E.T.S. de Ingeniería Industrial, Informática
y de Telecomunicación

Sistema HUD para el visionado de la velocidad desde el casco en motocicletas

Grado en Ingeniería
en Tecnologías Industriales

Trabajo Fin de Grado

Autor: Jose Javier Roldán Babiano

Director: Ángel María Andueza Unanua

Pamplona, 8 de junio de 2022

upna
Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

AGRADECIMIENTOS

Quiero agradecer, primeramente, el apoyo recibido por parte de mis padres, Carolina y Fernando, a lo largo de toda la carrera, que culmina con este proyecto. Gran parte de mi motivación a lo largo de todos estos años se ha mantenido gracias a ellos, y sin eso estoy seguro de que no habría llegado aquí.

También quiero agradecer el apoyo de mis amigos de siempre y los hechos en la carrera. Cerrar la biblioteca un sábado a la noche no se hace con cualquiera, y desde luego lo hemos hecho unas cuantas veces.

Finalmente me gustaría agradecer a los profesores que me han apoyado y animado, en especial a Ángel, mi tutor, quien desde que le propuse la idea para este TFG confió en mí y, sin saberlo, me dio ánimos cuando yo no veía claro el proyecto. No podría haber pedido un mejor tutor.

ABSTRACT

This work showcases the progress achieved in the development of a Heads Up Display for motorcyclists. It consists of a tiny translucent prism in front of the face shield where its speed is projected in real time, so the rider doesn't have to look to the dashboard (losing track of the road). This way helps to prevent accidents caused by this distractions. The speed data is obteined by a GPS module connected to a ESP32 microcontroller that processes the information and shows it on an OLED screen with a traslucent prism on it, so the rider can look through it in a way that its field of view isn't affected.

KEYWORDS

Motorcycling, Road safety, HUD, Arduino IDE, microcontroler, GPS.

RESUMEN

El presente proyecto muestra los avances conseguidos en el desarrollo de un dispositivo HUD (Heads Up Display) para motociclistas. Este consiste en un pequeño prisma transparente que se coloca delante de la visera del casco del motociclista en el que se proyecta su velocidad en tiempo real, de modo que la mirada no se tiene que desviar de la carretera al velocímetro para conocer esta información, mejorando su seguridad. La información se obtiene mediante un módulo GPS acoplado a un microcontrolador ESP32, que es quien procesa la información para luego ser transmitida a una pequeña pantalla OLED con un prisma encima, de manera que se proyecta en el mismo y así el usuario ve la información en el prisma, pero también a través de este, de modo que no interfiere en su campo de visión.

LISTA DE PALABRAS CLAVE

Motociclismo, seguridad vial, HUD, GPS, microcontrolador, Arduino.

1 ÍNDICE DE CONTENIDO

2 ÍNDICE DE FIGURAS

3 ÍNDICE DE TABLAS

1. INTRODUCCIÓN

En el ámbito de la automoción, la seguridad es una prioridad fundamental tanto para fabricantes como para usuarios. Y, de entre todos los tipos de automóviles, las motocicletas son el grupo más vulnerable, ya que el conductor apenas tiene protección en caso de accidente. Es por ello por lo que en este tipo de vehículos se potencia al máximo la seguridad activa, que son los elementos que contribuyen para evitar accidentes, como pueden ser las suspensiones, los neumáticos, la dirección… Aunque los elementos de seguridad pasiva (los que tratan de minimizar las consecuencias en caso de accidente) han avanzado mucho lo último años (creación del chaleco airbag, por ejemplo), en el caso de las motocicletas nunca podrán llegar al nivel de otro tipo de vehículos, como coches o camiones, ya que carecen de la integridad estructural para ello.

Los dispositivos electrónicos están cada vez más integrados en nuestro día a día, desde el smartphone hasta los relojes inteligentes, pasando por frigoríficos que te avisan cuando te faltan productos y hacen la compra por ti. Estos dispositivos se aplican en todo tipo de ámbitos, incluyendo el de la seguridad vial. Por una cuestión práctica se han implementado más en vehículos como coches y camiones (por ejemplo, cámaras que escanean el suelo y ajustan las suspensiones acordes a ello) que en motos, ya que disponen de un espacio mayor para su instalación y es más sencillo de diseñar. Irónicamente, esto deja una gran brecha de seguridad en el grupo más vulnerable en la carretera, las motos.

2. JUSTIFICACIÓN Y OBJETIVOS

La necesidad de mejorar la seguridad de los motociclistas en la carretera ha propiciado el desarrollo de este dispositivo HUD. El objetivo final es construir y testar un prototipo funcional en un ambiente controlado para comprobar la utilidad y funcionalidad de este. Los requisitos que ha de cumplir son:

- Mostrar la velocidad en tiempo real sin interrumpir el campo de visión
- Ser compacto
- Poder acoplarse y desacoplarse de cualquier casco
- Tener un coste asequible
- Poseer autonomía mínima de 6h (un día en moto aprox.)
- Ser fácilmente recargable

Puesto que se trata del desarrollo de un prototipo desde cero, primeramente, será necesario estudiar la viabilidad de este (ver si existen componentes y medios asequibles para llevarlo a cabo). Tras seleccionar los módulos que se van a usar, se comprobará su funcionamiento individual en placas de prototipado, de manera que se pueda también entender y adaptar el código a las necesidades del proyecto. Cuando todos los elementos funcionen de esta manera, se procederá al diseño y desarrollo de una placa PCB (placa de circuito impreso, o "printed circuit board" por sus siglas en inglés) con el objetivo de que el circuito sea lo más compacto posible, a la vez que fiable y fácilmente reproducible. A continuación, se diseñará la carcasa impresa en 3D en la que irá alojada la PCB con todos los módulos soldados a ella, así como el prisma y los acoples para integrarla correctamente en cualquier casco, y en la fase final se testeará de primera mano el funcionamiento del prototipo en un entorno controlado (como puede ser un parking), de manera que puedan realizarse las mejoras necesarias para tener un prototipo lo más cercano posible a un dispositivo comercial.

3. ESTADO DEL ARTE

En los últimos años se han desarrollado, con mayor o menor éxito, productos similares al que se está diseñando en este trabajo. Se muestran lo más relevantes a continuación.

3.1. NUVIZ

Es básicamente el dispositivo que se está replicando. Además de ver la velocidad a través de un prisma transparente colocado frente a la visera del casco, también se ve un mapa GPS y se pueden controlar funcionalidades como música desde el smartphone a través de bluetooth y llamadas de teléfono. En la parte delantera cuenta con una cámara para tomar vídeos/fotos, y el dispositivo entero puede controlarse desde un pequeño mando que puede colocarse en cualquier sitio al alcance de la mano.

Salió al mercado en 2017 con un precio de 699$, y pese a su relativo éxito inicial (las críticas generales fueron muy buenas), la compañía desapareció en 2020 sin dar explicaciones. Se cree que la causa principal fue que expiró la licencia de mapas que usaba para su GPS. En la actualidad, tanto la web americana como europea de Nuviz han desaparecido, y quien lo compró ya no tiene soporte por parte de la compañía. Además, al haber expirado las licencias de mapas, lo dispositivos quedan inutilizados por software y ya no ofrecen ninguna funcionalidad.

Figura 2. Nuviz instalado en un casco (www.moto1pro.com)

Figura 1. Vista del Nuviz desde el interior del casco (www.motorpasion.com)

3.2. EYELIGHTS

Es quizás el sistema HUD para motociclistas que más éxito ha tenido, con más de 15000 ventas desde su creación en 2017. El principio de funcionamiento y funcionalidades son similares a las del Nuviz, pero en este caso el sistema va por dentro del casco, y el prisma es macizo, más pequeño y se coloca a escasos centímetros del ojo. Su precio es de 499€. (https://eye-lights.com)

Figura 3. Eye-lights montado en un casco (www.eye-lights.com)

Figura 4. Vista del eye-lights desde el interior del casco (www.eye-lights.com)

3.3. DVISION

Dispositivo muy similar al Eyelights, pero en este caso en vez de tener un pequeño prisma dentro del casco, lo que hay es una pequeña pantalla transparente en la que se proyecta la información. Esta pantalla tiene un eje de manera que puede doblarse hacia arriba, dejando así el espacio de visión totalmente libre en caso de que se desee. Su precio es de 329€. (https://www.dvision-hud.com/en)

Figura 6. Dvision instalado en un casco (www.motor1.com)

Figura 5. Vista del Dvision desde el interior del casco (www.sheisarider.com)

3.4. SHOEI OPTICSON

Es un casco con el sistema HUD integrado, y aunque de momento es únicamente un prototipo, es la primera vez que una marca de renombre en el mundo de los cascos como Shoei plantea un producto similar. Aunque de momento no se tienen muchos detalles, el casco implementaría las funcionalidades base de mostrar la velocidad y GPS.

Figura 8.Prototipo del Shoei Opticson (www.soymotero.net)

Figura 7. Vista desde el interior del Shoei Opticson (www.soymotero.net)

4. DISEÑO FUNCIONAL

El diseño o arquitectura básica que se ha seguido en el proyecto se muestra la *Figura* 9. Como se puede ver, es una configuración centralizada alrededor del microcontrolador (μC) ESP32, y donde se pueden encontrar todos los elementos fundamentes de un canal de instrumentación o medida clásicos: el sistema sensor (GPS), el sistema de alimentación (batería y cargador) y el sistema de presentación (pantalla OLED). Los detalles de cada módulo empleado serán explicados en el apartado *Componentes periféricos.*

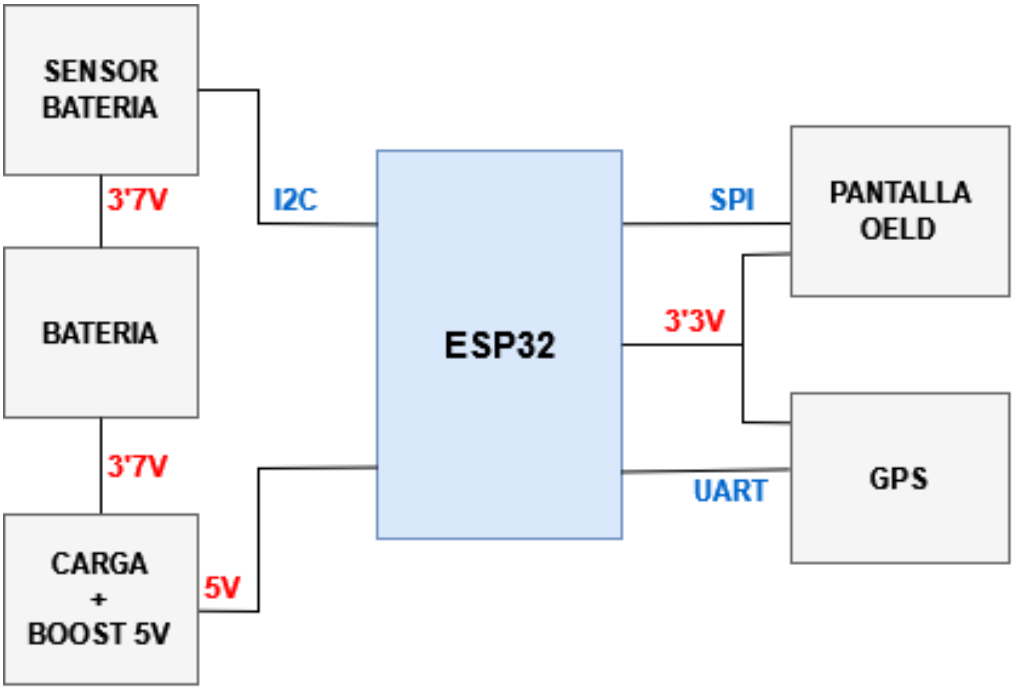


Figura 9. Diseño base

5. DISEÑO DEL HARDWARE

5.1. PLACAS DE HW LIBRE

Para el desarrollo de este prototipo se va a utilizar como microcontrolador una placa de hardware libre, es decir, que su esquema interno es de acceso público, por lo que es más simple a la hora de utilizarlo. Suelen constar de un microprocesador, pines externos conectados a este para facilitar el uso, y a veces sensores incorporados. Los requisitos mínimos que se buscará en la placa serán:

- Tamaño reducido.
- Protocolos de comunicación I2C, SPI, y UART. El USB facilita la programación, pero no es indispensable.
- Alimentación a través de un pin "Vin".
- Salida de tensión de 3'3V.

En cuanto a especificaciones como la frecuencia del microprocesador, memoria y demás, apenas se van a tener en cuenta ya que es el dispositivo no requiere de altas prestaciones y la mayoría de las placas que hay en el mercado actualmente lo cubren sobradamente. Dada esta circunstancia, a continuación, se muestran los μC's planteados para usarse en el proyecto y la justificación de la elección final llevada a cabo.

5.1.1. Arduino UNO

Es el más básico de la familia Arduino y el que más se suele usarse a la hora de hacer prototipos, ya que cuenta con pines hembra integrados, lo que facilita mucho su utilización. Debido a su tamaño (es el Arduino más grande después del Arduino Mega), no va a ser utilizado en el prototipo final, pero sí que se ha utilizado a la hora de realizar tests rápidos sobre el código, como por ejemplo a la hora de refinar la visualización en la pantalla OLED, ya que este Arduino cumple los requerimientos necesarios para ello.

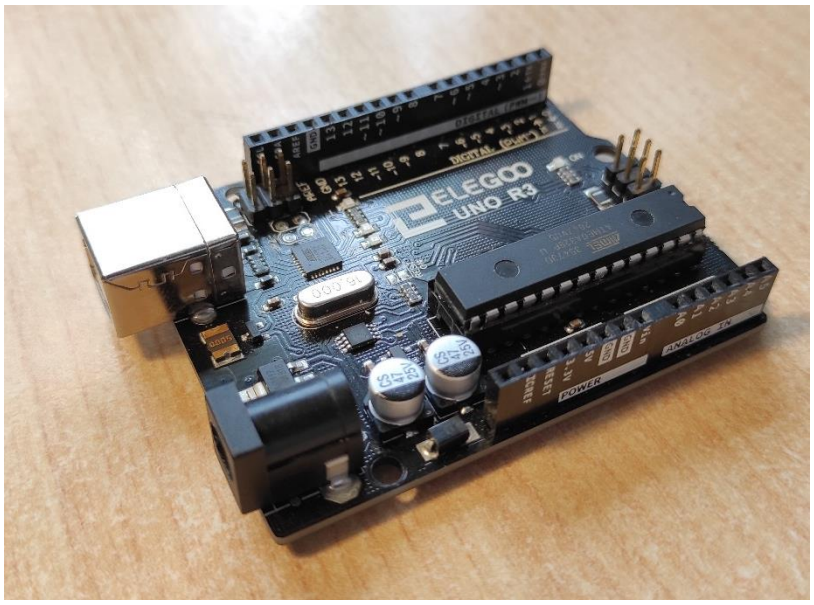


Figura 10. Arduino UNO de la empresa ELEGOO. Los pines hembra lo hacen muy adecuado para testear rápidamente el código.

5.1.2. Arduino Nano

Es, después del Arduino Pro Mini, el más pequeño, lo cual interesa mucho para este proyecto. Cumple con todas las especificaciones expuestas previamente, por lo que en conjunto es una muy buena opción. De hecho, los primeros pasos para llegar al prototipo final se dieron con el Arduino Nano. Sin embargo, al no contar con pines hembra, no se ha utilizado tanto a la hora de testear como el UNO.

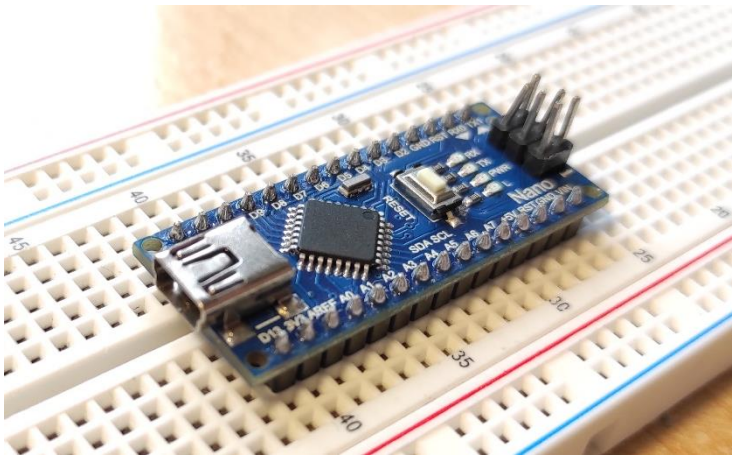


Figura 11. Arduino Nano. El no contar con pines hembra hace necesario el uso de una protoboard si quiere usarse sin soldar cables.

5.1.3. Arduino Pro Mini

Es el más pequeño de la familia Arduino. Al igual que el Nano, cumple con todos los requerimientos, pero en este caso no cuenta con USB para conectarlo al ordenador y programarlo directamente, sino que se utiliza un conector FTDI externo para conectarlo al ordenador. Este es la causa principal por la que no se llegó a utilizar, ya que el proceso de programación y testeo puede ser engorroso y a la hora de fabricar prototipos es deseable una mayor agilidad. Sin embargo, para un posible producto final podría ser interesante.

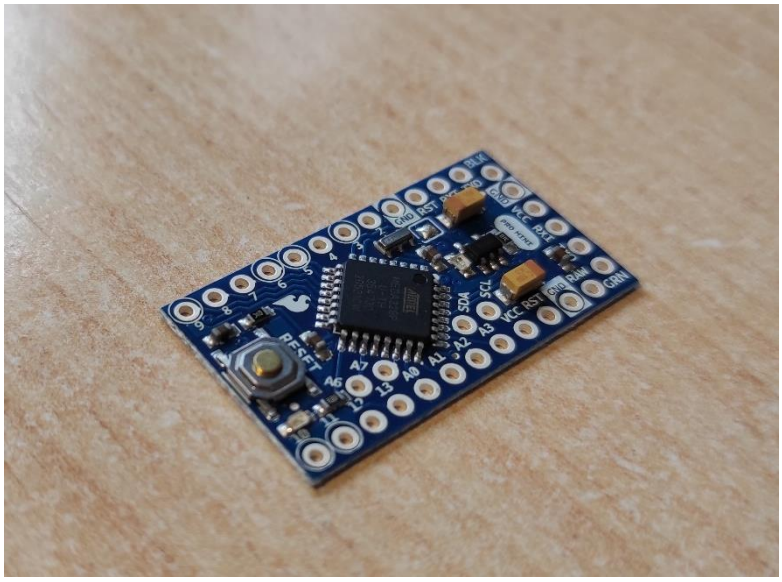


Figura 12. Arduino Pro Mini, el más pequeño de todos.

5.1.4. ESP32

Creado y desarrollado por Espressif Systems, es similar a un Arduino Nano, pero algo más grande (entre el Nano y el Uno, aunque más cercano al primero), más potente y con conexión Wifi y Bluetooth integradas, además de sensores capacitivos en varias de sus patillas, un sensor Hall y un sensor de temperatura interna. Cuenta con todas las conexiones necesarias para el proyecto, incluido el puerto USB.

