



Universidad Nacional De Santiago Del Estero
Facultad de Ciencias Exactas y Tecnologías
“VEHICULO ACUATICO AUTONOMO”



PROYECTO FINAL DE INGENIERÍA

Para obtener el título de:
INGENIERO ELECTRÓNICO

Autores:
Bertero Cambrini Matias Exequiel
Vizcarra Savino Eduardo Orlando

Asesor:
Ing. Nicolas Mercado

Santiago del Estero – Argentina

2023

I. AGRADECIMIENTOS

II. DEDICATORIAS.

RESUMEN

En el marco de este trabajo final de grado, se ha llevado a cabo el desarrollo de un prototipo de Vehículo Acuático Autónomo (VAA) de propósitos generales.

El sistema diseñado se compone de dos elementos fundamentales que colaboran para el funcionamiento integral del VAA. En primer lugar, se encuentra una unidad terrestre de comando e información que establece comunicación con el VAA a través de señales de radiofrecuencia. Esta unidad terrestre desempeña un papel crítico en la supervisión y control del vehículo, así como en la recopilación de datos e información relevante.

Por otro lado, el corazón del VAA reside en una estructura flotante diseñada para operar de manera autónoma sobre el agua. Esta estructura incorpora tres compartimentos estancos, cada uno con un propósito específico. En el primero de estos compartimentos, se aloja el microcontrolador junto a otros módulos electrónicos, responsables de ejecutar una variedad de funciones necesarias para el funcionamiento automático del VAA.

En el segundo compartimento estanco, se encuentra el sistema mecánico esencial para llevar a cabo las diversas maniobras y movimientos requeridos por el VAA. Esta sección del prototipo garantiza la capacidad de navegación y maniobrabilidad en el entorno acuático, lo que es esencial para cumplir con los objetivos de propósitos generales del vehículo.

El tercer compartimento estanco facilita la comunicación bidireccional con la unidad terrestre, lo que permite el control remoto y la transmisión de datos vitales.

Un aspecto destacable de este proyecto es la elección de utilizar módulos de hardware libre y la programación del microcontrolador mediante software de código abierto. Esta decisión promueve la transparencia y la accesibilidad en el desarrollo tecnológico, facilitando la replicación y la colaboración en futuros proyectos similares.

Se implementa una comunicación crucial entre el vehículo y tierra firme mediante la tecnología LoRa, permitiendo una conexión robusta y de largo alcance que garantiza la transferencia efectiva de datos y comandos. Además, se emplea el protocolo MQTT para visualizar los datos en un dashboard desarrollado en Node-RED, proporcionando una interfaz de control e información accesible y eficiente para los operadores del VAA.

Esta integración de tecnologías de comunicación LoRa y MQTT agrega una capa adicional de versatilidad y capacidad de supervisión al proyecto, mejorando su funcionalidad y utilidad en diversas aplicaciones.

INDICE

1	Capítulo 1: INTRODUCCION	10
1.1	Antecedentes.....	11
1.2	Tipos de VAA	12
1.2.1	Vehículos Submarinos Autónomos (AUVs)	12
1.2.2	Vehículos Superficiales Autónomos (ASVs)	14
1.2.3	Vehículos Anfibios Autónomos	15
1.2.4	Boyas Autónomas.....	16
1.2.5	Vehículos Autónomos para Búsqueda y Rescate	17
1.2.6	Vehículos Autónomos de Limpieza.....	18
1.3	Planteamiento del Problema.....	19
2	Capítulo 2: HARDWARE UTILIZADO y PROTOCOLOS DE COMUNICACIÓN	23
2.1	Módulos Comunes en el Sistema de Tierra Firme y el Vehículo.....	27
2.1.1	Modulo ESP32.....	27
2.1.2	Protocolo SPI y modulo LoRa RFM95.....	29
2.1.2.1	Modulo LoRa y Protocolo LoRa	32
2.1.2.1.1	Protocolo LoRa.....	34
2.2	Módulos Adicionales del Vehículo	34
2.2.1	Protocolo I2C y modulo GY-271	35
2.2.1.1	Modulo GY-271	37
2.2.2	Protocolo UART, modulo GPS NEO 6M y Sensor JSN-SR04T	38
2.2.2.1	Modulo GPS NEO 6M.....	41
2.2.2.2	Sensor JSN-SR04T	43
2.2.3	Modulo L298N	45
2.2.4	Modulo Step-Down XL4005.....	47
2.2.5	Modulo Step-Up XL6009	49
3	Capítulo 3: SISTEMA MECANICO	51

