desde la ubicación actual hasta el waypoint correspondiente.

▪ **Compás:** Ubicado en el centro. Indica gráficamente la dirección hacia la que se encuentra el waypoint correspondiente.

▪ **Índice de waypoint:** Ubicado en el centro. Indica el número de waypoint hacia el cual se está navegando, así como si ha sido alcanzado (☑) o no (☐).

CONTROLES:

▪ **Arriba:** Avanza hasta el siguiente waypoint y muestra sus respectivas indicaciones.

▪ **Abajo:** Retrocede hasta el anterior waypoint y muestra sus respectivas indicaciones.

▪ **Selección:** Si se pulsa brevemente, cambia a la siguiente pantalla. Si se mantiene presionado muestra el menú de configuración.

• Datos en tiempo real

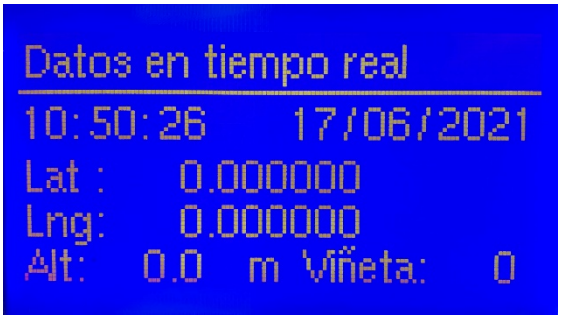


Fig. 12. Pantalla datos en tiempo real.

CONTENIDO:

▪ **Fecha y hora actual:** muestra la fecha y la hora obtenidas de la señal GPS. La hora se muestra con la corrección de uso horario especificada en el menú de configuración.

▪ **Latitud y longitud:** muestra las coordenadas de la ubicación actual con precisión de 6 decimales.

▪ **Altitud:** muestra la altitud actual sobre el nivel del mar.

▪ **Viñeta actual:** muestra la viñeta actual del roadbook detectada a través del módulo infrarrojo. Su valor puede corregirse como se indica a continuación.

CONTROLES:

▪ **Arriba:** Corrige el valor de viñeta actual incrementándolo en 1 unidad.

▪ **Abajo:** Corrige el valor de viñeta actual decrementándolo en 1 unidad.

▪ **Selección:** Si se pulsa brevemente, cambia a la siguiente pantalla. Si se mantiene presionado muestra el menú de configuración.

4.3. Configuración del sistema

El sistema SmartCAP puede configurarse de dos formas distintas. A continuación de describen cada una ellas.

4.3.1. Configuración mediante el menú de SmartCAP

Para entrar al menú de configuración del dispositivo se debe mantener pulsado el botón de selección en cualquiera de las pantallas principales a excepción de la de "Odómetro". Acto seguido se mostrará una lista en la que se encuentran los diferentes parámetros y opciones de configuración del dispositivo. A continuación, se describe cada una de ellas.

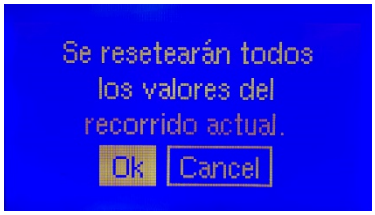
Fig. 13. Opciones del menú configuración.

NOTA

Los controles de navegación en el menú de configuración son diferentes a los del menú principal. Para avanzar entre las diferentes opciones se debe pulsar el botón **ABAJO.** Para seleccionar alguna de ellas de debe presionar el botón **ARRIBA.** Se trata de un menú circular por lo que al llegar a la última opción (Salir) se vuelve a la primera de ellas (Reset trayecto).

▪ Reset trayecto

Permite borrar todos los datos del recorrido actual para empezar uno nuevo. Se pondrán a cero los siguientes datos: distancia total, distancia parcial, velocidad media, velocidad máxima, tiempo de trayecto, waypoint actual y viñeta actual.