Teniendo en cuenta no solo los requerimientos básicos sino también posibles futuras implementaciones al prototipo (ver apartado *Líneas futuras)*, se ha seleccionado el ESP32 como microcontrolador a utilizar.

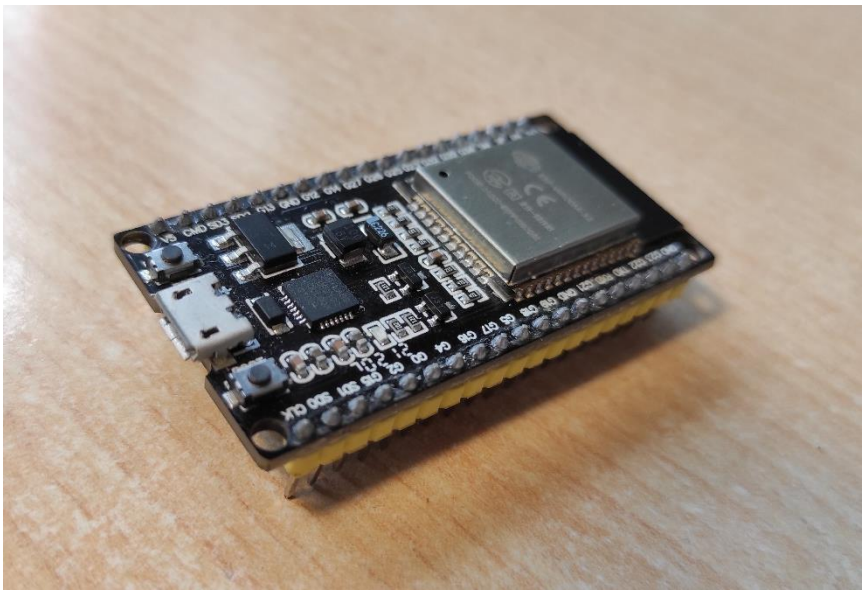


Figura 13. El ESP32 va a ser el microcontrolador en el que se va a basar el prototipo

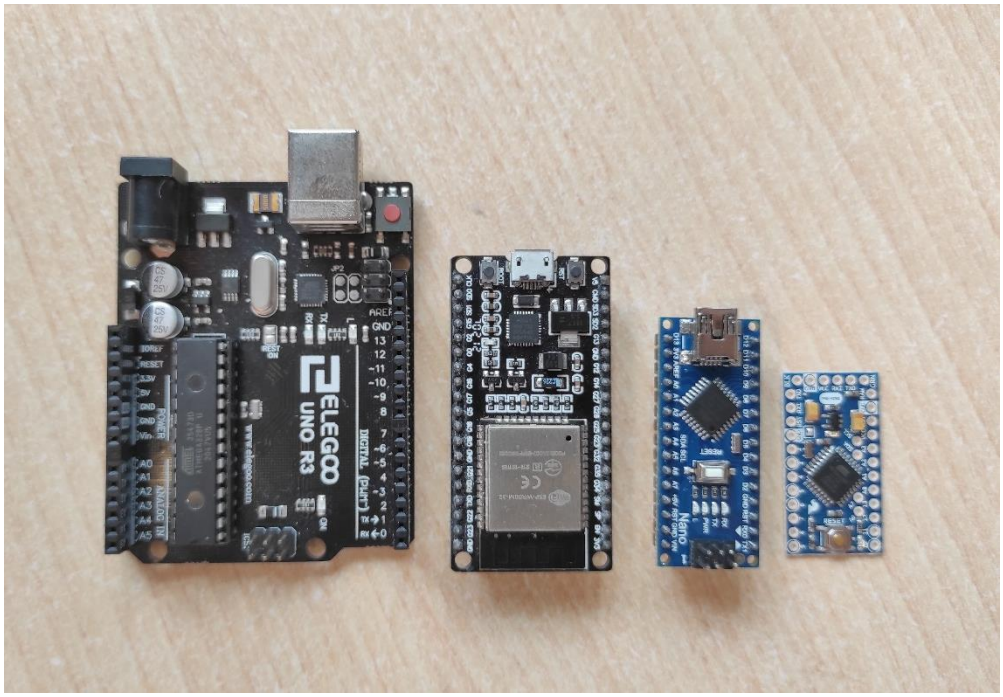


Figura 14. Comparativa de dimensiones. De izquierda a derecha: Arduino UNO, ESP32, Arduino Nano y Arduino Pro Mini.

5.2. COMPONENTES PERIFÉRICOS

5.2.1. Batería

Teniendo en cuenta que el dispositivo será portátil, es necesario que cuente con una batería para que pueda funcionar de forma autónoma, y, además, esta ha de ser lo más pequeña y ligera posible, además de contar con la suficiente capacidad de carga. Por todo ello, se ha utilizado una batería Lipo (polímero de litio), ya que es la que mayor densidad energética ofrece. En concreto, se ha seleccionado el modelo SR674361P, con una capacidad de 2.000 mAh, una tensión de 3'7 V y una energía de 7'4 Wh. Además, cuenta con un circuito integrado de protección contra sobrecarga, sobredescarga y cortocircuito.

Figura 15. Batería Lipo

Para ver los ensayos de carga y descarga de esta batería en el prototipo final, ir al apartado *Ensayos y resultados*.

5.2.2. Cargador / Boost a 5V

Puesto que para cargar una batearía Lipo se necesita utilizar el método de corriente constante tensión constante (CC/CV, *Figura 17*), tendrá que haber un circuito que regule estos parámetros. Para ello se utilizará el módulo PowerBoost 1000 Charger de Adafruit que, además, eleva la tensión de la Lipo a los 5V necesarios para alimentar el ESP32.

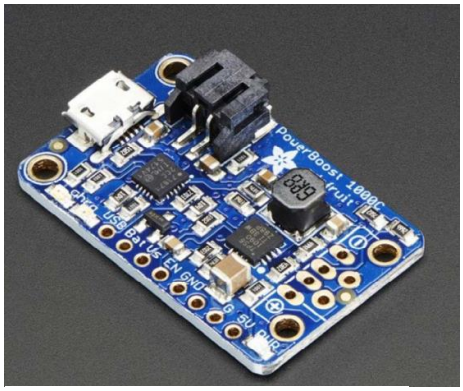


Figura 16. Cargador / boost 5V

Como se acaba de comentar, las necesidades específicas de las baterías Lipo (no deben sobrepasar los 4'2V o se corre riesgo de explosión, por ejemplo) hacen necesario el método de carga CC/CV. En este método, el controlador de carga comienza suministrando un valor de corriente constante, de manera que la carga y la tensión de la batería van aumentando. Cuando esta tensión llega a 4'2V, se cambia a la fase de tensión constante, ya que nunca debe de subir de 4'2V. La carga seguirá subiendo a esta tensión poco a poco hasta llegar al 100%.

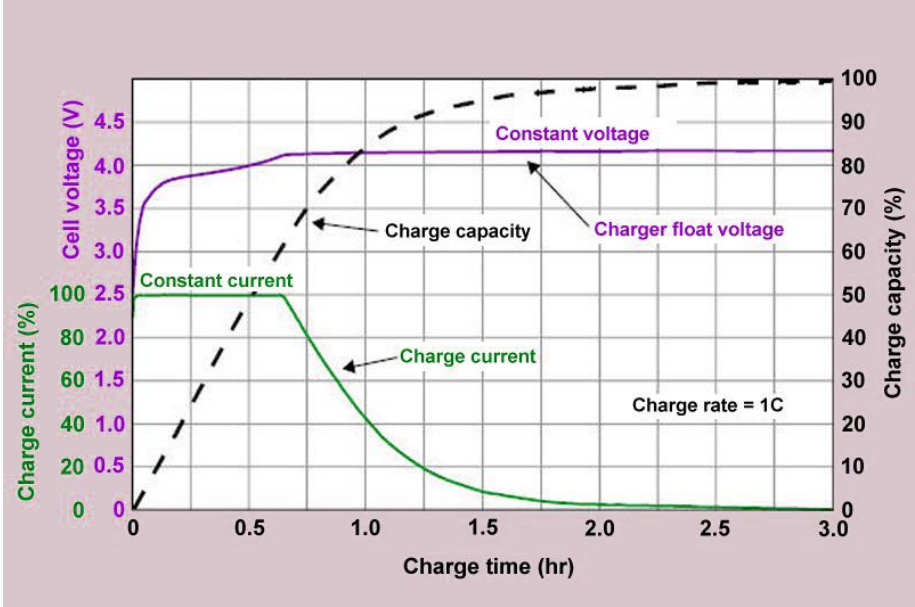


Figura 17. Gráfica de carga baterías lipo

5.2.3. GPS

La posición y velocidad se obtendrán, como se ha comentado previamente, por señal de GPS. Esta se obtendrá gracias al módulo GY-GPS6MV2, que transmitirá los datos al ESP32 mediante comunicación UART (Tx, Rx), alimentado a 3'3V. En su datasheet se indica que la primera conexión se da en menos de 27 segundos, y a partir de entonces si se apaga y se enciende, se conectará en menos de 3 segundos. Para ver las pruebas empíricas, ir a apartado *Tiempo de conexión a GPS*.

Figura 18. Módulo GPS

5.2.4. OLED

El elemento que proyecta la información al prisma es una pantalla OLED de 128x32 pixeles, alimentada a 3'3V. Como se ha comentado en apartados anteriores, el protocolo de comunicación que utilizará será SPI. Su reducido tamaño la hace ideal para el proyecto.

Figura 19. Pantalla OLED

5.2.5. Sensor carga batería

Se desea mostrar por la pantalla en todo momento el nivel de carga de la batería Lipo. Sin embargo, la curva de descarga de estas es altamente no lineal, por lo que la manera más precisa de conocer su estado es mediante un sensor de carga ("fuel gauge"). Se ha utilizado el ofrecido por la marca DFROBOT, que cuenta con un algoritmo patentado que estima el porcentaje de carga de manera precisa y lo comunica al microcontrolador mediante protocolo I2C.

Figura 20. Sensor de carga de batería

5.3. DISEÑO

Se pueden distinguir 2 partes claramente diferenciadas en el diseño del prototipo: la parte eléctrica (esquema eléctrico y PCB), y la parte mecánica (prisma y carcasa exterior). A continuación, se explica en detalle cada una de ellas.

5.3.1. Esquemático

Teniendo en cuenta los protocolos de comunicación empleados (ver apartado *Protocolos de comunicación utilizados*), el esquema eléctrico (realizado con el programa EasyEDA, explicado en el apartado *PCB*) queda como puede observarse en la *Figura 21*. El ESP32 se alimenta a 5V, por lo que la batería Lipo (que proporciona 4'2V) deberá conectarse primero al Boost a 5V y de ahí al µC. Éste cuenta a su vez con una salida de 3'3V mediante la cual se alimentarán el resto de los componentes (GPS, pantalla OLED, sensor de carga de batería).

Puede observarse que alrededor de la mitad de los pines del ESP32 quedan en desuso. Esto se debe a que, como se ha comentado anteriormente, es un µC que ofrece muchos más recursos de los que se utilizarán en este proyecto, por lo que en realidad su aprovechamiento no será óptimo. El objetivo de ello es dejar puertas abiertas a futuras mejoras, como incorporar más sensores o más salidas.

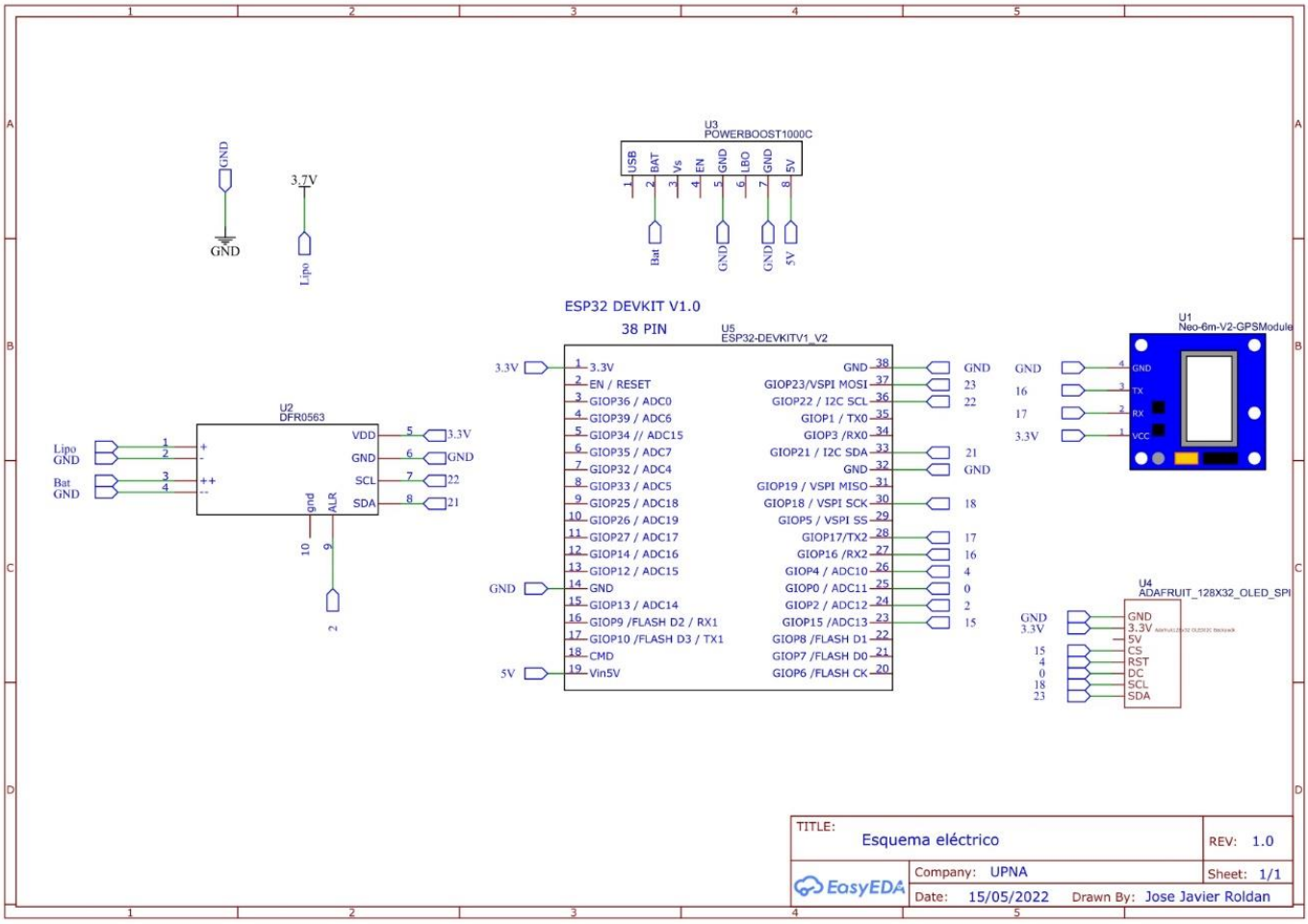


Figura 21. Esquema eléctrico

5.3.2. PCB

La necesidad de diseñar y fabricar una PCB viene de la búsqueda de minimizar al máximo el tamaño del dispositivo, así como tratar de hacer que sea lo más robusto posible.

El programa que se ha elegido para diseñar la PCB ha sido EasyEDA. Una de las características más interesantes que posee es que, aunque tiene aplicación de escritorio, los datos se almacenan en la nube, por lo que se puede

acceder y trabajar en los diseños desde ordenadores diferentes con una misma cuenta.

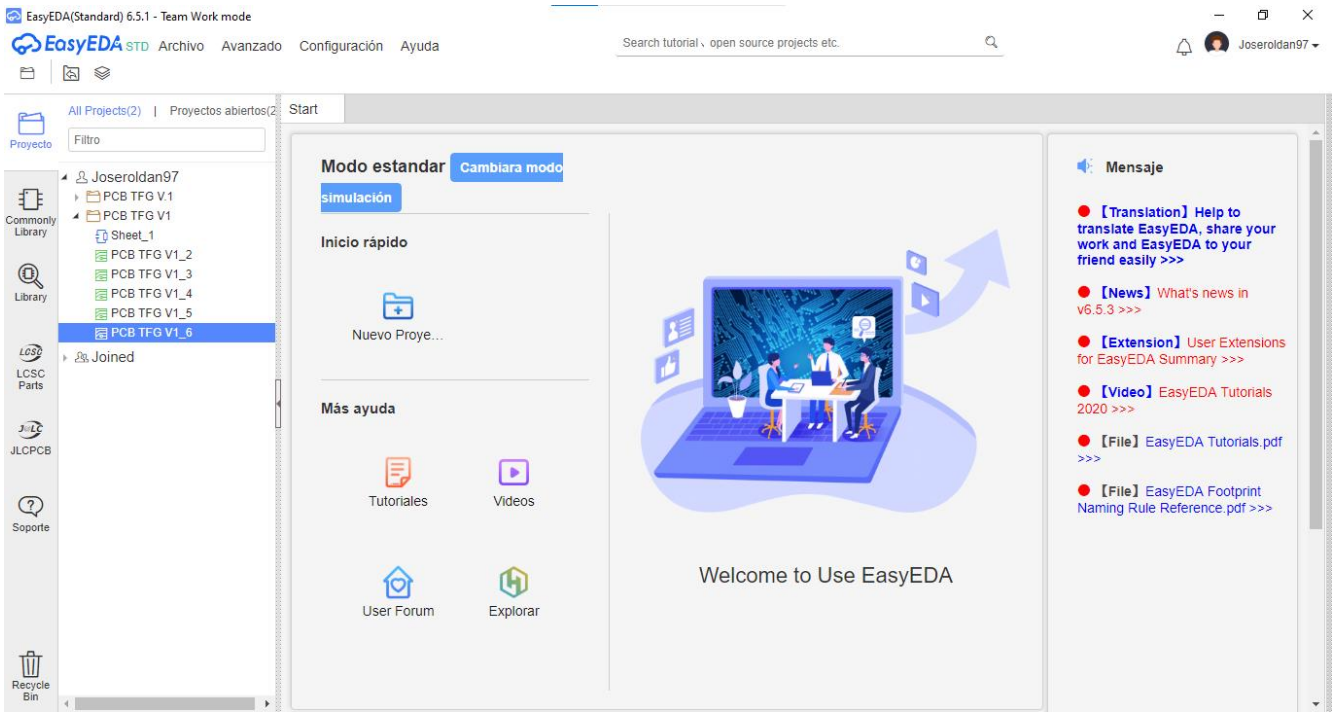


Figura 22. Página de inicio de EasyEDA

El hecho de que sea un programa tan enfocado en la nube ofrece numerosas ventajas además de la que ya se ha comentado. Por ejemplo, como puede verse en la parte central izquierda de la *14Figura 22* (apartado "Joined") se puede trabajar en proyectos de manera conjunta, lo cual puede ser muy interesante si se está desarrollando un diseño entre varias personas. Incluso en su librería ofrece un apartado en el cual la comunidad puede subir sus diseños de módulos específicos que quizás no son tan comunes y cuyo diagrama no es ofrecido por el fabricante. Como es obvio, esto conlleva el peligro de que, como cualquier usuario puede subir lo que quiera, se suban circuitos erróneos. Sin embargo, todos los diseños son fácilmente editables por lo que únicamente habrá que estar pendiente de corregir los potenciales fallos. En el caso de este proyecto, todos los módulos estaban disponibles en el apartado "comunidad" de la librería. Sin embargo, el del módulo GPS tenía dos pines cambiados respecto al modelo usado, por lo que con se corrigió y se pudo terminar el diseño adecuadamente.