3.1	Motor DC RS-455PA.....	51
3.2	Eje acoplado al Motor	51
3.3	Hélice.....	54
3.4	Servo Motor y Aleta como Timón	56
3.4.1	Aleta como Timón.....	58
4	Capítulo 4: SISTEMA DE ALIMENTACION AUTONOMA.....	62
4.1	Componentes de un sistema de alimentación fotovoltaico	62
4.2	Principio de Funcionamiento del Panel Solar	64
4.3	Tipos de Paneles Solares	65
4.3.1	Panel solar LTC Electronics.....	65
4.4	Reguladores de Carga.....	68
4.4.1	Regulador LTC Electronics SC1024A/U	70
4.5	Acumuladores de Energía	71
4.5.1	Batería de gel	71
5	Capítulo 5: PROTOCOLO MQTT y BROKER MQTT	73
5.1	Protocolo MQTT	73
5.2	Broker MQTT	74
6	Capítulo 6: SISTEMA DE TIERRA FIRME	76
6.1	Gateway	76
6.2	Software	77
6.3	Software del Servidor	78
7	Capítulo 7: SISTEMA DE NAVEGACION.....	86
7.1	Sistema de Posicionamiento Global	86
7.1.1	Componentes del Sistema GPS	86
7.1.2	Funcionamiento del GPS.....	87
7.1.3	Codigos NMEA.....	89
7.2	Formula de Haversine	90

7.3	Formula del Azimut.....	90
7.4	Software del sistema de navegación	91
8	Capítulo 8: CARACTERISTICAS ADICIONALES DEL VEHICULO	110
8.1	Consideraciones y funcionamiento	110
9	Capítulo 9: INTERFAZ DE VISUALIZACION DE DATOS	114
9.1	Node-RED y Dashboard	114
9.2	Programación realizada en Node-RED.....	117
10	Capítulo 10: PRUEBAS DE FUNCIONAMIENTO y AUTONOMIA	123
10.1	Sensor de Tensión y Corriente del Vehículo.....	123
10.2	Prueba 1: Funcionamiento del motor y aleta	131
10.3	Prueba 2: Funcionamiento del Tracker GPS	133
10.4	Prueba 3: Funcionamiento de ambos bloques en conjunto en un entorno acuático	135
10.5	Autonomía	140
11	Capítulo 11: CONCLUSIONES	142
12	ANEXO A Plano del Vehículo y Vistas 3D.....	147
13	ANEXO B Esquemas de Conexión	150

OBJETIVOS

Objetivos Generales

- Desarrollar un dispositivo acuático capaz de movilizarse a diferentes puntos automáticamente.
- Vincular a servidor externo mediante comunicación inalámbrica para coordinación de acciones.
- Desarrollar sistema de detección de obstáculos.
- Investigar y aplicar técnicas de control y coordinación de motores.
- Consolidar conocimientos adquiridos durante el cursado de la carrera de Ingeniería Electrónica.

Objetivos Específicos

- Dimensionar y seleccionar los componentes principales (placa de desarrollo, sensores, dispositivo de localización, drivers, etc) y complementarios (conductores, gabinetes, baterías, bms, etc).
- Realizar la programación del controlador para el movimiento del dispositivo.
- Diseñar un prototipo con comunicación inalámbrica para la transmisión de datos.
- Levantar y poner en funcionamiento servidor para la recopilación de información y procesamiento de datos.
- Obtener un prototipo de bajo costo y bajo consumo energético.
- Crear una interfaz gráfica para visualizar diferentes parámetros.