▪ Modo de uso

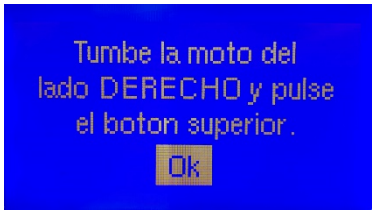
Permite seleccionar el modo de uso con el que se quiere utilizar el dispositivo. El modo **Tracking remoto** activa la conexión WiFi y el reporte de telemetría a la plataforma web. El modo **Offline** desactiva dicha funcionalidad. El modo escogido se guarda en memoria para el siguiente encendido del sistema al salir del menú de configuración.

▪ Configurar WiFi

Permite visualizar el estado de la conexión WiFi, y el SSID y la contraseña del punto de acceso con el que está configurado el sistema para conectarse. Esta opción solo es relevante si el modo de uso seleccionado es "Tracking remoto". Tanto el SSID como la contraseña deben especificarse en el archivo **wifi.txt** que se encuentra dentro de la carpeta **config** en la tarjeta de memoria.

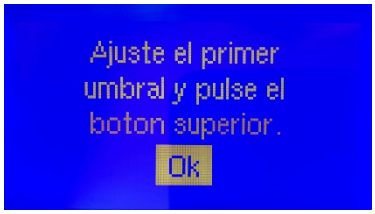
▪ Configurar caída

Permite establecer los valores de posición de la moto que serán interpretados como caída para el sistema. El asistente indica claramente el proceso que se debe seguir para establecer dichos umbrales. Los valores son almacenados en la memoria al salir del menú de configuración.

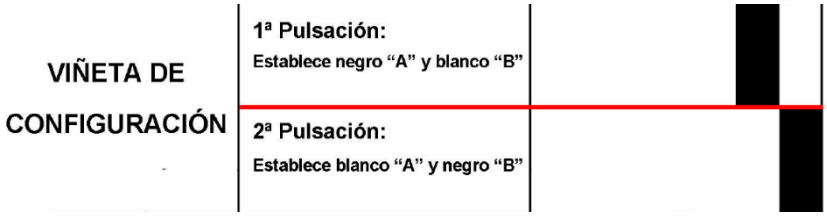


Una vez configurados los valores de caída el sistema comprobará continuamente si se ha producido un accidente. En caso de detectar una caída, se mostrará una alerta por pantalla y se notificará en la plataforma web en caso de estar activado el modo Tracking Remoto.

▪ Configurar IR



Permite establecer los umbrales de detección del patrón impreso en el roadbook. Este proceso debe realizarse cada vez que se inicie un nuevo recorrido pues las condiciones de iluminación pueden haber variado con respecto a la ultima configuración. El asistente indica claramente los pasos a seguir para configurar correctamente el sistema. La configuración debe realizarse sobre una viñeta de configuración como que se muestra a continuación.



▪ Registrador de ruta

Permite iniciar o detener el guardado de ruta en la tarjeta de memoria. Si se activa dicha opción, se crea un archivo GPX de nombre la fecha y hora actual, en el que se registra la ruta seguida por el piloto. Al detener el registro, se finaliza el archivo de ruta. Esta ruta puede visualizarse a través de algún software de terceros copiando el archivo de la tarjeta microSD a un equipo externo

▪ Unidades

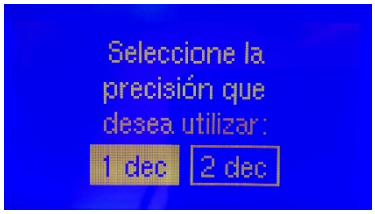
Permite seleccionar el sistema métrico en el cual se muestran los datos de distancias, velocidad y altitud. El sistema métrico utiliza kilómetros, kilómetros por hora y metros; y el sistema imperial millas, millas por hora y pies respectivamente.

▪ Idioma

Permite seleccionar el idioma en el que se muestran los textos y menús del sistema. Se puede seleccionar entre español, inglés, francés, portugués, alemán y catalán.

▪ Precisión

Permite seleccionar el número de decimales con el que se muestra la distancia parcial en la pantalla Odómetro. En el caso de 2 decimales se tiene una precisión de 10 metros, y en el caso de 1 decimal de 100 metros. Se puede cambiar de un número a otro sin perder el valor de distancia actual.