En cuanto al diseño, lo principal que se buscó fue, por un lado, que la conexión micro USB del módulo de carga de la batería quedase en un lateral de la placa, de manera que fuese fácilmente accesible al usuario final, y por otro, que las pistas fueran lo más cortas posibles para evitar al máximo posibles ruidos e interferencias. Cabe destacar que todos los componentes estarán conectados a la PCB encima de esta mediante tecnología THT (Through-Hole-

Technology o tecnología de agujeros pasantes), menos la pantalla OLED, ya que se colocará en otro extremo de la carcasa para que quede lo más visible al motorista. Se conectará a la PCB mediante cables, lo que le proporcionará a su vez flexibilidad para realizar posibles ajustes.

Se decidió un ancho de pista de 0'7mm tanto para las de datos como para las de potencia, ya que al ser un circuito DC y de baja tensión y corriente no se necesitaban requerimientos mucho más específicos. Otro de los motivos para elegir este ancho es que la fabricación de la PCB se realizó con la fresadora cónica de la Upna, con la cual se recomienda que las pistas sean al menos de 0'5mm. Esta fresadora parte de una placa de dieléctrico cubierta de cobre a ambas caras, y al ir retirando la capa de cobre con la punta va creando el diseño en la cara. En vez de retirar todo el cobre menos el de las pistas, lo que se ha hecho ha sido retirar únicamente el estrictamente contiguo a estas, aislándolas. Después, todo el cobre que no es pista se conecta con las masas, de manera que todas las posibles masas del circuito están cortocircuitadas. Además, este diseño (masa a lado y lado de cada pista) ayuda a evitar interferencias y ruidos no deseados.

Como en cualquier PCB, se han evitado al máximo los giros de 90 grados de las pistas para afectar lo menos posible a la señal. Aunque se ha intentado diseñar de modo que todas las pistas y componentes quedes en una única cara (facilitando así la soldadura), han sido necesarias 3 agujeros pasantes o vías para pasar una pista de alimentación por la cara inferior. Con todo, el diseño final ha sido el siguiente:

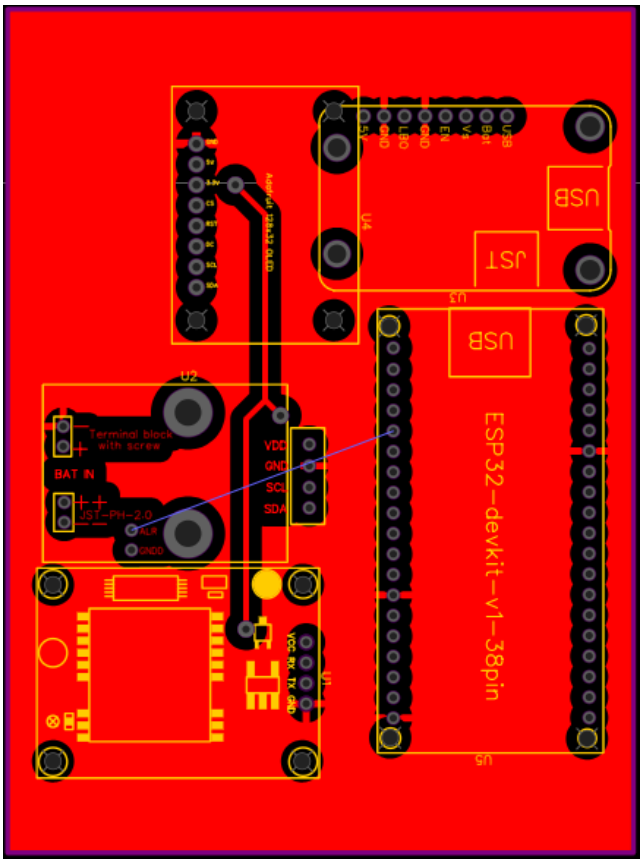


Figura 23. Vista superior de la PCB

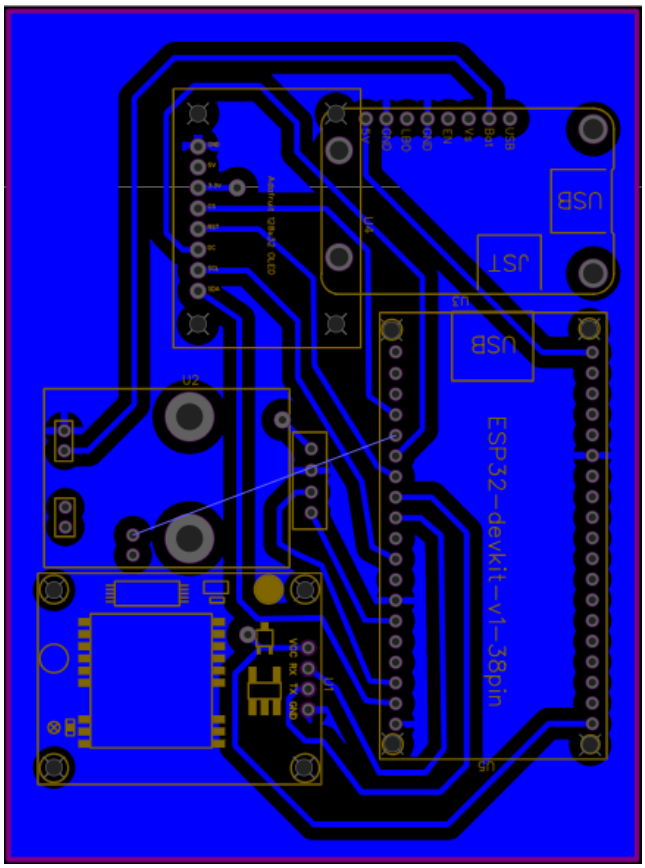


Figura 24. Vista inferior de la PCB

Algo a tener en cuenta es que en ningún momento la capa superior e inferior están conectadas, por lo que es necesario hacerlo de manera manual mediante una vía (soldando un cable a ambos lados a través de un agujero pasante, por ejemplo). Esto es importante tenerlo en cuenta también a la hora de diseñar, puesto que los elementos se colocan en un lado y sus pines se sueldan en el otro, siendo este último en el que están las pistas. Esto no se tuvo en cuenta en el diseño de la primera PCB, y como no es simétrica, hubo que descartarla y cambiar todas las pistas de cara para poder soldar los elementos correctamente.

Una vez diseñada y fabricada la PCB, lo primero que se hizo fue comprobar la continuidad de las conexiones entre pads mediante un multímetro. Fue aquí donde se comprobó que había algunos pines de masa que no tenían la conexión adecuada. Para solucionarlo, se creó un agujero pasante (1) para conectar la cara superior e inferior y que la superficie grande de ambas sea masa. A continuación, se hicieron dos pasantes más (2,3) en lugares estratégicos y, soldando, se consiguió solucionar. En la *Figura 26* y *Figura 27* puede verse la PCB con flechas indicando estos puntos a continuación.

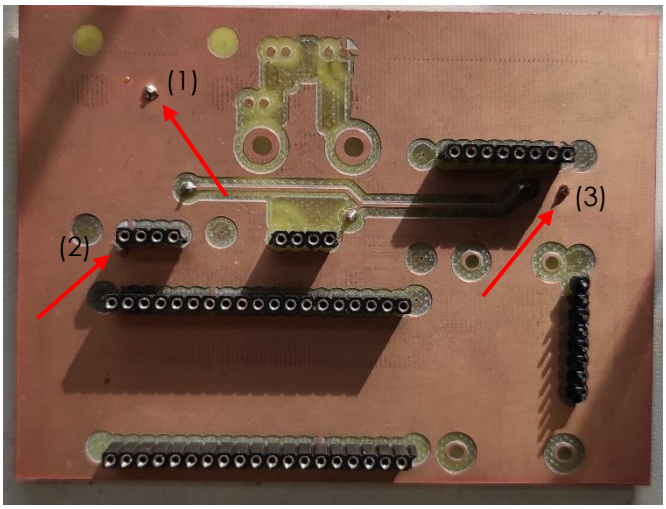


Figura 26. Vista superior PCB. Flechas rojas indican agujeros pasantes necesarios para la correcta conexión de todas las masas

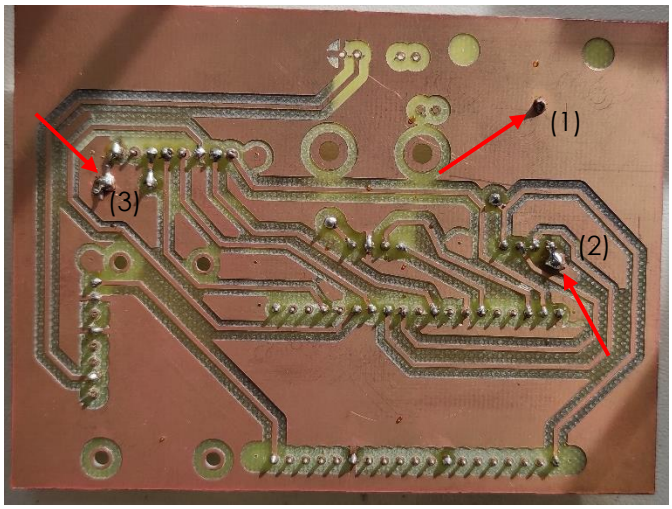


Figura 25.Vista inferior PCB

Realizar unas buenas soldaduras de los pines a la placa es fundamental para para la correcta conexión y transmisión de información a través de esta. Se intentará que, al acabar de soldar con el estaño, quede un pequeño cono que garantice la integridad de la soldadura, como puede verse en el dibujo a continuación.

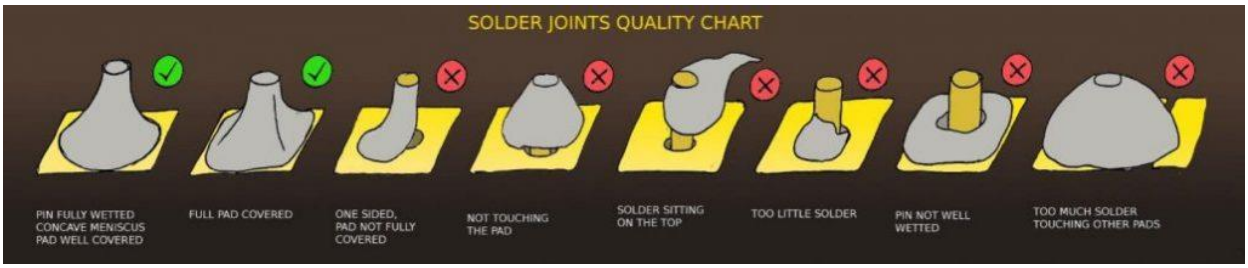


Figura 27. Comparación de calidad de soldaduras

Teniendo este dibujo en mente, se soldaron todos los pines necesarios, así como algunos extras para mejorar la rigidez de la unión entre pines y PCB.

Figura 28. Puesto de soldadura con la PCB lista para trabajar en ella

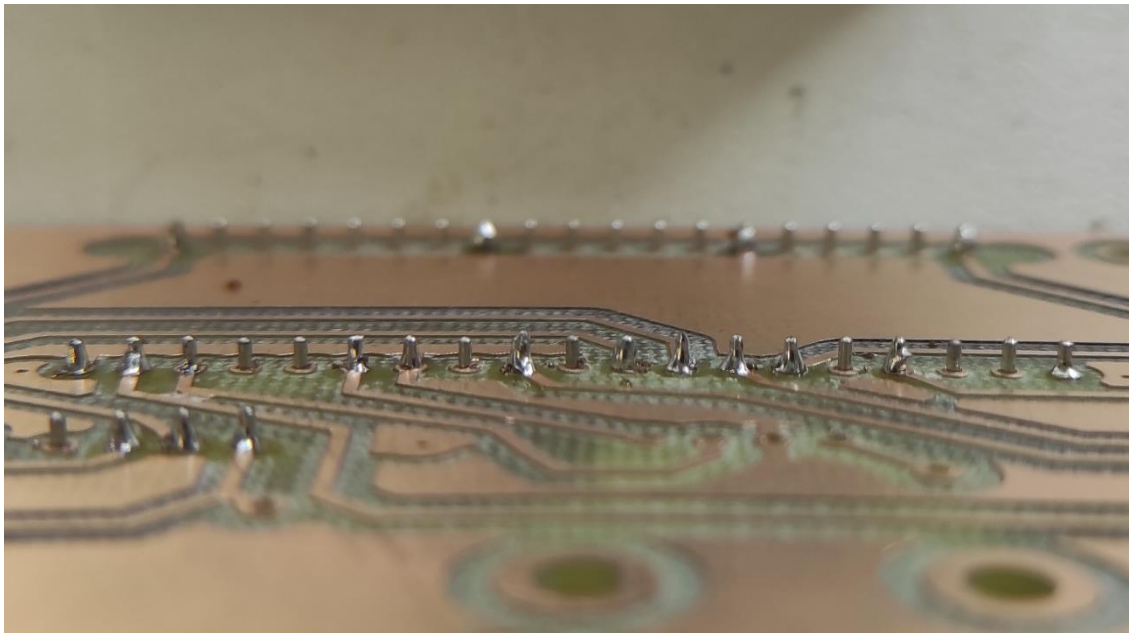


Figura 29. Primer plano de algunas de las soldaduras realizadas. En la mayoría de ellas puede intuirse el cono de estaño que se busca.

Tras comprobar con el multímetro la correcta conexión entre todos los pines, se colocaron todos los elementos, quedando de la siguiente manera.

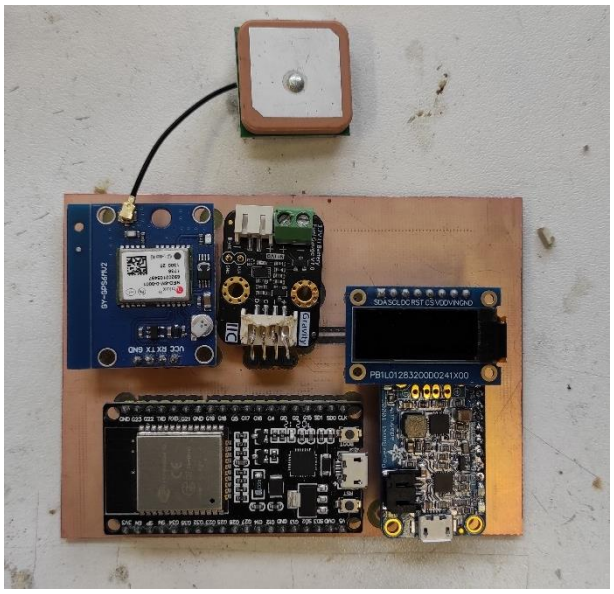


Figura 30. Vista superior de la PCB con los módulos (menos la batería) conectados a los pines hembra soldados

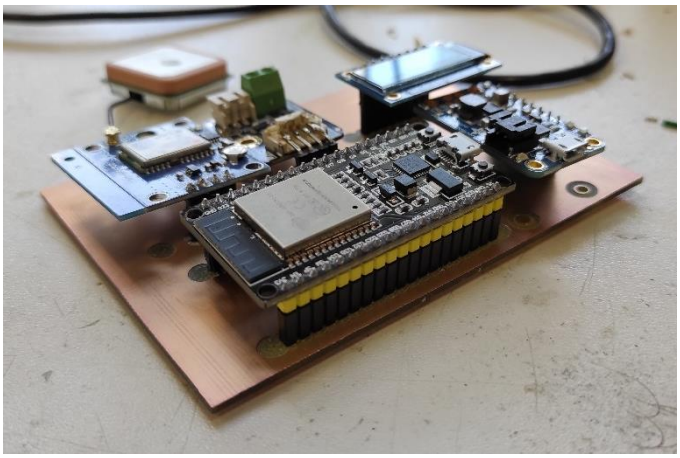


Figura 31. Vista lateral de la PCB con los módulos conectados

5.3.3. Prisma

Otro elemento de hardware fundamental es el prisma a través del cual el usuario ve la información proyectada por la pantalla OLED. Una vista lateral del esquema básico de este puede verse en la *Figura 32*:

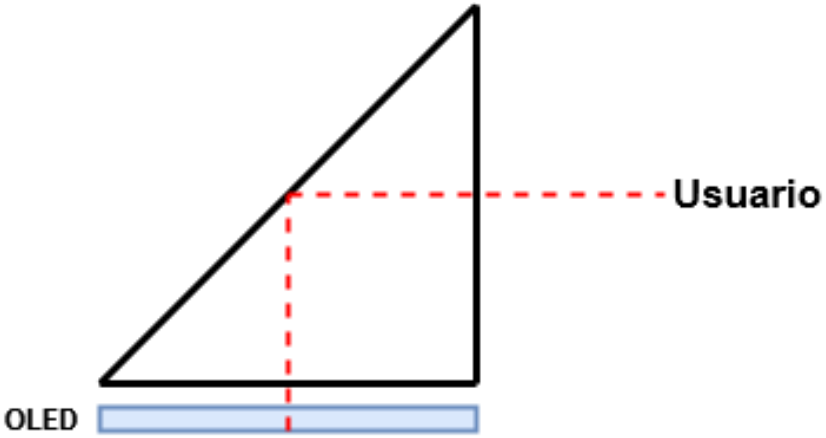


Figura 32. Diseño básico del prisma.

Dado que el objetivo es que el usuario vea la información en una pantalla transparente, lo que se hará es proyectar la pantalla (horizontal) en un plástico reflectante a 45 grados, de manera que se refleje y el usuario lo vea perfectamente. Debido a este giro, la información se verá invertida, por lo que habrá que indicar en el código que la información se invierta en la pantalla OLED primero.