1 Capítulo 1: INTRODUCCION

En el contexto de la ingeniería electrónica y la innovación tecnológica aplicada a la gestión de recursos hídricos, los Vehículos Acuáticos Autónomos (VAA) se perfilan como herramientas de vanguardia con un enfoque específico en la exploración y monitoreo de reservorios de agua de gran extensión. Estos sistemas autónomos representan una evolución significativa en la manera en que abordamos la inspección y supervisión de infraestructuras hidráulicas, optimizando la eficiencia y precisión en la recopilación de datos cruciales para la operación y mantenimiento de estos entornos acuáticos.

En este contexto, el presente trabajo de grado tiene como objetivo central el diseño, desarrollo y optimización de un VAA adaptado para su implementación en reservorios de agua de grandes dimensiones. A través de la aplicación de principios de ingeniería electrónica, sistemas embebidos y tecnologías de navegación, se busca crear una plataforma autónoma capaz de navegar de manera segura y eficiente en estos entornos acuáticos específicos.

El trabajo abordará aspectos fundamentales como la selección y adaptación de sensores para la navegación, adaptados a las particularidades de estos espacios acuáticos, permitiendo la generación de rutas eficientes y la toma de decisiones autónomas en tiempo real.

A través de la integración de tecnologías de comunicación y sistemas de control remoto, se buscará habilitar la interacción con el VAA a distancia, lo que puede ser crucial en términos de accesibilidad y eficiencia en la recopilación de datos en entornos potencialmente peligrosos o de difícil acceso.

En última instancia, este proyecto pretende no solo desarrollar un Vehículo Acuático Autónomo específico para la exploración de reservorios, sino también sentar las bases para futuras investigaciones y desarrollos en la aplicación de tecnologías electrónicas en la gestión y conservación de recursos hídricos en contextos locales.

1.1 Antecedentes

En el año 2021 el alumno de la carrera de Ingeniería Electrónica José Ignacio Alba realizó su Tesis Final de Grado sobre “Sistema de monitoreo de calidad del agua aplicado al embalse Termas de Rio Hondo y ríos Tributarios”, en el cual desarrolló un prototipo de boyas para medir diferentes parámetros del embalse, estas boyas se colocarían en ciertas partes del embalse para tomar las muestras respectivas del agua.



Figura 1: Boya realizada por José Ignacio Alba por <https://noticias.mitelefe.com/>, 2020

A partir de este antecedente, surge la inspiración que da vida a este proyecto, el cual se centra en otorgar movilidad y capacidad de desplazamiento al dispositivo mencionado. La concepción y desarrollo de esta idea se detallarán con mayor profundidad en la sección donde se planteará el problema a resolver.

La Universidad de Loyola lanza en el año 2020, la realización de drones de agua para detectar contaminación en aguas superficiales y embalsadas de Andalucía.



Figura 2: Drones acuáticos usados en la investigación de la Universidad Loyola Andalucía, por <https://www.diariodesevilla.es/>, 2020

En el año 2019 estudiantes de Escuela de Educación Técnica nº1 “Dr. Pedro Radio” de Victoria Entre Ríos crearon un dron acuático para medir niveles de contaminación. El dispositivo se alimenta con energía solar y es autónomo. Tiene sensores para detectar el nivel de sal en el agua y cuenta con un potenciómetro que indica el pH.