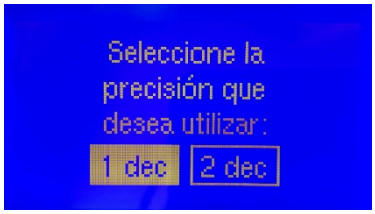


▪ Uso horario

Permite seleccionar el uso horario para efectuar la corrección horaria sobre la hora UTC reportada por el sistema GPS.

▪ Restaurar dispositvo

Devuelve el dispositivo al estado de fábrica. Borra todos los datos del recorrido actual, además de los umbrales y configuración del sistema. Al seleccionar dicha opción se muestra un mensaje de advertencia tras el cual se reinicia el dispositivo. Se recomienda hacer una copia de seguridad de los archivos de la tarjeta microSD antes de efectuar el proceso de restauración.



▪ Salir

Cierra el menú de configuración, guarda todos los datos modificados en la memoria microSD y muestra la pantalla del menú principal desde la que se accedió al menú de configuración.

NOTA

Para que los datos de configuración modificados se guarden en la memoria del sistema (tarjeta microSD), se debe seleccionar la opción **SALIR** del menú de configuración. Si se configura el sistema y se desconecta directamente sin salir del menú de configuración, los datos no se guardarán.

4.3.2. Configuración mediante tarjeta microSD

Los datos de configuración que han sido establecidos a través del menú de configuración del propio SmartCAP, son almacenados en la memoria microSD en formato JSON. Si se conecta dicha tarjeta a un equipo externo, se puede acceder a los archivos de configuración y modificararlos manualmente mediante un editor de textos, como por el ejemplo el "Bloc de notas" de Windows. A continuación se muestra un ejemplo de cada fichero.

config/config.txt


```
{
  "irConfigured": false,
  "fallConfigured": true,
  "deviceMode": 0,
  "deviceLanguage": 0,
  "useMetrical": true,
  "useTwoDecimals": true,
  "timeZoneOffset": 2,
  "irThA": 0,
  "irThB": 0,
  "rxFallTh": 2.830199,
  "ryFallTh": -8.299368,
  "rzFallTh": 3.595118,
  "lxFallTh": 2.485986,
  "lyFallTh": 9.4085,
  "lzFallTh": 0.803165
}
```


config/wifi.txt


```
{
  "ssid":"iPhone de Juan",
  "pass":"password"
}
```


data/init.txt


```
{
  "totalDistance": 0,
  "partialDistance": 0,
  "meanSpeed": 0,
  "maxSpeed": 0,
  "t_hours": 0,
  "t_minutes": 0,
  "t_seconds": 0,
  "vignette": 0,
  "isMetrical": true
}
```


▪ config.txt

En este fichero se guardan todos los parámetros de configuración de las diferentes funcionalidades del sistema, así como los umbrales de configuración del detector IR y de caídas.

NOTA
Se recomienda no modificar este archivo manualmente a no ser que tengan conocimientos básicos del formato de intercambio de datos JSON. Un error de escritura en el archivo podría provocar un fallo en el sistema.

▪ wifi.txt

En este fichero se especifican el SSID y la contraseña del punto de acceso al que se debe conectar el SmartCAP para reportar los datos a la plataforma. El valor especificados para ambos campos debe estar entrecomillado (""). El tamaño máximo para el SSID y la contraseña es de 32 y 64 caracteres respectivamente.