En cuanto a los materiales utilizados, primeramente, se planteó la opción de hacer un prisma macizo de metacrilato o algún material similar, pero se descartó debido a la complejidad de su fabricación y al peligro potencial que suponía, ya que al ser macizo su peso y rigidez eran mayores, y en el caso de que se despegase del prototipo e impactase contra la cara del usuario, había riesgo de lesiones graves. La siguiente opción (la que se utilizó finalmente) fue crear el prisma recortando láminas de plástico y pegándolas. De esta manera quedaría hueco, evitando los peligros potenciales nombrados anteriormente. Tanto para los laterales como para el rectángulo vertical y horizontal se ha utilizado plástico de 1mm de espesor, que es más que suficiente para aportar rigidez. La cara reflectante a 45º se hizo primeramente del mismo material. Sin embargo, no reflejaba lo suficiente, por lo que se cambió de material y se utilizó una lámina de plástico reflectante utilizado en los coches para crear sistemas HUD reflejando el móvil en este plástico pegado al parabrisas (ver *Figura 33*).

Figura 33. Plástico reflectante para crear un sistema HUD en cualquier coche

Una vez se tuvieron todos los componentes recortados, se utilizó pegamento extrafuerte para unir todos los elementos. Sin embargo, se produjo una reacción con el plástico que hizo que quedase excesivamente opaco, (*Figura 34*). Aunque quedó inutilizable, sirvió para comprobar que el tamaño del prisma era el adecuado.

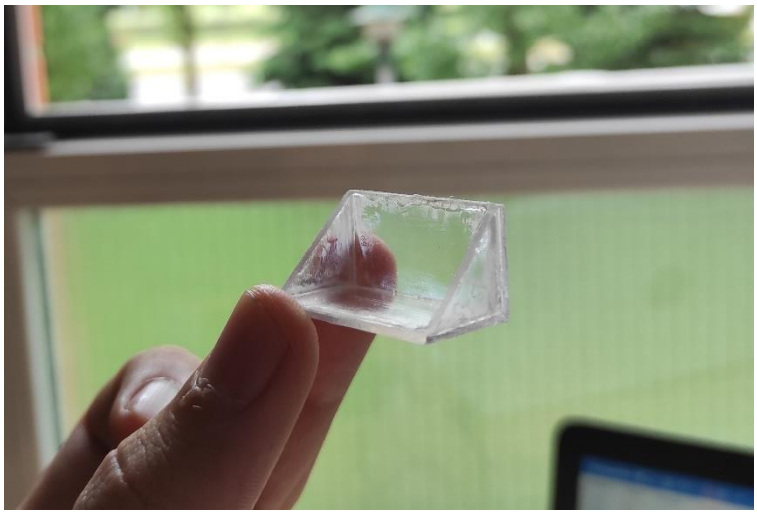


Figura 34. Prisma con el plástico alterado por el pegamento.

La solución para unir el plástico fue utilizar pegamento caliente. Aunque los bordes no quedaron tan lisos como con el pegamento extrafuerte, el plástico quedó limpio y translucido. A continuación, se fijó con más pegamento caliente a la pantalla OLED, quedando de la siguiente manera:

Figura 35. Prisma final sobre la pantalla OLED.

5.3.4. Carcasa exterior

En cuando a hardware, lo último que se fabricó fue la carcasa exterior, puesto que se necesitaban el resto de los componentes terminados para poder diseñarla y que todo encajase lo mejor posible. Se decidió imprimir en 3D, ya que permite personalizar al máximo el diseño y se consigue una muy buena robustez.

El programa utilizado para diseñar la carcasa ha sido SolidWorks, ya que es muy usado en el mundo de la ingeniería y ofrece herramientas más que de sobra para el objetivo que se busca. En cuanto al diseño, se buscó ahorrar el máximo espacio posible. Para ello se colocó la batería justo debajo de la PCB, para lo cual en la carcasa se hizo un pequeño hundimiento en el cual entraba la batería perfectamente. El resto de la carcasa se diseñó alrededor de la PCB teniendo en cuenta sus medidas. Se añadió un agujero para colocar un interruptor, y por error no se colocó el agujero para el cargador, que se hizo a posteriori. Ambos agujeros se hicieron en la cara inferior (una vez el dispositivo ya está montado en el casco), de manera que por gravedad es menos probable que se ensucien y estropeen. También se hicieron dos acoples para enganches de Gopro, de manera que mediante uno de ellos se podía unir el dispositivo al casco, y mediante el otro se podía unir un brazo articulado para colocar la pantalla en la posición adecuada. También se diseñó una pequeña base para poder enganchar la pantalla al brazo articulado. Los planos pueden verse en el Anexo *Planos*.

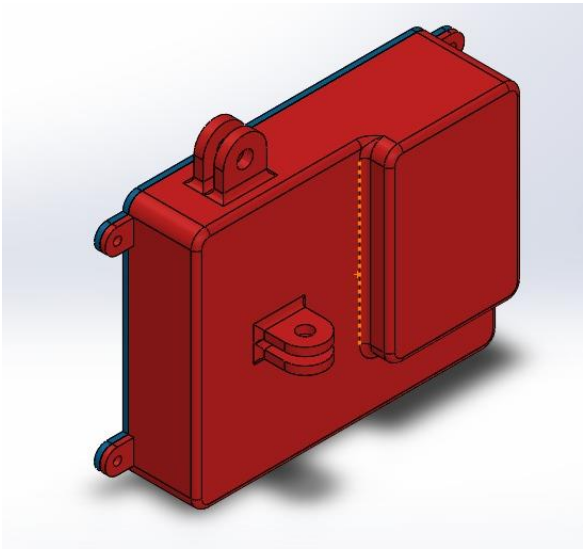


Figura 37. Vista frontal del diseño de la carcasa.

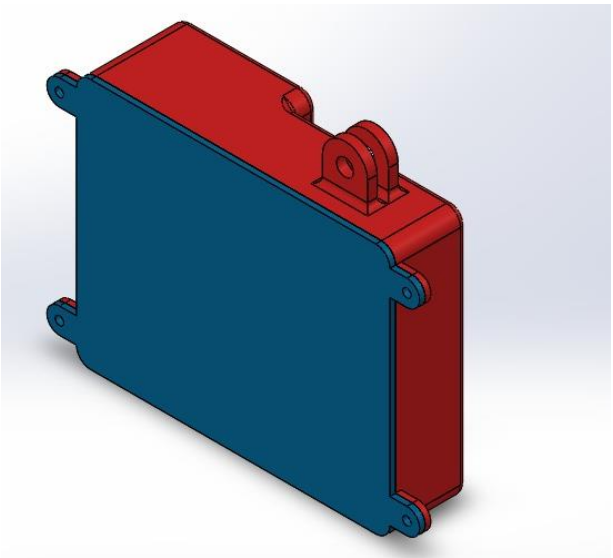


Figura 36. Vista trasera del diseño de la carcasa con la tapa.

Una vez terminado el diseño en SolidWorks, es necesario "traducir" ese archivo al lenguaje de la impresora 3D, indicando los parámetros de impresión, materiales, velocidad… Para ello se utilizó el programa Cura 3D. El material

utilizado fue plástico ABS, ya que es el más común en este tipo de impresoras, con un espesor de 2mm para aportar rigidez.

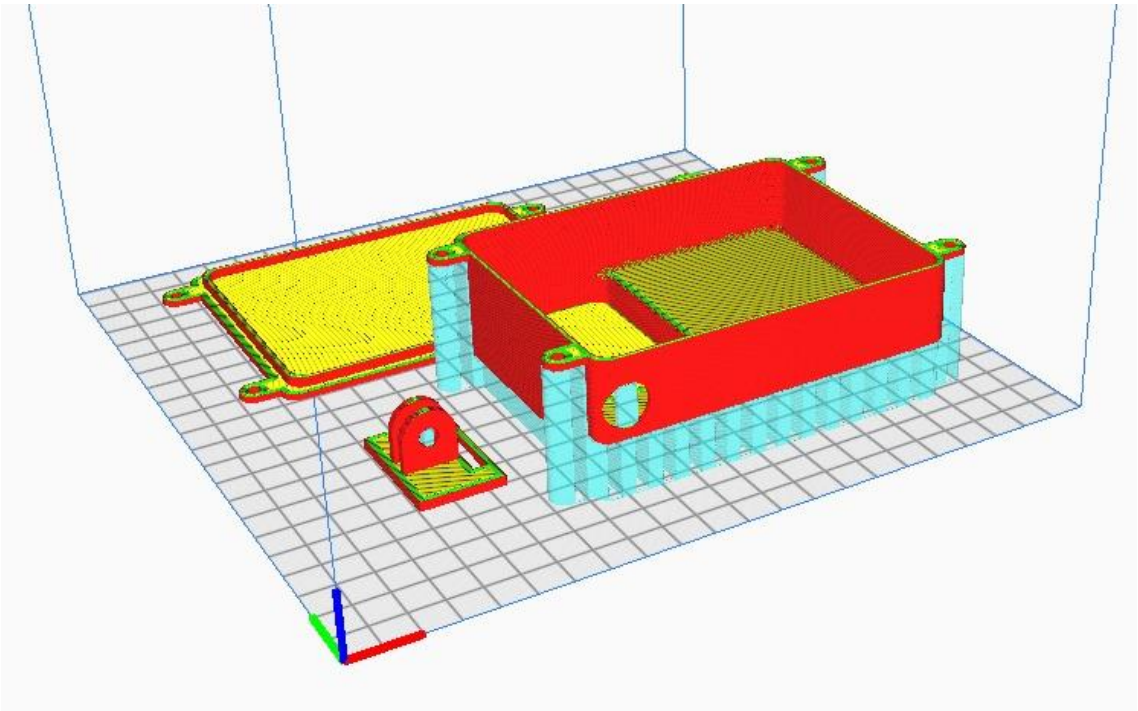


Figura 38. Vista desde el programa Cura 3D de cómo va a ser la impresión. En azul claro están los rellenos que tiene que hacer la impresora para poder imprimir bien.

Tras alrededor de 10 horas y media de impresión, el resultado fue el siguiente.

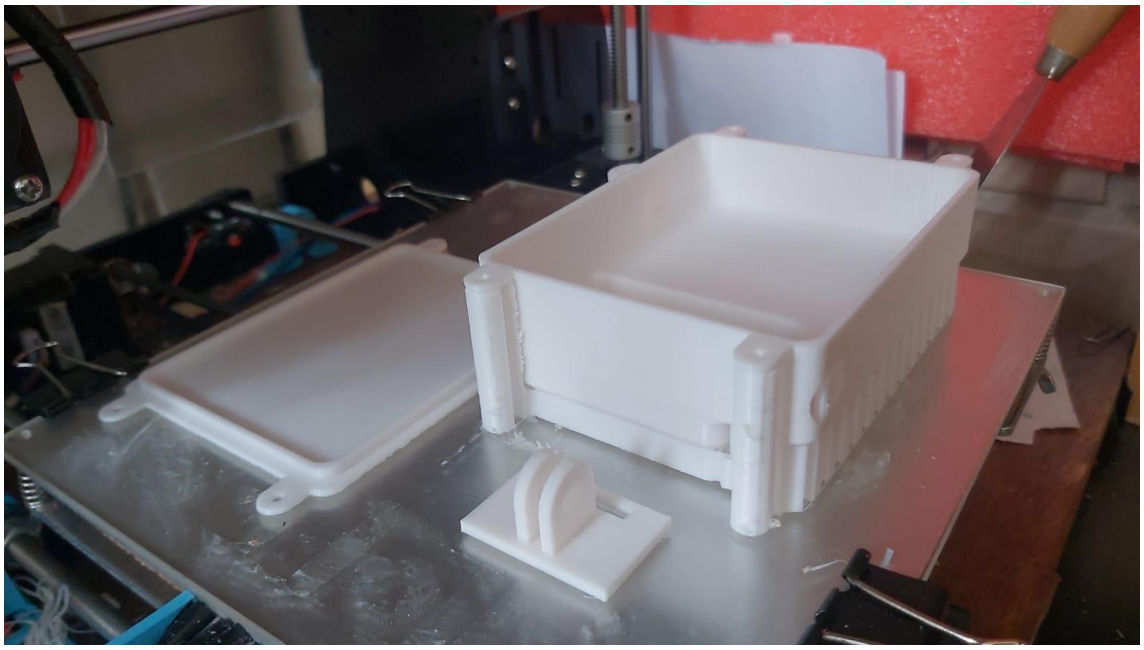


Figura 41. Elementos recién impresos.

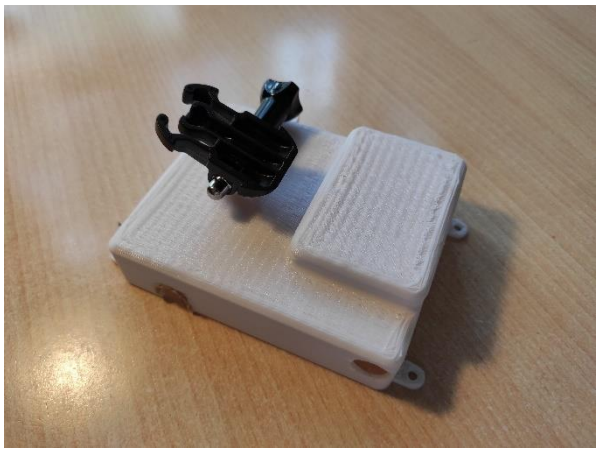


Figura 40. Exterior de la carcasa 3D

Figura 39. Interior de la carcasa 3D

Tras incorporar todos los elementos y soldar los cables de la pantalla, el resultado fue el siguiente.

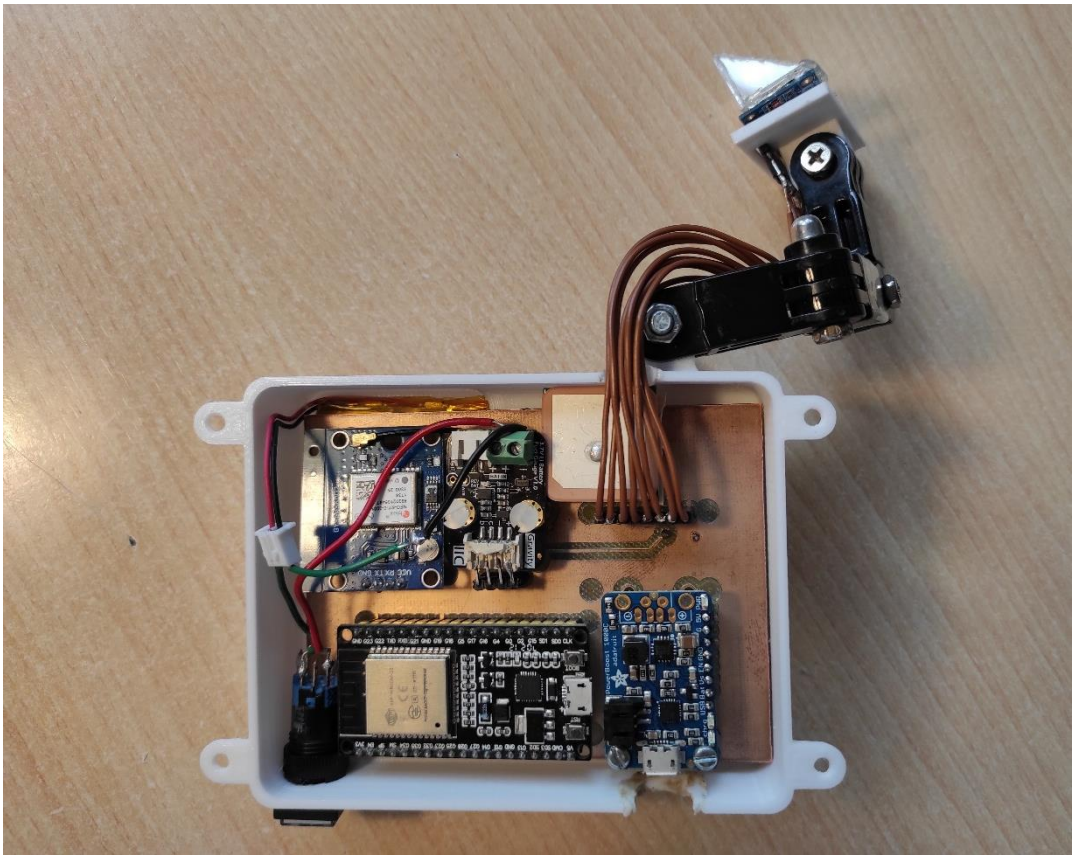


Figura 42. Interior de la carcasa con todo montado.

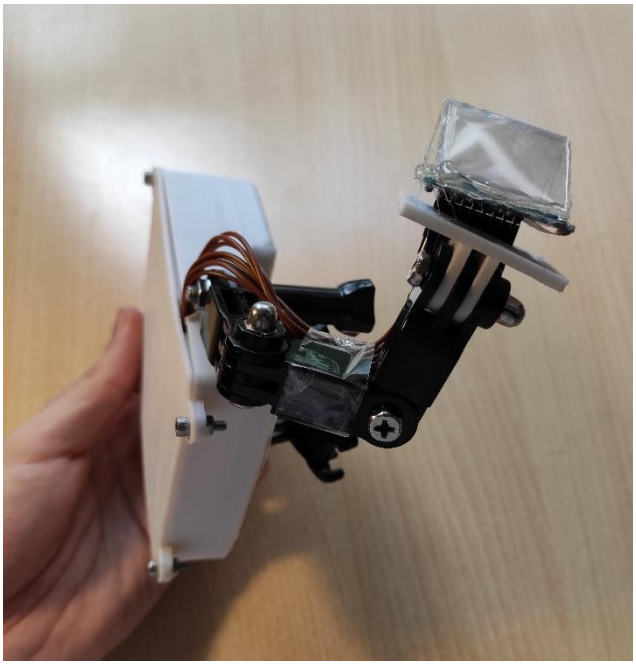


Figura 44. Detalle del brazo de la carcasa.

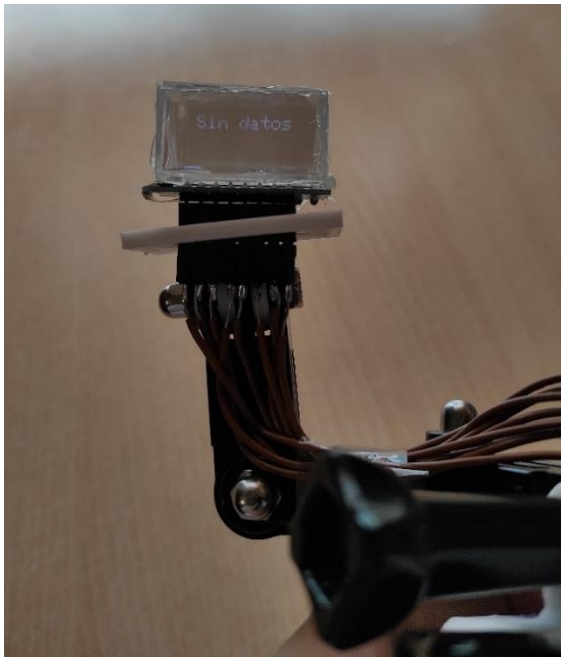


Figura 43. Detalle del prisma en el brazo.