Figura 3: Los alumnos de la Escuela Técnica 1 Pedro Radio de Victoria, Entre Ríos, junto al dron del Proyecto Kanneon, especializado en la toma de muestras de diversos valores del agua, por <https://www.lanacion.com.ar/>, 2019

1.2 Tipos de VAA

Un vehículo acuático autónomo (VAA) es un tipo de dispositivo robótico diseñado para operar en entornos acuáticos, como océanos, mares, ríos, lagos y otros cuerpos de agua. La característica principal que define a estos vehículos es su capacidad para operar de manera autónoma, lo que significa que pueden realizar tareas sin la necesidad de intervención humana constante. Esto se logra a través de sistemas avanzados de control, sensores y algoritmos de navegación. En la actualidad existen diferentes tipos de VAA, de acuerdo a las necesidades que se busquen, pueden ser submarinos, superficiales o anfibios. También podemos encontrar las boyas autónomas, vehículos autónomos para búsqueda y rescate, para investigación científica, para limpieza.

1.2.1 Vehículos Submarinos Autónomos (AUVs)

Estos son vehículos que se sumergen bajo la superficie del agua y se desplazan sin necesidad de intervención humana. Pueden llevar a cabo misiones de mapeo del fondo

marino, recolección de datos oceanográficos, inspecciones de infraestructuras subacuáticas y más.



Figura 4: Vehículo Submarino Autónomo “El Atún Rojo” por “<https://gdmissionsystems.com/>”, 2020.

La familia de productos Bluefin Robotics de General Dynamics Mission Systems’ consta de UUV y tecnologías relacionadas para clientes de defensa, comerciales y científicos en todo el mundo. Estos UUV, que varían de 9 a 21 pulgadas de diámetro, pueden cumplir una variedad de misiones de defensa, comerciales y científicas, que incluyen encuestas en tierra y en alta mar, inspección de infraestructura, contramedidas de minas, investigación científica y exploración y una variedad de inteligencia, vigilancia y reconocimiento (ISR) y aplicaciones de seguridad.

En el transcurso de 19 años, Bluefin Robotics diseñaría, desarrollaría y emplearía una flota en evolución de vehículos autónomos para una variedad de clientes comerciales y de defensa. General Dynamics Mission Systems adquirió Bluefin Robotics en 2016, agregando la compañía a su línea de negocios de Sistemas Marítimos y Estratégicos. Hoy, General Dynamics Mission Systems continúa avanzando los límites de las tecnologías UUV a través del desarrollo de las plataformas Bluefin Robotics, incluido EE. UU.

Ref: [<https://gdmissionsystems.com/articles/2023/04/19/featured-news-celebrating-25-years-of-bluefin-robotics>]

1.2.2 Vehículos Superficiales Autónomos (ASVs)

Estos vehículos operan en la superficie del agua y son utilizados para recopilar datos meteorológicos, oceanográficos y ambientales. También pueden ser utilizados en tareas de seguimiento y vigilancia.



Figura 5: Vehículo Superficial Autónomo “USV SB100 Pro de GPAsabots” por
<https://www.oceanografialitoral.com/>

Las dimensiones de los USV van desde un metro escaso (GPAsabots, por ejemplo) hasta embarcaciones de porte mediano capaces de realizar trabajos más exigentes. Se trata, probablemente, de la categoría más dinámica dentro del sector, con un gran número de fabricantes ofreciendo diversidad de aplicaciones.

La navegación sin cables se realiza mediante posicionamiento GPS y una ruta prefijada o bien mediante control remoto, vía radio, normalmente. Aunque disponen de habilidad de navegación autónoma, siempre es posible tomar el control por parte del operador, en forma remota, vía radio

La arquitectura de los USV se basa en los siguientes sistemas:

- Sensores para la percepción de su entorno, que forman el sistema de navegación
- Sistema de guiado, para poder seguir una trayectoria
- Módulo de gestión y control, formado por el sistema de navegación y el de guiado

- Sistema de planificación dinámica, que sería el más importante, ya que dota al USV de autonomía para responder ante situaciones cambiantes del entorno, como puede ser la evasión de obstáculos, sin intervención humana.

El mercado nos ofrece gran cantidad de modelos, dependiendo de nuestras necesidades, tanto para aplicaciones comerciales y científicas, como militares.