▪ init.txt

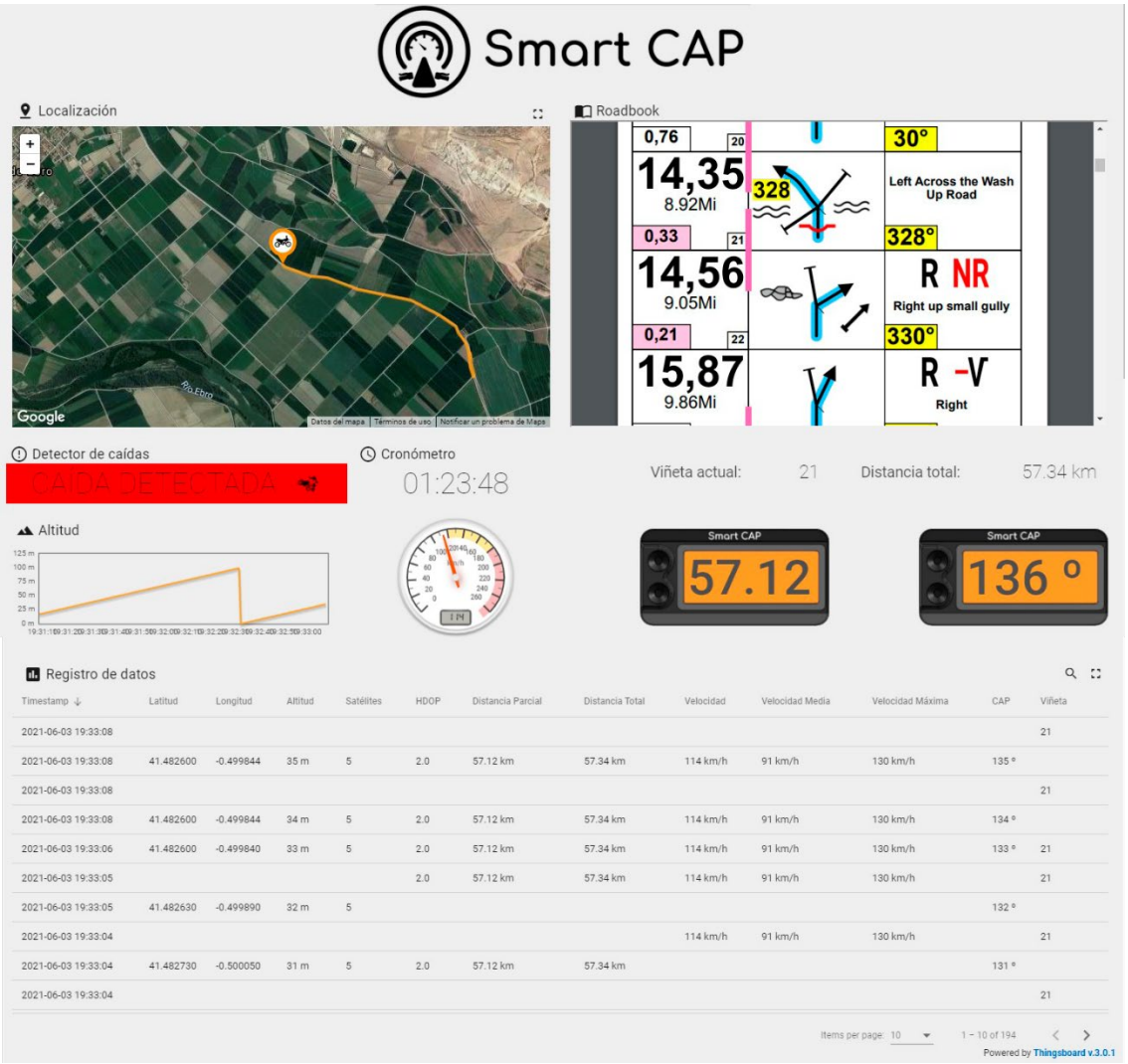
Este fichero contiene los valores del trayecto a los que se inicializará el SmartCAP al encenderse. Permite al usuario programar el sistema para iniciarse en valores concretos de un recorrido, ideal para repetir etapas a partir de puntos concretos.

El campo isMetrical hace referencia al sistema métrico en el que están referidos los valores (true = métrico, false = imperial).

5. Plataforma web y visualización de telemetría

Para acceder a la plataforma tan solo debe acceder al link que se le ha proporcionado, y asegurarse de que el SmartCAP está correctamente configurado y conectado al punto de acceso móvil (se recomienda utilizar un smartphone).

Una vez este conectado, el sistema comenzará la transmisión de datos a la plataforma y podrá visualizar desde cualquier navegador la telemetría del vehículo en el que se haya instalado el SmartCAP.



NOTA
Actualmente la visualización del roadbook en PDF está en una versión provisional. El archivo a visualizar no puede ser modificado y se corresponde con un roadbook de ejemplo.