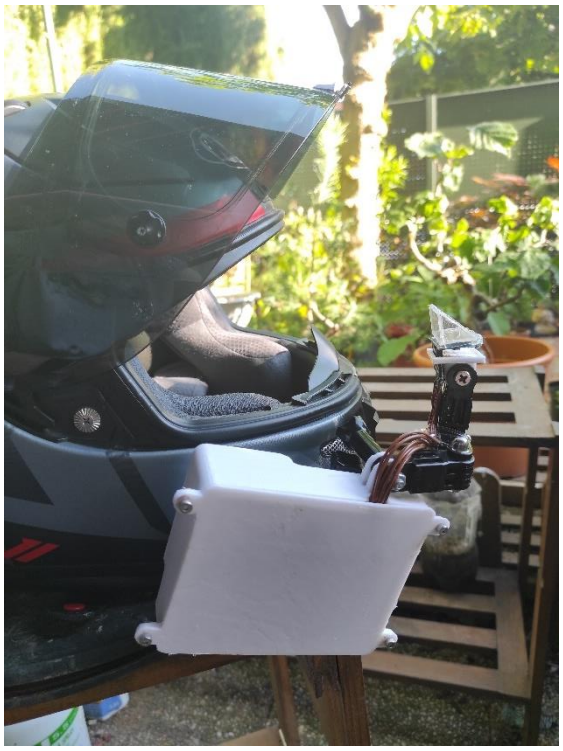


Figura 46. Vista lateral del prototipo en el casco

Figura 45. Prototipo instalado en el casco.

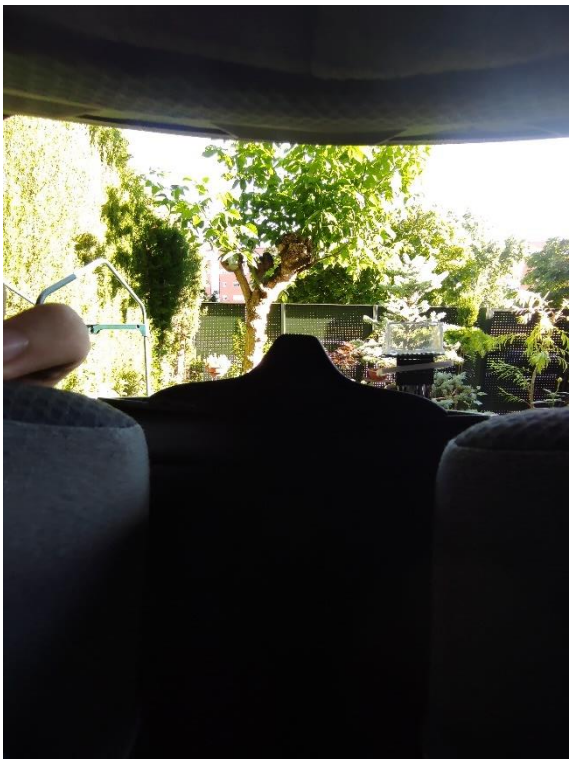


Figura 48. Vista desde el interior del casco en día soleado

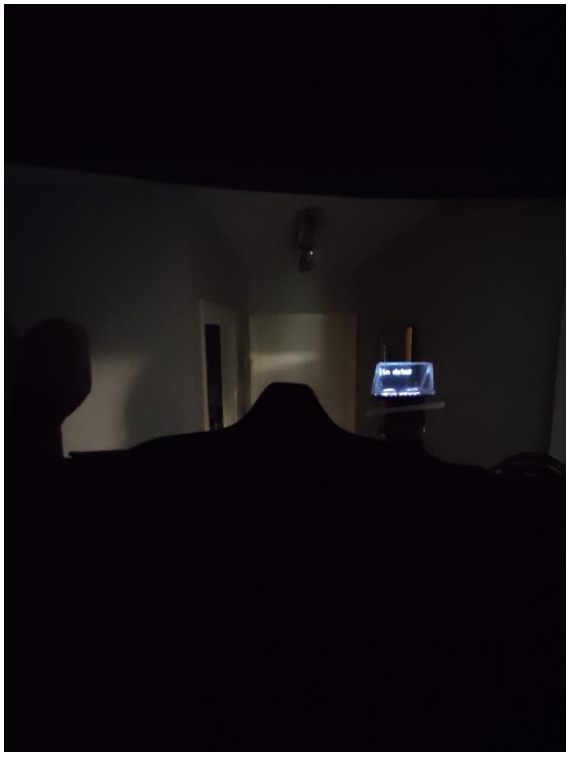


Figura 47. Vista desde el interior del casco en entorno oscuro.

Como puede comprobarse, es un dispositivo bastante grande, aunque tampoco en exceso. Sin embargo, hay que tener en cuenta que es el primer prototipo, por lo que hay mucho margen de mejora. Para reducir el tamaño, puede tanto optimizarse el diseño de la PCB como elegir componentes más pequeños (Arduino Pro Mini en vez del ESP32, otro módulo GPS, otra pantalla, etc.).

6. SOFTWARE

6.1. ARDUINO IDE

Para programar el microcontrolador se pueden utilizar una gran variedad de lenguajes, como C, microPython o JavaScript, pero debido a su facilidad de uso se empleará el Arduino IDE (Entorno de desarrollo integrado, o Integrated Development Environment, por sus siglas en inglés), con el que se escribirá, compilará y cargará el programa en el ESP32. Al igual que el hardware usado en el proyecto, es libre y gratuito, y puede utilizarse desde cualquier sistema operativo.

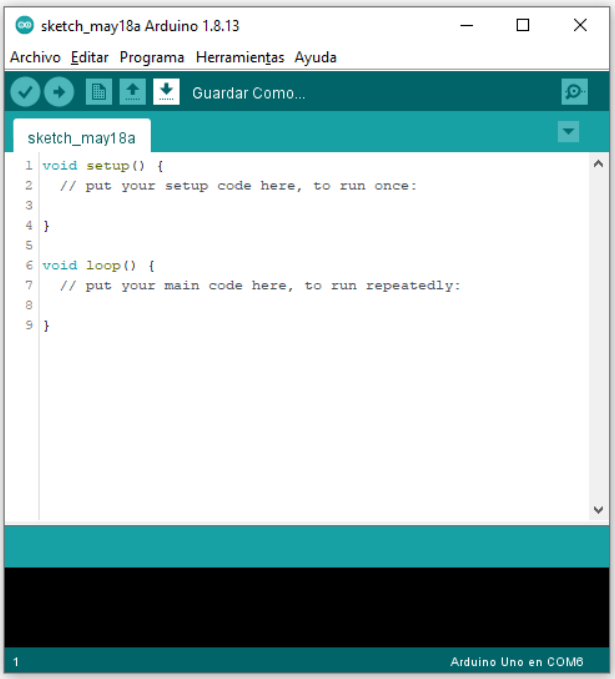


Figura 49. IDE de Arduino

La estructura general de un programa en Arduino suele ser la siguiente:

1. Definición de variables, constantes y librerías que se van a utilizar.
2. Void setup(), en el cual se realizan acciones que únicamente se van a repetir una vez (como por ejemplo inicial la comunicación serial).
3. Void loop(), en el cual se pone el código principal del programa que se va a repetir de manera continuada.
4. Void_nombre_funcion() en la que se define una función, la cual se llamará desde el programa principal.

Para poder programar el ESP32 con el Arduino IDE, es necesario primero deberemos añadir la nueva tarjeta de desarrollo, para que el programa la reconozca y pueda gestionarla correctamente. Para ello habrá que ir a preferencias y añadir la URL que proporciona el fabricante donde se indica.

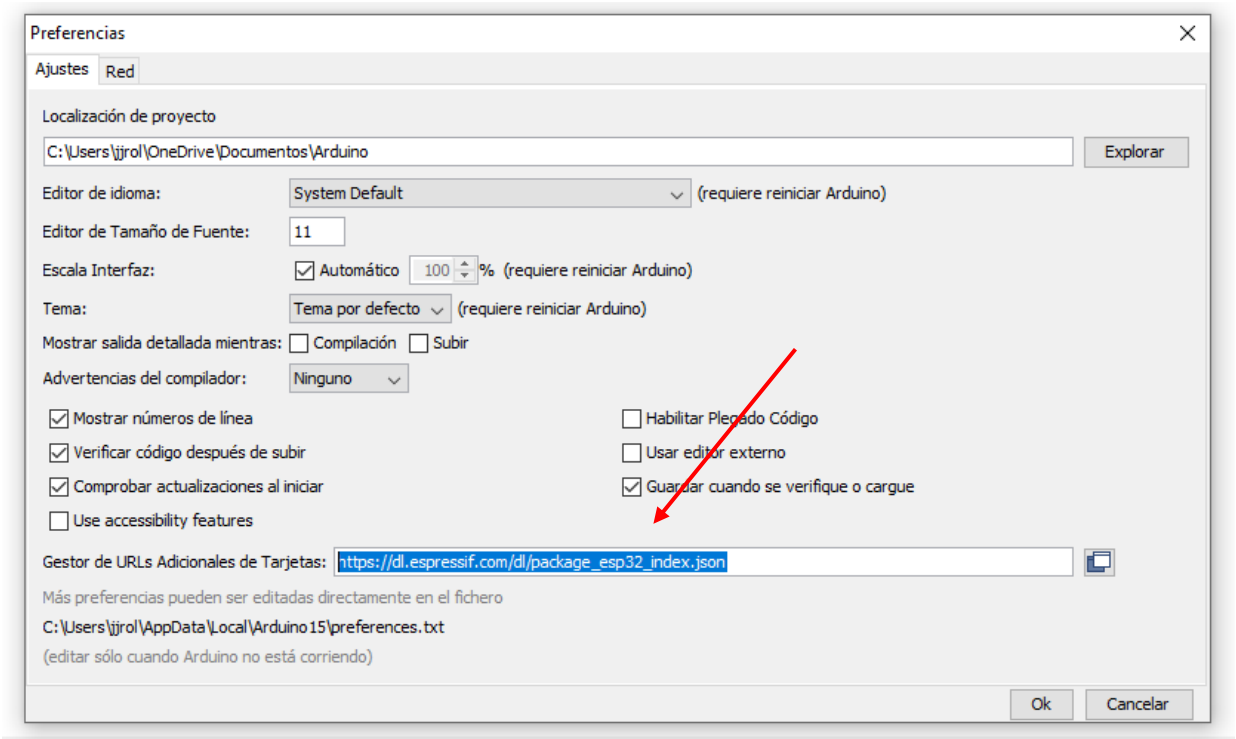


Figura 50. Procedimiento para añadir el ESP32 al entorno de desarrollo de Arduino.

A continuación, se seleccionará la placa de desarrollo en concreto, ya que hay una gran variedad de tipologías de ESP32. En este caso, se utilizará el "Dev Module" o módulo de desarrollo, que es el básico.

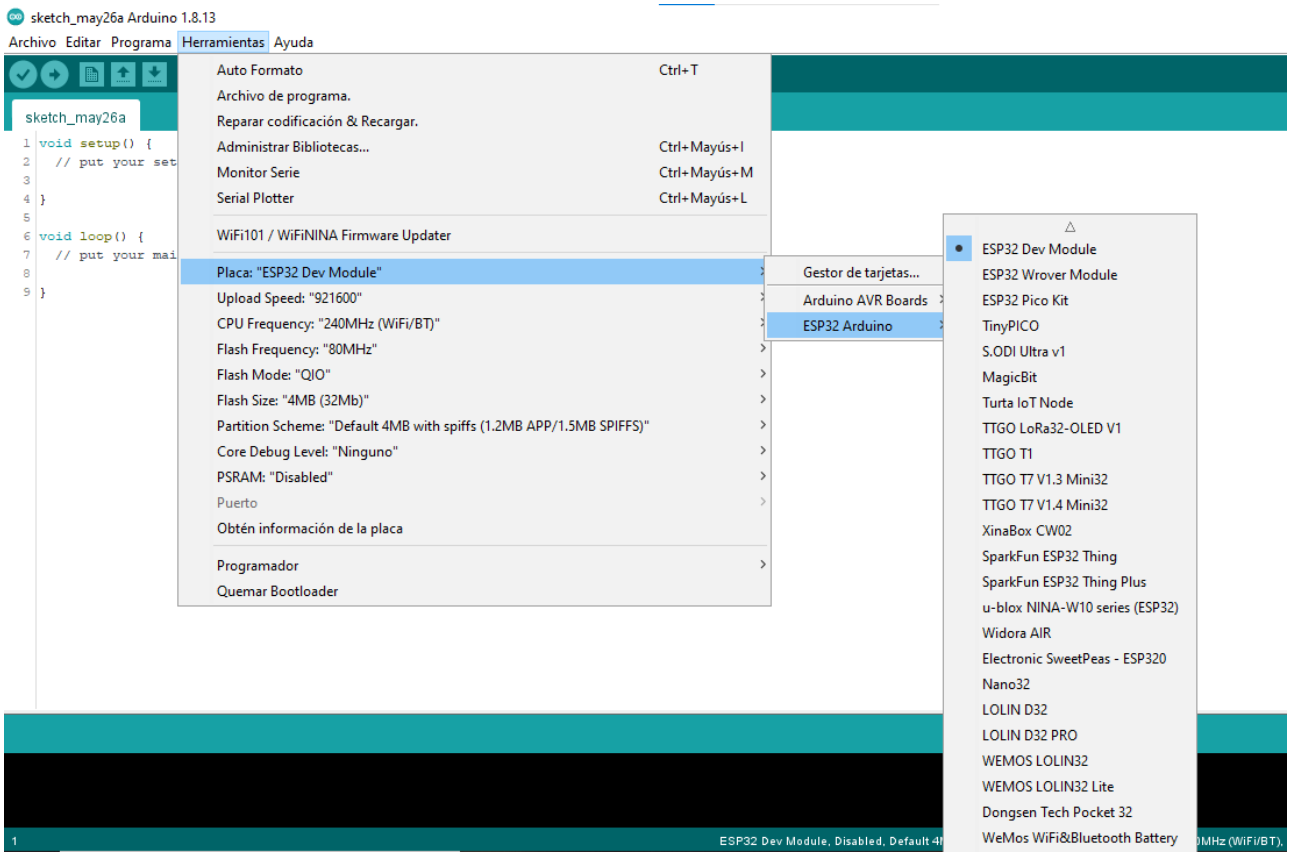


Figura 51. Selección del ESP32.

Tras esta selección, ya está todo listo para programar el ESP32 como si fuera un Arduino. Lo único que habrá que tener en cuenta es que, tras compilar el programa (que tardará algo más que con un Arduino normal), habrá que presionar el botón de "Boot" del ESP32 cuando el programa lo indique.

6.2. DIAGRAMAS DE FLUJO

El objetivo final del programa es proyectar en una pantalla OLED la velocidad del dispositivo y porcentaje de la batería que lo alimenta. Para ello, el código contará con dos partes claramente diferenciadas: el programa principal, que gestiona la conexión del módulo GPS con los satélites, y la función print_speed(), a la cual se llama desde el programa principal y que se encargará de leer y mostrar por la pantalla la velocidad y el porcentaje de batería en el caso de que esta información esté disponible. Ambos diagramas se muestran a continuación.

Diagrama general:

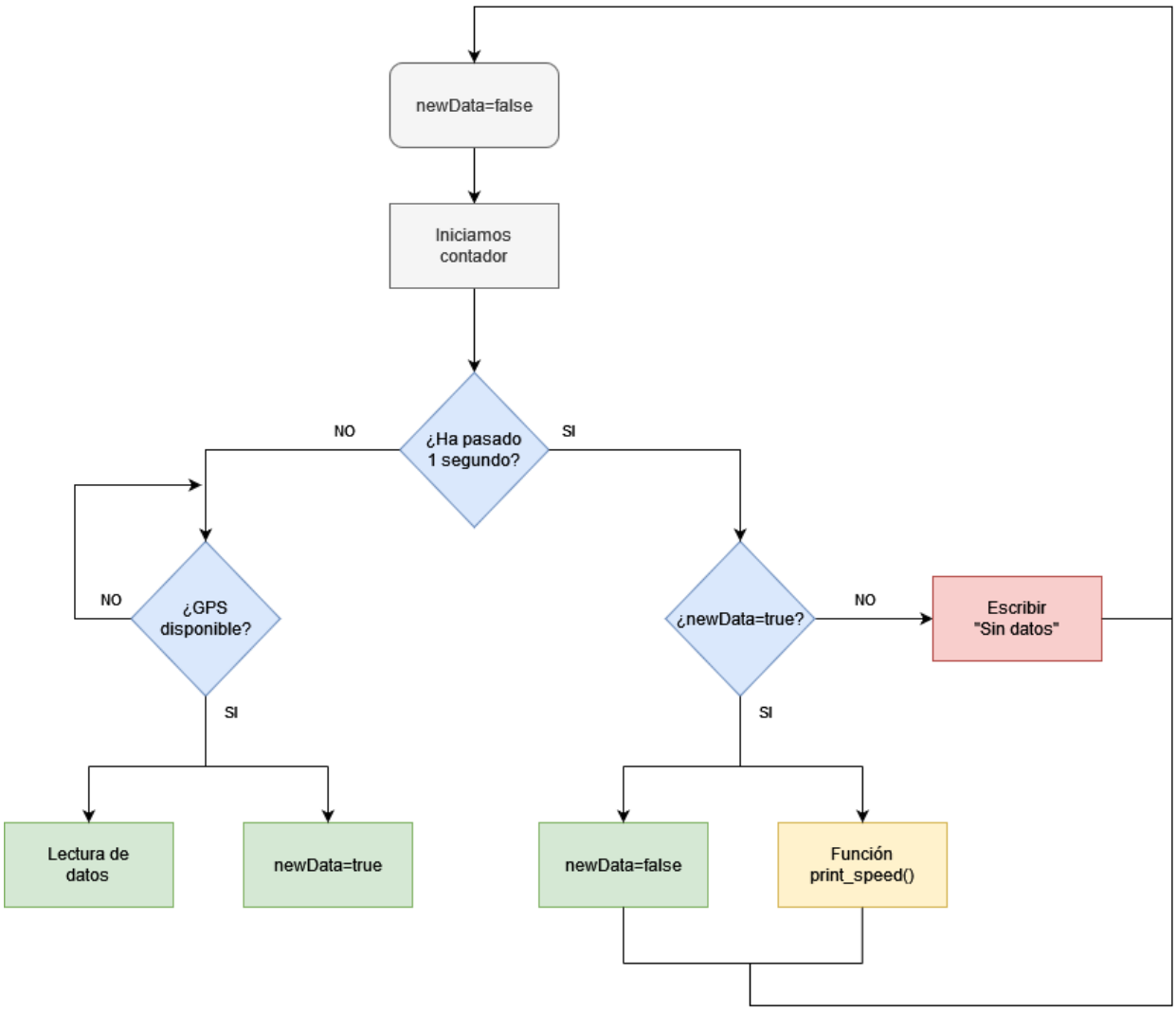


Figura 52. Diagrama de flujo general

Diagrama función print_speed():

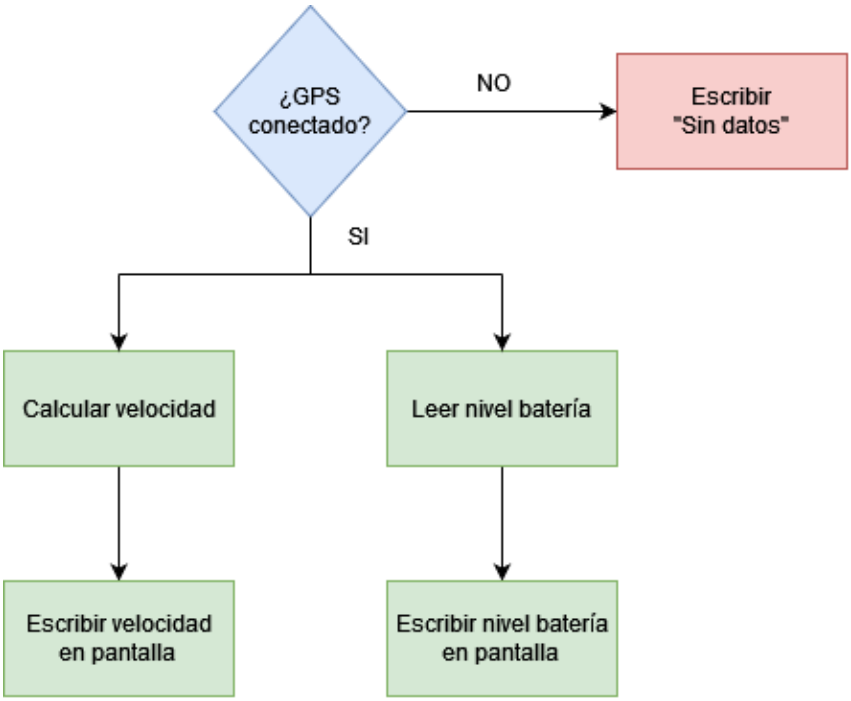


Figura 53. Diagrama de flujo de la función print_speed()

El programa comienza definiendo la variable "newData" como falsa. A continuación, se iniciará un contador, de manera que, mientras no haya pasado 1 segundo, el microcontrolador comprobará si el GPS está disponible. Si lo está, leerá que hay datos y cambiará el estado de la variable "newData" a verdadera. Si no lo está, se quedará en ese bucle hasta que pase el segundo entero del contador inicial.

Una vez ha pasado el segundo de tiempo, el microcontrolador chequeará si la variable "newData" es verdadera. Si no lo es, querrá decir que el GPS no ha estado disponible durante el segundo anterior, por lo que escribirá "Sin datos" en la pantalla OLED para comunicárselo al usuario, y volverá al inicio del programa. Si la variable sí que es verdadera, se cambiará a falsa (como si se bajase un flag), se llamará a la función "print_speed()" y se volverá al inicio del programa.

La función "print_speed()" comienza comprobando si se ha conseguido establecer conexión con algún satélite GPS (puede estar disponible pero no haber conseguido establecer conexión). Si se ha conseguido, el microcontrolador calculará la velocidad con esos datos y leerá el sensor de carga de la batería, y mostrará esos datos por la pantalla. Si no, mostrará un mensaje de "Sin datos" al igual que antes.

6.3. PROTOCOLOS DE COMUNICACIÓN UTILIZADOS

Como es de esperar, nos valdremos de protocolos de comunicación en serie, puesto que, aunque el paralelo es más rápido, también necesita de una línea (cable) por bit, además de otra para masa y otra para alimentación. Se han utilizado los siguientes protocolos:

6.3.1. I2C

Sistema síncrono que únicamente necesita 2 líneas de comunicación:

-SDA: Serial Data

-SCL: Serial Clock

Aunque no es el protocolo más rápido, puede generar una red de maestros/esclavos mucho más interconectada, ya que puede tener más de 1 maestro, aun usando únicamente 2 cables. Cada mensaje empieza/termina mediante solicitudes del maestro.

Se ha utilizado para la comunicación entre el sensor de carga de la batería y el ESP32

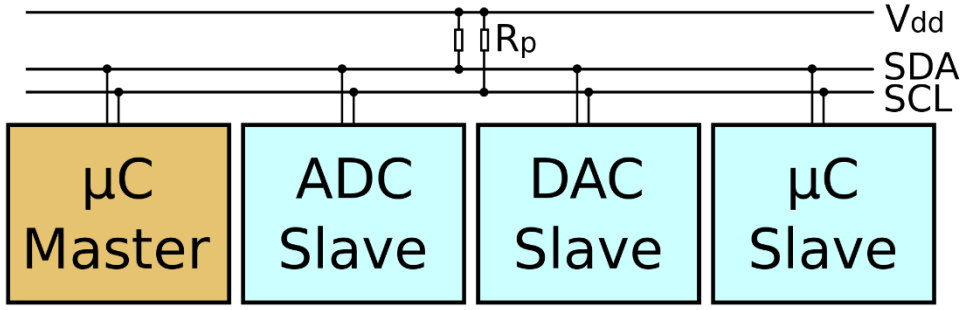


Figura 54. Esquema comunicación I2C (maestro microcontrolador y esclavos ADC, DAC y otro microcontrolador)

6.3.2. SPI

Sistema síncrono que necesita de 4 líneas de comunicación:

-SCK: Serial Clock

-MOSI: Master Out Slave In (línea de envío de datos)

-MISO: Master In Slave Out (línea de recibimiento de datos)

-SS: Slave Select (línea de selección de esclavo)

Sólo hay 1 maestro, varios esclavos. Los esclavos no pueden comunicarse entre ellos. Es la forma de comunicación serial más rápida.

Las pantallas OLED habitualmente se comunican con los microcontroladores mediante I2C. Sin embargo, con el objetivo de utilizar los distintos tipos de protocolos de comunicación que ofrece el microcontrolador ESP32 para comprobar su funcionamiento, se ha utilizado SPI para comunicarla con el microcontrolador.

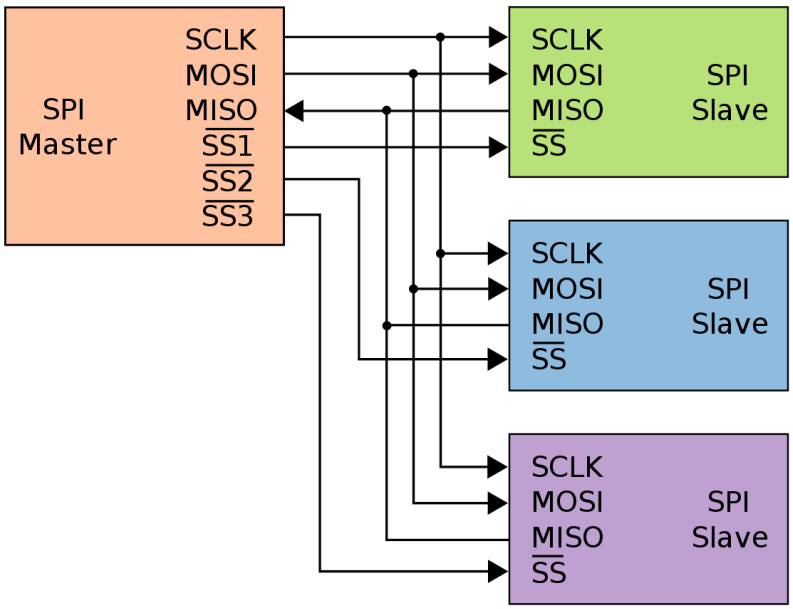


Figura 55. Esquema comunicación SPI (un maestro y tres esclavos)

6.3.3. UART

Asíncrono, necesita solo de 2 cables:

-Tx: Transmisor

-Rx: Receptor

Únicamente permite conectar 2 dispositivos, que en este caso son el GPS y el ESP32.

6.4. LIBRERÍAS UTILIZADAS

6.4.1. HARDWARESERIAL

Permite la comunicación serial **UART** con el microcontrolador a través del pin que se elija, en este caso el 1 para comunicar con el **GPS**.

```
32 //-------GPS (UART)----------------------------------
33
34 #define rxPin 16
35 #define txPin 17
36 HardwareSerial neogps(1);
```

Figura 56. Código HardwareSerial

6.4.2. WIRE

Permite la comunicación serial I2C con el microcontrolador. En el caso del ESP32, el pin SDA (datos serie) es el 21, mientras que el SCL (reloj serie) es el 22, y se usará para transmitir la información del lector de nivel da carga de batería.

6.4.3. SPI

Permite la comunicación SPI con el microcontrolador siempre que este actúe como maestro. En este caso se utiliza para la transmisión de información a la pantalla OLED (ver librería *Adafruit_SSDD1306* para más detalle)

6.4.4. ADAFRUIT_SSDD1306

Es una librería para controlar las pantallas OLED (SPI o I2C) basadas en el controlador SSDD1306, como es este caso. A continuación, puede verse un extracto del código en el que se configura el tamaño de la OLED, los pines de conexión y el protocolo de comunicación SPI.

```
21 //-------OLED (SPI)-----------------------------
22
23 #define Alto_pantalla 32    // OLED
24 #define Ancho_pantalla 128 // OLED
25 #define OLED_DC     0
26 #define OLED_CS     15
27 #define OLED_RESET  4
28
29 Adafruit_SSD1306 display(Ancho_pantalla, Alto_pantalla,
30   &SPI, OLED_DC, OLED_RESET, OLED_CS);
```

Figura 57. Código configuración OLED

6.4.5. TINYGPS++

Traduce los datos obtenidos del módulo GPS a valores entendibles. Algunas de los datos que proporciona a partir del GPS son:

- Latitud: *gps.location.lat()*
- Longitud: *gps.location.long()*
- Altitud: *gps.altitude.meters()*
- Velocidad (km/h): *gps.speed.kmph()*
- Número de satélites conectados: *gps.satellites.value()*

La mayoría de los datos pueden mostrarse en diferentes unidades (metros, millas, nudos…), pero, como estamos en España, se mostrará la velocidad en

kilómetros por hora, como puede verse a continuación. Se ha asignado el valor de la velocidad a una variable de números enteros para no mostrar decimales por la pantalla.

```
140 //-------VELOCIDAD-----------------------------
141
142   int velocidad;
143
144   display.setCursor (0,0);
145   display.setTextSize(4);
146   velocidad=gps.speed.kmph();
147   display.print(velocidad);    //Muestra la velocidad en km/h obtenida por GPS
148   display.setCursor (50, 15);
149   display.setTextSize(2);
150   display.print(" km/h");
```

Figura 58. Código parámetros GPS

Como se ha explicado previamente, la información mostrada por la pantalla deberá de estar invertida para que, una vez reflejada en el prisma, el usuario la vea de manera correcta. También, aunque sea una pantalla OLED monocromática, deberá de definirse el color en el que se va a escribir (blanco en este caso). Esto se consigue con los siguientes comandos:

```
131   display.clearDisplay();
132   display.ssd1306_command(0xA0);
133   display.setTextColor(WHITE);
```

Figura 59. Código para configurar cómo se muestra la información.

6.4.6. DFROBOT_MAX17043

Librería proporcionada por DFROBOT para poder utilizar de manera simple el lector de carga de la batería Lipo (*readPercentege()*). Al igual que con la velocidad, se asigna la lectura a una variable de números enteros para no mostrar decimales por la pantalla.

```
150 //-------BATERIA--------------------------------
151   int nivel;
152 
153   display.setCursor (70, 0);     // ubica cursor en coordenas
154   display.setTextSize(2);
155   nivel=gauge.readPercentage(); //Asigna al integer 'nivel' el valor del %
156                                 //de batería,que es decimal (queremos entero)
157   display.print(nivel);
158   display.println(" %");
159 
160    display.display();
161   }
```

Figura 60. Código lector carga Lipo

7. ENSAYOS Y RESULTADOS

7.1. TEST DE DESCARGA

Como en cualquier dispositivo electrónico autónomo, la duración de la batería es un aspecto clave. En este caso, el objetivo será que aguante una salida en moto entera sin necesidad de recarga. Ese tiempo se estima en unas 6h.

Para realizar el test, se dejó el dispositivo cargado funcionando (habiendo establecido conexión GPS) en una terraza. El tiempo de descarga de la batería de 2000mAh fue de en torno a 7 horas y 15 minutos, por lo que cumple el requisito holgadamente.

Si se quisiera aumentar la autonomía, podría optimizarse el código haciendo que el microcontrolador entrase en modo sleep periódicamente, o haciendo que la pantalla OLED parpadease a una frecuencia a la cual el ojo humano la viese como encendida pero en verdad se está encendiendo y apagando a una velocidad muy alta.

7.2. TIEMPO DE CONEXIÓN A GPS

Un aspecto importante que es necesario estudiar es el tiempo que tarda el dispositivo comenzar a proporcionar valores de velocidad. Este estudio se llevó a cabo en una terraza junto a un edificio, por lo que la señal obtenida era esperablemente peor que en campo abierto. Tras realizar varias pruebas, los resultados fueron los siguientes:

- El tiempo que tarda desde que se enciende por primera vez hasta que establece una conexión adecuada con los satélites es de 1 minuto y 44 segundos.
- Tras esta primera conexión, se fue apagando durante 10 segundos y volviendo a encender, estableciendo conexión tras el apagado en una media de 12 segundos.
- Se repitió el procedimiento anterior pero esta vez apagando durante 1 minuto y se obtuvo un resultado similar, 10 segundos.

Es necesario destacar que la presencia de elementos entre los satélites y la antena GPS del prototipo interfiere mucho en el tiempo de la primera conexión. Quizás ese es el motivo por el cual los tiempos de conexión difieren tanto con los proporcionados con el fabricante. De hecho, dentro de una

habitación (aunque se esté cerca de una ventana) ha sido imposible establecer conexión.

7.3. PRUEBAS DE CAMPO

Para comprobar el funcionamiento del módulo GPS utilizado, se realizó un test en el que, yendo en coche, se comparó la velocidad obtenida mediante el GPS utilizado en el proyecto y el GPS del teléfono móvil, ya que este último tiene una precisión bastante buena y servirá como referencia. Los resultados mostraron que, aunque cuando hay cambios bruscos de velocidad al GPS del proyecto le cuesta algo más de tiempo ajustarse, en general la respuesta es muy similar y, por lo tanto, satisfactoria.

Un vídeo realizando el tests puede verse en este enlace o escaneando el código QR a continuación.

Figura 61. Enlace QR al vídeo de prueba del módulo GPS:

En cuanto al dispositivo final, como puede verse en la *Figura 62*, muestra de manera adecuada los valores obtenidos en el prisma, permitiendo ver a través de este. Sin embargo, se ha comprobado que, si el día es excesivamente soleado, la potencia de la pantalla no es suficiente para proyectar de manera adecuada la información.

Figura 62. Prisma mostrando la velocidad y porcentaje de carga.

8. ANÁLISIS DE VIABILIDAD COMO NEGOCIO

A la vista de los resultados obtenidos y del nicho de mercado que podría cubrir el dispositivo diseñado, se va a realizar un pequeño análisis sobre la viabilidad de la idea como negocio. Primeramente, se mostrará y analizará el cuadro Canvas, de manera que puede comprobarse la solidez de la idea como negocio. Seguidamente se estudiará la aplicación de la metodología Lean-Startup para analizar cómo podrían mejorarse las posibilidades de éxito del negocio, y finalmente se hará un pequeño estudio económico para analizar la rentabilidad del mismo.

8.1. CUADRO CANVAS

El cuadro Canvas es una herramienta de gestión que permite conocer los puntos fuertes y débiles de un negocio, indicando cómo direccionar los esfuerzos para tener la mayor probabilidad de éxito. El cuadro Canva de un negocio basado en la venta del dispositivo que se está desarrollando en este documento sería el siguiente:

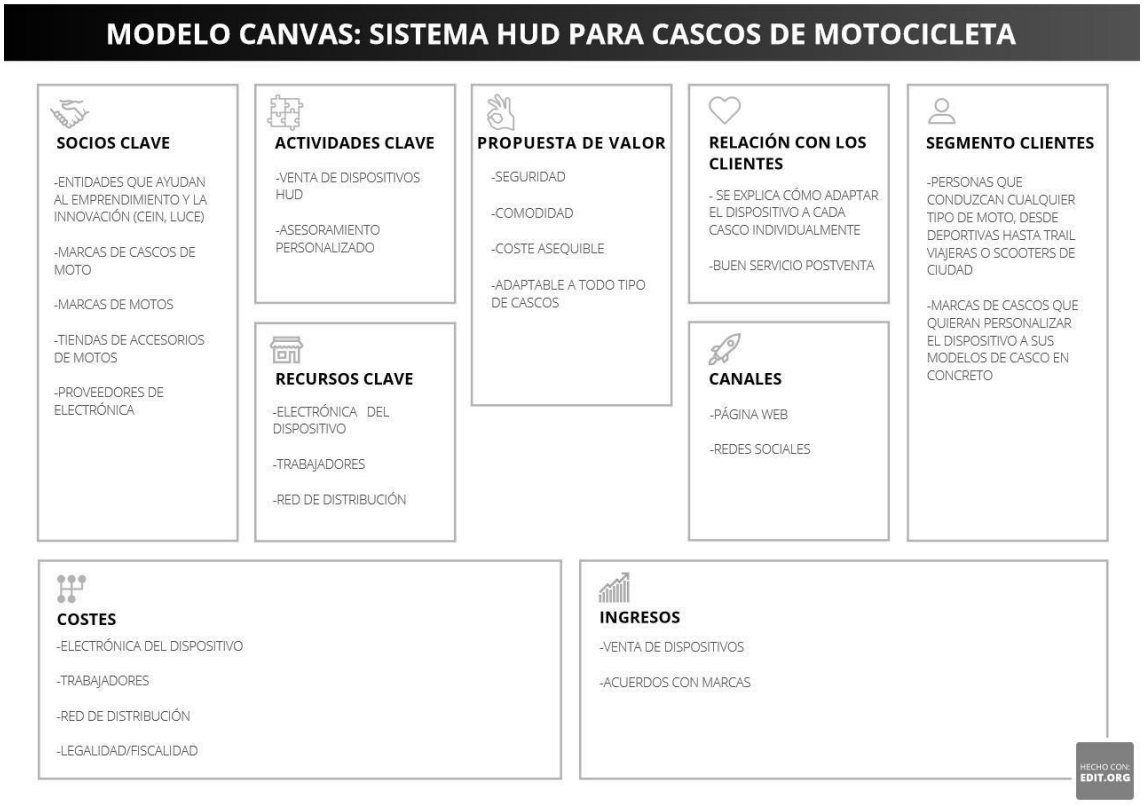


Figura 63. Cuadro Canvas del modelo de negocio

8.1.1. Propuesta de valor

Se refiere a qué resolvemos al cliente, así como por qué deberían elegirnos a nosotros en vez de a la competencia. Las dos ideas principales son la seguridad y la comodidad, ya que el objetivo del producto es permitir al motorista ver de manera continua la velocidad sin desviar la mirada de la carretera. Como queremos que el producto tenga el mayor público posible dentro de los motoristas, también haremos que sea adaptable a cualquier casco. Además, y teniendo en cuenta que la mayoría de los accesorios en el mundo de las motos suelen ser muy caros, haremos que el dispositivo sea de un precio asequible, algo que es factible ya que se basa en electrónica básica.

8.1.2. Segmento clientes

El cliente principal son los motoristas con interés por nuevas tecnologías que les puedan ayudar a mejorar su seguridad en la carretera. En comparación con los coches o camiones, las motos son un usuario de riesgo en la carretera, ya que en un accidente siempre van a salir perdiendo, por lo que mejorar la seguridad siempre es un objetivo.

El otro tipo de clientes de este dispositivo son las marcas de cascos con interés por personalizar el dispositivo a sus productos. Es una estrategia similar a la que usan los fabricantes de intercomunicadores, ya que hay los generales que valen para cualquier casco, y luego algunas marcas incorporan los suyos directamente.

Figura 64. Intercomunicador genérico

Figura 65. Intercomunicador integrado

8.1.3. Relación con los clientes

De manera similar a los intercomunicadores genéricos, este dispositivo podrá adaptarse a cualquier modelo de casco. Sin embargo, y aunque se diseñará de manera que su instalación sea lo más intuitiva posible, siempre habrá clientes a los que les cueste instalar este tipo de dispositivos, por lo que se ofrecerá in servicio personalizado de ayuda para quien lo necesite, estableciendo así una relación de confianza con el cliente.

Habrá también un servicio post venta que se encargará de tratar con temas como la garantía, reparación de posibles averías o recepción de sugerencias.

8.1.4. Canales

Puesto que se trata de vender un producto físico, los canales de comunicación con el cliente serán la página web y las redes sociales. Se publicará los productos/versiones disponibles, así como novedades en cuanto a trabajos con otras marcas del mundo de las motos.

Tanto desde la web como desde las redes se podrán recibir quejas, propuestas o inquietudes de los clientes.

8.1.5. Actividades clave

El núcleo de este negocio será la venta de los dispositivos al público ya nombrado anteriormente, ya que de ahí saldrá prácticamente el 100% del beneficio. Sin embargo, el asesoramiento personalizado será importante también, ya que establecerá una buena relación con los clientes, de manera que mediante el "boca a boca" se ampliará nuestro público, llegando de esta manera a interesar nuestros servicios a marcas de cascos.

8.1.6. Recursos clave

Los recursos clave para que esta idea de negocio funcione son, por una parte, los trabajadores de la empresa, puesto que son ellos quienes programan, construyen y mejoran el producto. Claramente, el otro recurso principal será la electrónica en sí con la que se construyen los dispositivos. Con esos dos elementos, el negocio podría empezar ya a producir el producto, por lo que lo único que quedaría sería tener una buena red de distribución del producto para hacerlo llegar de manera eficiente a la mayor cantidad de clientes posibles en el menor tiempo

8.1.7. Socios clave

Durante el nacimiento de este negocio, sería fundamental el apoyo de entidades que apoyan a la innovación, como el CEIN (Centro Europeo de Empresas e Innovación de Navarra) o el LUCE (Laboratorio Universitario de Creación de Empresas), puesto que podrán aportar una gran experiencia y recursos para hacer que el negocio funcione.

Los acuerdos con proveedores de electrónica serían fundamentales también, ya que al comprar en grandes cantidades suelen ofrecerse precios más competitivos, y es algo que interesa para conseguir el que producto final tenga el menor coste posible.

Finalmente, sería interesante llegar a acuerdos con marcas de cascos para tener así un número fijo de ventas, aunque también a marcas de motos ya que quizás les interesa adaptar esa tecnología de alguna manera. Los acuerdos con tiendas de accesorios serían también fundamentales, puesto que es ahí donde la mayoría de los motociclistas van a comprar, y queremos tener la mayor visibilidad posible.

8.1.8. Gastos

Los gastos serán principalmente lo invertido en los recursos clave, es decir, los salarios de los trabajadores, la electrónica, y la red de distribución. Los temas fiscales serán inevitables para establecer legalmente la empresa también.

Ya que no se requiere mucho espacio para producir el producto, inicialmente no sería necesario alquilar un espacio en específico para este fin, ya que se podría hacer en cualquier sitio que alguno de los socios crea conveniente. Sin embargo, si el negocio aumentase en gran medida, sí que sería necesario alquilar un sitio específico para trabajar, por lo que en esa situación sería un gasto también.

8.1.9. Ingresos

Los ingresos principales vendrían de la venta del producto, bien sea desde la página web o desde las tiendas de accesorios. Los acuerdos con marcas de cascos serían un ingreso más seguro en comparación con este, puesto que las marcas te comprarían números mucho mayores para implementar en sus cascos.

8.2. METODOLOGÍA LEAN-STARTUP

Para intentar asegurar el éxito este modelo de negocio, se utilizaría la metodología Lean-Startup, cuyo eje principal es el cliente. Los pasos de este método son los siguientes:

8.2.1. Hipótesis

Tras estudiar a los clientes y sus necesidades, se plantea una primera hipótesis en cuanto a la resolución de un problema que creemos que funcionará en el mercado. En este caso, la hipótesis sería: "*los usuarios de motocicletas quieren evitar al máximo desviar la mirada de la carretera, por lo que un dispositivo que les indique su velocidad de manera continuada sin entorpecer la visión les interesaría*".

8.2.2. Producto Mínimo Viable (MVP)

Es la mínima cantidad de producto básico, es decir, que tienen suficientes características para satisfacer a los usuarios iniciales, que se pueden fabricar para comprobar si la hipótesis es correcta. En este caso se fabricarían 4 prototipos iniciales, y como cada prototipo lo puede utilizar más de un usuario (es de quita y pon, y se adapta a cualquier casco), y se distribuiría de la siguiente manera hasta haber recogido los datos suficientes:

- Tiendas de accesorios de motos: se buscarán 2 de las tiendas más grandes de Pamplona (hay varias) y se dejar 1 prototipo en cada una, enseñando al vendedor a colocar el dispositivo en un casco. Se permitirá que quien quiera pueda usarlo libremente durante un tiempo limitado, con el objetivo de que varios usuarios lo prueben cada día, y con la única condición de ofrecer un feedback tras su utilización mediante una encuesta.

- Amigos y conocidos moteros: se dejarán 2 prototipos a los amigos y conocidos moteros para obtener su feedback. El mundo de las motos es un mundo relativamente cerrado, por lo que una manera fácil de llegar a más gente es a través de amigos moteros de amigos moteros.

- Concentraciones moteras: la concentración motera más grande de Navarra es la NavaRider Day, que se celebra todos los años en octubre y a la que acuden más de mil motoristas. El acto principal de la concentración es la ruta en moto, pero después hay merchandising,

sorteos, actividades… Podría colocarse un puesto para que, al igual que en las tiendas, quien quiera pueda comprobar el funcionamiento del dispositivo. A lo largo del año hay varias concentraciones de este tipo, por lo que 2 prototipos en cada una sería ideal, ya que los testeadores de estas concentraciones serán el cliente potencial más claro. Estos 2 prototipos serán los que se han dejado a amigos y conocidos, ya que las concentraciones son días puntuales y durante esos días nos interesa que nuestros prototipos estén ahí.

Figura 66. Grupo de motoristas en una NavaRider

8.2.3. Métrica relevante

Tras haber lanzado el producto mínimo viable, el siguiente paso sería estudiar los datos recogidos que se consideran indicadores de si la hipótesis ha sido acertada o no. En este caso serán, por una parte, el número de usuarios que han utilizado los prototipos, ya que, si el número es alto, indica que hay un interés generalizado, y por otra parte los resultados de las encuestas. Este último es el más importante, ya que los usuarios deberán de valorar de manera numérica su grado de satisfacción con el producto, así como otras métricas como comodidad, facilidad de instalación, utilidad… Se usará la misma encuesta en todos los casos para obtener unos datos homogéneos.

8.2.4. Feedback y pivotaje

Gracias a las encuestas se sabrá si el producto básico cumple la hipótesis inicial. Si se observan tendencias generalizadas en las respuestas respecto a temas de mejora y temas que han gustado mucho, se pivotará en esa dirección de manera que el esfuerzo vaya siempre bien encaminado.

Tras aplicar las mejoras que se consideren, se sacará el MVP2 y se repetirá el proceso. Si la respuesta general es positiva, se acelerará el crecimiento del negocio invirtiendo el dinero necesario de manera que se pueda escalar lo más rápidamente posible. En este caso en concreto implicaría no solo empezar a fabricar el producto final, sino también establecer acuerdos finales con las tiendas y marcas de cascos mostrándoles los datos obtenidos de las encuestas.

8.3. VIABILIDAD ECONÓMICA

Los precios de los elementos (se han buscado en páginas de bajo coste) se muestran en una tabla a continuación:

Dispositivo	ESP32	GPS	Batería	Carga/boost 5V	Sensor batería	Pantalla OLED	Carcasa 3D	Pegatinas para el casco	PCB
Precio (€)	3'93	3'06	4'48	0'90	8'87	3'14	5	1'79	2

Tabla 1. Precios elementos

El precio de la PCB se ha estimado teniendo en cuenta los precio que ofrecen fabricantes como JLC PCB, y el de la carcasa 3D, por su peso.

Así pues, el precio aproximado de fabricación por unidad sería de 30€, pero habría que tener en cuenta que, al hacer grandes pedidos, el coste de estos será menor. A continuación, se muestra una tabla con 4 posibles escenarios: uno muy malo en el que solo se vende el 25% de lo producido, uno un poco menos malo en el que se vende el 50%, uno bueno en el que se vende el 75% y el mejor, en el que se vende el 100% de unidades producidas. Se ha tomado como ejemplo 50 unidades producidas, y para cada porcentaje de venta se han establecido 4 precio diferentes, desde 50€ hasta 150€, es decir, algo menos del 10% de un casco convencional.

Unidades producidas	Coste/unidad (€)	Precio venta (€)	Ventas esperadas (%)	Inversión (€)	Ganancias totales (€)	Beneficios (€)
50	30	50	25	1500	625	-875
50	30	75	25	1500	937,5	-562,5
50	30	100	25	1500	1250	-250
50	30	150	25	1500	1875	375

50	30	50	50	1500	1250	-250
50	30	75	50	1500	1875	375
50	30	100	50	1500	2500	1000
50	30	150	50	1500	3750	2250
50	30	50	75	1500	1875	375
50	30	75	75	1500	2812,5	1312,5
50	30	100	75	1500	3750	2250
50	30	150	75	1500	5625	4125
50	30	50	100	1500	2500	1000
50	30	75	100	1500	3750	2250
50	30	100	100	1500	5000	3500
50	30	150	100	1500	7500	6000

Tabla 2. Estudio económico en función de ventas y precios

Se ha realizado un primer sondeo por redes sociales mediante una encuesta para para comprobar el interés del público motero en la idea, y saber más o menos cuanto estarían dispuestos a pagar. Los datos obtenidos se muestran a continuación:

Figura 67. Encuesta pregunta 1

Figura 69. Encuesta pregunta 2

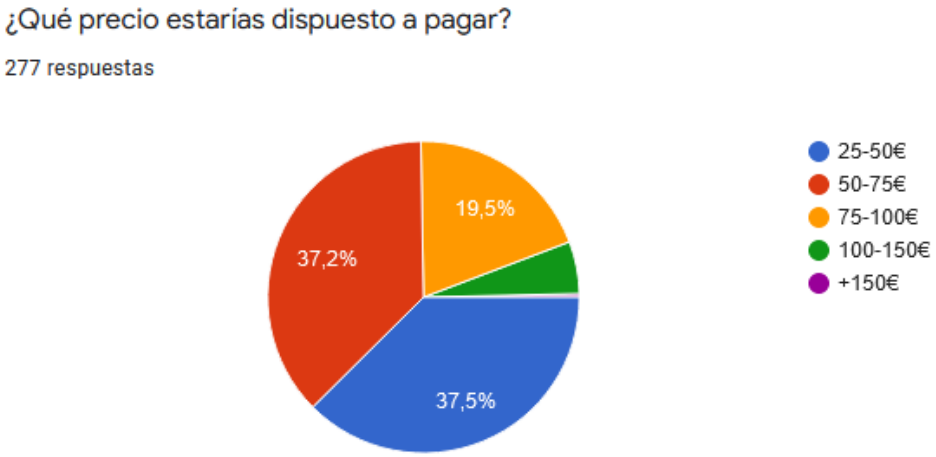


Figura 68. Encuesta pregunta 3

De 277 personas, 104 dijeron entre 25-50€, 103 entre 50-75€, 54 entre 75-100€, 15 entre 100-150€ y 1 +150€. Cabe recalcar que, al haberse hecho por redes sociales, el público al que ha llegado la encuesta ha sido más bien joven, y por lo general su poder adquisitivo suele ser menor que el de personas con mayor edad.

A la luz de los datos, he considerado que un buen precio de salida del producto serían 60€, puesto que cubre una gran parte del público encuestado y con un buen márketing se podría llegar a convencer a los indecisos del grupo que eligió 25-50€. Si repetimos la *Tabla* 1 con este precio, se obtendría lo siguiente:

Unidades producidas	Coste/unidad (€)	Precio venta (€)	Ventas esperadas (%)	Inversión (€)	Ganancias totales (€)	Beneficios (€)
50	30	60	25	1500	750	-750
50	30	60	50	1500	1500	0
50	30	60	75	1500	2250	750
50	30	60	100	1500	3000	1500

Tabla 3. Análisis económico a un precio de venta de 60€

Un Elevator pitch (explicación breve y concisa de 2-3 minutos sobre una idea de negocio con el objetivo de convencer a posibles inversores del potencial de la idea) de esta idea de negocio puede verse en este enlace, o escaneando el código QR a continuación.

Figura 70. Enlace QR a un Elevator pitch sobre esta idea de negocio

9. CONCLUSIONES

A la vista de los resultados obtenidos, se puede decir que el primer prototipo ha sido un éxito. El objetivo principal, que era mostrar la velocidad en tiempo real, se ha cumplido de manera adecuada, mientras que el resto se han cumplido también en mayor o menor medida. Los puntos a mejorar son, por una parte, el tamaño del dispositivo, que ya se ha comentado cómo podría resolverse, y por otro la potencia de la pantalla OLED para proyectar la información en el prisma de manera que sea utilizable en un día soleado.

En cuanto a la precisión de los datos obtenidos por el módulo GPS, se pueden dar por aceptables, ya que teniendo en cuenta el precio de este, los resultados son muy similares a los que se obtienen mediante el GPS de un teléfono móvil (ver apartado *Pruebas de campo*)

10. APRENDIZAJES

Al tratarse de un proyecto que involucra desde el planteamiento de la idea inicial hasta la fabricación de esta, ha habido numerosos aprendizajes que de otra manera habría sido imposible adquirir. El proceso para llegar al prototipo final ha sido el siguiente:

1. Planteamiento inicial
2. Análisis de viabilidad
3. Elección de componentes
4. Diseño (hardware y software)
5. Fabricación
6. Testeo

El entorno académico generalmente se centra solo el apartado 4, es decir, tanto el diseño del circuito como la programación. Esto hace que se pierda una parte muy interesante y enriquecedora del proceso, ya que no se fomenta el planteamiento y desarrollo de nuevas ideas. Es por ello por lo que considero que el hecho de haber pasado por el proceso completo de fabricación de un prototipo ha sido una experiencia altamente enriquecedora y que me ayudará sin duda en futuros proyectos.

En cuanto a aspectos más técnicos, el mayor aprendizaje ha sido el diseño de una PCB. Es una cualidad que cualquier interesado en el mundo de

la electrónica debería tener y que yo personalmente hasta este momento no había adquirido. Es por ello por lo que la primera PCB diseñada fue un fracaso, como ya se ha comentado. Sin embargo, de eso se aprendió y la segunda fue un éxito, pese a que podría haberse optimizado más.

11. LÍNEAS FUTURAS

Como se ha comentado previamente, en este proyecto únicamente se ha desarrollado un primer prototipo funcional, pero existe un gran abanico de posibles innovaciones y mejoras respecto al mismo.

La primera sería hacer el dispositivo más pequeño, optimizando su diseño. Para ello, se podría sustituir el ESP32 por un Arduino Pro mini, así como encontrar módulos de tamaño más reducido, y rediseñando la PCB se conseguirían grandes mejoras. Incluso podrían implementarse los circuitos tanto del microcontrolador como de los módulos en una PCB que recoja todas las conexiones, de manera que únicamente se implementen los pines y pistas que se van a utilizar, optimizando en gran medida el dispositivo.

Otra posible línea de desarrollo es convertirlo en un sistema IoT, ya que el ESP32 tiene Wifi y bluetooth integrados. Se podría programar el dispositivo para que, teniendo el teléfono móvil en modo router, se conecte al Wifi de este y envíe datos (posición, velocidad, tiempo de ruta…) a otro teléfono, o a un correo o base de datos que podamos consultar tras la ruta en moto. Podría programarse también para que, en caso de accidente (podría detectarlo por ejemplo con un acelerómetro), envíe las coordenadas con un mensaje de alerta a un contacto que elijamos.

El abanico de posibilidades que existe con este proyecto es tremendamente amplio, limitado prácticamente solo por la creatividad del diseñador. Es por ello por lo que se continuará trabajando en el mismo hasta obtener un producto lo más útil y comercial posible.

BIBLIOGRAFÍA Y REFERENCIAS

[1] Autor. (17 de mayo de 2022). *Seguridad activa y pasiva del vehículo*. Club Europeo de Automovilistas. Recuperado el 17 de mayo de 2022 de https://www.cea-online.es/blog/128-seguridad-activa-y-pasiva-del-vehiculo

[2] J. (6 de junio de 2018). *Review NUVIZ HUD, el gadget todo en uno para la moto*. Motos Garrido. Recuperado el 17 de mayo de 2022 de https://www.motosgarrido.com/blog/review-nuviz-hud/

[3] McMurphy, R. (3 de julio de 2017). *Nuviz All in one heads up display*. RideApart. Recuperado el 17 de mayo de 2022 de https://www.rideapart.com/reviews/253531/nuviz-all-in-one-heads-up-display-gear-review/

[4] Hughes, J. (11 de enero de 2020). *Nuviz HUDs stop working as company tanks*. RideApart. Recuperado el 17 de mayo de 2022 de https://www.rideapart.com/news/392362/nuviz-hud-stop-working-company-tanks/

[5] Rubio, J. (26 de abril de 2022). *Shoei Opticson, el revolucionario casco con sistema HUD*. Soy Motero. Recuperado el 17 de mayo de 2022 de https://www.soymotero.net/shoei-opticson-casco-moto-sistema-hud

[6] Beningo, J. (21 de enero de 2020). *Cómo seleccionar y usar el módulo ESP32 con Wifi/BT adecuado para una aplicación IoT industrial*. Digi-Key Electronics. Recuperado el 17 de mayo de 2022 de https://www.digikey.es/es/articles/how-to-select-and-use-the-right-esp32-wi-fi-bluetooth-module

[7] Clavijo, C. (17 de mayo de 2022). *Modelo Canvas: ¿Qué es para qué sirve y cómo se utiliza?* HubSpot. Recuperado el 17 de mayo de 2022 de https://blog.hubspot.es/sales/modelo-canvas

[8] Gonzalez, O. (12 de marzo de 2019). *Todo lo que necesitas saber sobre las baterías de Ion Litio. BricoGeek*. Recuperado el 17 de mayo de 2022 de

https://blog.bricogeek.com/noticias/tecnologia/todo-lo-que-necesitas-saber-sobre-la-baterias-de-ion-litio-lipo/

[9] Ahmadlogs. (19 de mayo de 2021) *Arduino IDE Examples- ESP32 GPS tracker*. https://github.com/ahmadlogs/arduino-ide-examples/tree/main/esp32-gps-tracker

[10] Boot & Work Corp. S.L., Support Team (18 de febrero de 2019). *Cómo utilizar la librería Software Serial en el controlador industrial Arduino PLC*. Industrial Shields. Recuperado el 17 de mayo de 2022 de https://www.industrialshields.com/es_ES/blog/blog-industrial-open-source-1/post/como-utilizar-la-libreria-de-software-serial-en-el-controlador-industrial-arduino-plc-99

[11] Guerra, J. (18 de noviembre de 2021). *SPI con Arduino y sensor BMP 280*. ProgramarFacil. Recuperado el 17 de mayo de 2022 de https://programarfacil.com/blog/arduino-blog/spi-con-arduino-bmp280/

[12] Hart, M. (20 de febrero de 2018). *TinyGPS+++*. Arduiniana. Recuperado el 17 de mayo de 2022 de http://arduiniana.org/libraries/tinygpsplus/

[13] I²C. (2022, 22 de abril). *Wikipedia, La enciclopedia libre*. Fecha de consulta: 09:30, mayo 18, 2022 desde https://es.wikipedia.org/w/index.php?title=I%C2%B2C&oldid=143065265.

[14] Serial Peripheral Interface. (2022, 22 de abril). *Wikipedia, La enciclopedia libre*. Fecha de consulta: 09:30, mayo 18, 2022 desde https://es.wikipedia.org/w/index.php?title=Serial_Peripheral_Interface&oldid=143065352.

[15] Autor. (30 de agosto de 2019). *13 problemas comunes de soldadura de PCBs que debes evitar*. Hacedores. Recuperado el 20 de mayo de 2022 de https://hacedores.com/13-problemas-de-soldadura-de-pcbs/

Fdo.:

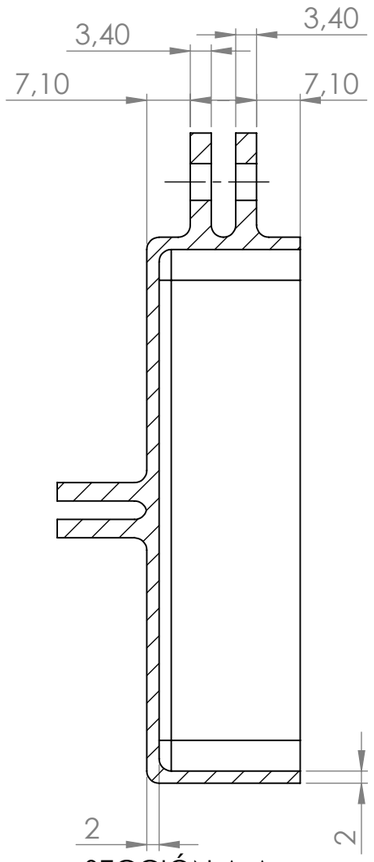
Jose Javier Roldán

ANEXOS

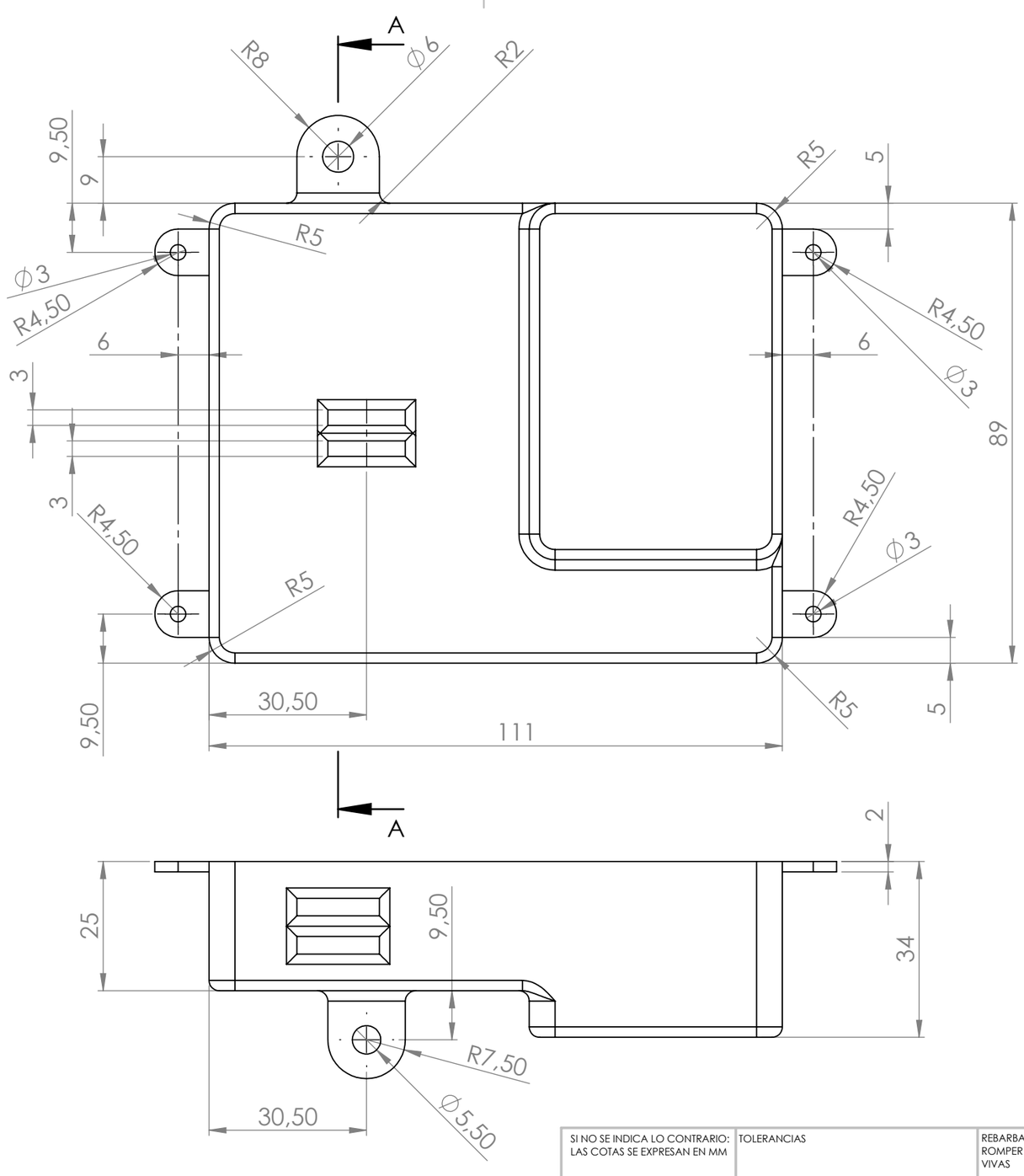
1. DATASHEETS

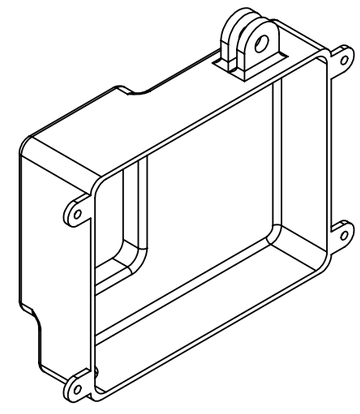
- ESP32: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf
- OLED: https://es.farnell.com/midas/mdob128032ev-ws/pantalla-oled-cob-128x32-p-xeles/dp/3154940
- Cargador/Boost a 5V: https://www.adafruit.com/product/2465
- Sensor de batería: https://www.digikey.be/htmldatasheets/production/3333927/0/0/1/dfr0563.html
- Batería: https://www.farnell.com/datasheets/2369105.pdf
- GPS: https://content.u-blox.com/sites/default/files/products/documents/NEO-6_DataSheet_%28GPS.G6-HW-09005%29.pdf

2. PLANOS



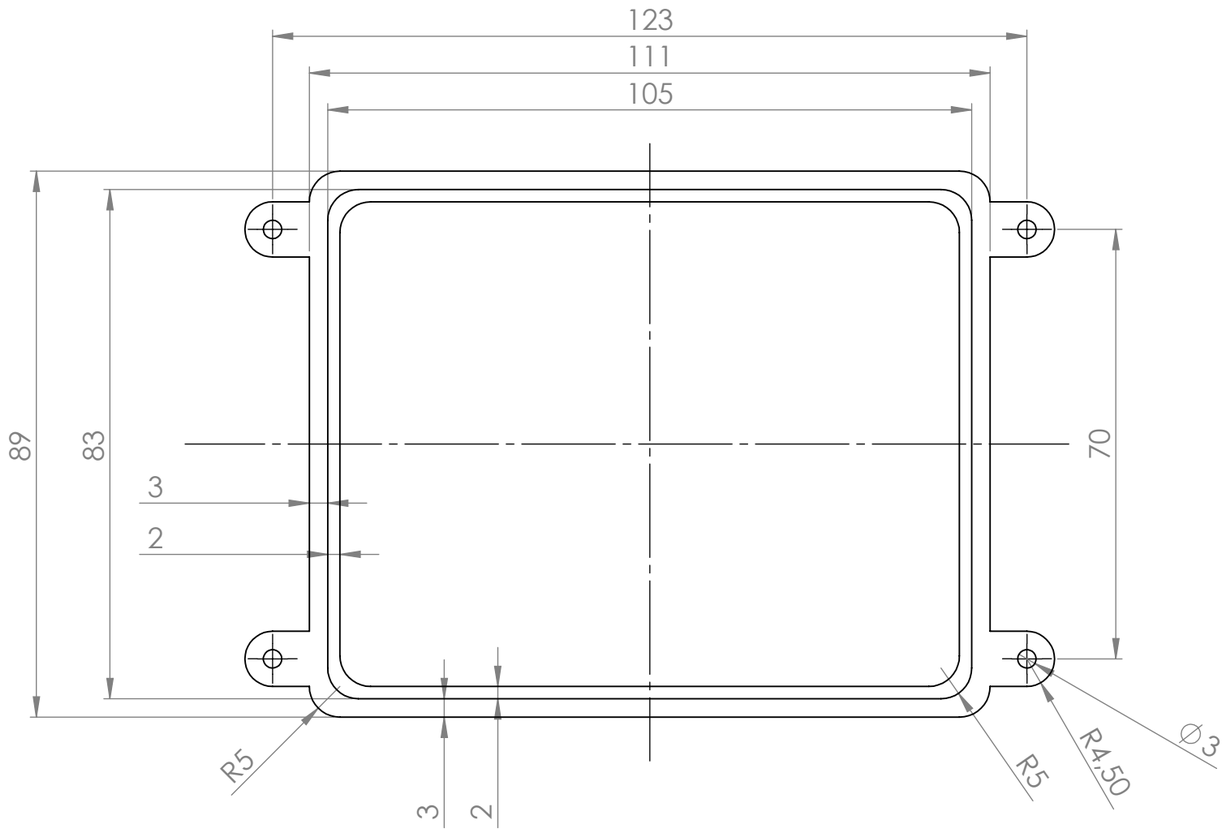
SECCIÓN A-A

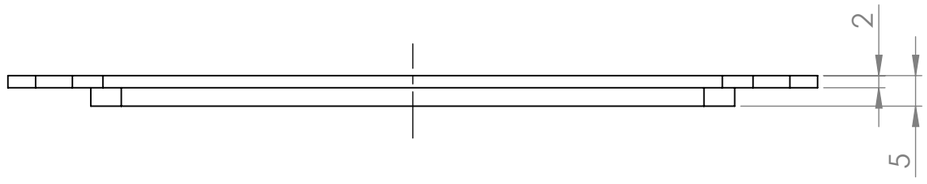


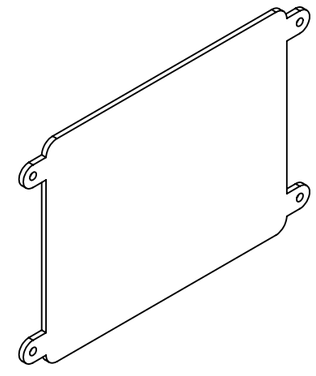


Nota: Radios de acuerdo de 2 mm salvo indicación

SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM	TOLERANCIAS	REBARBAR Y ROMPER ARISTAS VIVAS	NO CAMBIE LA ESCALA
NOMBRE Y APELLIDOS: Jose Javier Roldán	FIRMA	FECHA: 27/05/2022	TÍTULO: Carcasa prototipo
	MATERIAL: ABS	N.º DE DIBUJO: 1	A3
		ESCALA:1:1	HOJA 1 DE 1







SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM	TOLERANCIAS	REBARBAR Y ROMPER ARISTAS VIVAS	NO CAMBIE LA ESCALA
NOMBRE Y APELLIDOS: Jose Javier Roldán	FIRMA	FECHA: 27/05/2022	TÍTULO: Tapa carcasa
	MATERIAL: ABS	N.º DE DIBUJO: 2	A3
		ESCALA:1:1	HOJA 1 DE 1

3. CÓDIGO ARDUINO

```
/*
 *       Título: Sistema HUD para el visionado de la velocidad desde
el casco en motocicleta
 *       Autor: Jose Javier Roldan
 *       Director:Angel Maria Andueza
 *       Pamplona, 3 de junio de 2022
 *
 * */

//IMPORTANTE: CONFIGURAR ARDUINO IDE PARA EL ESP32

//-------COMUNICACIONES ESP32----------------------------

      //I2C--> GPIO-21(SDA); GPIO-22(SCL)
      //SPI--> GPIO-0(DC); GPIO-15(CS);GPIO-4(RESET); GPIO-18(SCL);
GPIO-23(SDA)
      //UART--> GPIO-17(RX GPS); GPIO-16(TX GPS);

//-------LIBRERIAS-------------------------

#include <WiFi.h>
#include <HardwareSerial.h>
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_SSD1306.h>
#include <TinyGPS++.h>
#include "DFRobot_MAX17043.h"

//-------OLED (SPI)-----------------------------

#define Alto_pantalla 32    // OLED
#define Ancho_pantalla 128 // OLED
#define OLED_DC     0
#define OLED_CS     15
#define OLED_RESET  4

Adafruit_SSD1306 display(Ancho_pantalla, Alto_pantalla,
  &SPI, OLED_DC, OLED_RESET, OLED_CS);

//-------GPS (UART)---------------------------------

#define rxPin 16
#define txPin 17
HardwareSerial neogps(1);
TinyGPSPlus gps;  //Crea los datos GPS analizando la libreria

//-------LECTOR BATERIA (I2C)-----------------------

#define ALR_PIN                2
#define PRINT_INTERVAL         2000
#define BAT_PERCENTAGE         32

DFRobot_MAX17043        gauge;

uint8_t       intFlag = 0;

//-------PROGRAMA-----------------------------

void interruptCallBack()  //La interrupcion del modulo lector de
bateria
{
```

```
  intFlag = 1;
}

void setup() {

   Wire.begin();         // inicializa bus I2C
   display.begin(SSD1306_SWITCHCAPVCC, 0x3C); // inicializa pantalla
con direccion 0x3C

   Wifi_BT_off(); //Llamamos a la funcion para apagar Wifi y Bluetooth
y que no consuman bateria

  //Alarma bateria
  pinMode(ALR_PIN, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(ALR_PIN), interruptCallBack,
FALLING);  //default alert is 32%

  //Comienza comunicacion serial con Neo6mGPS
  neogps.begin(9600,SERIAL_8N1,rxPin,txPin);

  display.clearDisplay();
  display.display();
  delay(2000);

}
//-------BUCLE PRINCIPAL-----------------------

void loop() {

  boolean newData = false;    //Variable booleana VERDADERO/FALSO
  for (unsigned long start = millis(); millis() - start <
1000;)  //Durante 1 segundo
  {
    while (neogps.available())
    {
      if (gps.encode(neogps.read()))  //Lee que hay nuevos datos
      {
        newData = true;
      }
    }
  }

  if(newData == true)
  {
    newData = false;
    print_speed();  //Llama a la función que te da los valores de
velocidad y batería
  }
  else
  {
    display.clearDisplay();
    display.setTextColor(WHITE);   // establece color al unico
disponible (pantalla monocromo)
    display.ssd1306_command(0xA0);  //Inverir horizontal para que el
reflejo se vea bien
    display.setCursor(0, 0);
    display.setTextSize(2);
    display.print("Sin datoss");
    display.display();
  }
```

```
}

//-------FUNCION PARA WIFI/BT OFF-------------

void Wifi_BT_off(){
    WiFi.mode(WIFI_OFF);     // Apaga Wifi
    btStop();                //Apaga bluetooth
}

//-------FUNCION PARA SACAR VALORES-------------

void print_speed()
{
  display.clearDisplay();
  display.ssd1306_command(0xA0);  //Inverir horizontal para que el
reflejo se vea bien
  display.setTextColor(WHITE);   // establece color al unico
disponible (pantalla monocromo)

  if (gps.location.isValid() == 1)  //Se ha conectado a GPS
  {

//-------VELOCIDAD-----------------------------

  int velocidad;

  display.setCursor (0,0);
  display.setTextSize(4);
  velocidad=gps.speed.kmph();
  display.print(velocidad);   //Muestra la velocidad en km/h obtenida
por GPS
  display.setCursor (50, 15);
  display.setTextSize(2);
  display.print(" km/h");

//-------BATERIA-------------------------------

  int nivel;

  display.setCursor (70, 0);
  display.setTextSize(2);
  nivel=gauge.readPercentage();
  display.print(nivel);
  display.println(" %");
   display.display();
  }

//-------SIN DATOS----------------------------
  else
  {
    display.clearDisplay();
    display.setTextColor(WHITE);   // establece color al unico
disponible (pantalla monocromo)
    display.ssd1306_command(0xA0);  //Inverir horizontal para que el
reflejo se vea bien
    display.setCursor(0, 0);
    display.setTextSize(2);
    display.print("Sin datos");
    display.display();
  }
}
```