Ref:[<https://www.oceanografialitoral.com/pregunta-frecuente/vehiculos-asv-auv-rov-y-otros/>]

1.2.3 Vehículos Anfibios Autónomos

Estos vehículos son capaces de operar tanto en el agua como en tierra, lo que los hace adecuados para tareas que requieren desplazamiento entre diferentes medios.



Figura 6: Vehículo Anfibio Autónomo “SHERP” por “<https://www.larazon.es/>”, 2023

La ONU usará este vehículo autónomo no tripulado y con IA en zonas de conflicto desde 2024.

- Se denomina SHERP, y lleva siendo usado desde 2012, pero hasta ahora siempre con conductor.
- La IA permitirá combinar datos de diversas fuentes para que el vehículo pueda ser totalmente autónomo.
- Es un vehículo anfibio y curiosamente la sede de la compañía que los fabrica está en Rumanía.

Ref:[https://www.niusdiario.es/economia/motor/20230819/onu-vehiculo-autonomo-tripulado-ia-conflicto-2024_18_010270290.html]

1.2.4 Boyas Autónomas

Estas boyas son utilizadas para la recopilación de datos oceanográficos, meteorológicos y ambientales. Pueden flotar en la superficie o ser ancladas a diferentes profundidades para medir parámetros específicos.



Figura 7: Boya Autónoma por <https://www.ecoticias.com/>, 2011

Instalan en Argentina la primera boyta autónoma de monitoreo de lagunas.

La boyta fue diseñada y construida íntegramente en el Laboratorio de Diseño de Instrumental del IADO y se logró a partir de una adaptación de las Estaciones de Monitoreo Ambiental Costero (EMAC) que el IADO ha venido desarrollando e incorporando a su programa de monitoreo desde hace años. De esta forma se podrán obtener datos relacionados con la velocidad y dirección del viento, la presión, la humedad y temperatura del aire; la conductividad y temperatura del agua y la concentración de sedimentos en suspensión y nivel del agua.

La información generada por la misma es actualizada cada media hora y puede consultarse a través de la página web <http://emac.criba.edu.ar>. No obstante, los sensores muestrean todos los parámetros ambientales cada 5 minutos.

Ref:[https://www.ecoticias.com/naturaleza/46027_instalan-en-argentina-la-primeraboya-autonoma-de-monitoreo-de-lagunas]

1.2.5 Vehículos Autónomos para Búsqueda y Rescate

Algunos vehículos autónomos están diseñados para misiones de búsqueda y rescate en entornos acuáticos. Pueden ser utilizados para localizar y rescatar personas en peligro en el agua.



Figura 8: Vehículo Autónomo para Búsqueda y Rescate “Skua” por <https://www.opisantacruz.com.ar/>, 2023

Muy lejos del territorio verde típico de su Entre Ríos natal, el vehículo de exploración Skua se desplaza en suelo antártico, donde el paisaje tiene pocas diferencias y la orientación es compleja. Con cámaras térmicas y ópticas de largo alcance y de cercanía de 360° para la navegación, sensores de calidad del aire, temperatura y derrames, el sistema no tripulado puede transmitir toda la información a través de antenas a más de 30 kilómetros, dar apoyo científico y de búsqueda y rescate.

Ahora el Skua está en la sede entrerriana de la empresa, en donde le realizan nuevas pruebas y actualizaciones de software, en base a la experiencia que ya tuvo en la Antártida, para luego regresar en el último trimestre de 2023.

Ref:[<https://www.opisantacruz.com.ar/2023/06/10/van-por-aire-tierra-y-agua-asi-son-los-vehiculos-autonomos-de-exploracion-y-vigilancia-disenados-en-la-argentina/>]

1.2.6 Vehículos Autónomos de Limpieza

Algunos diseños de vehículos acuáticos autónomos se centran en la limpieza de basura y contaminantes en cuerpos de agua.

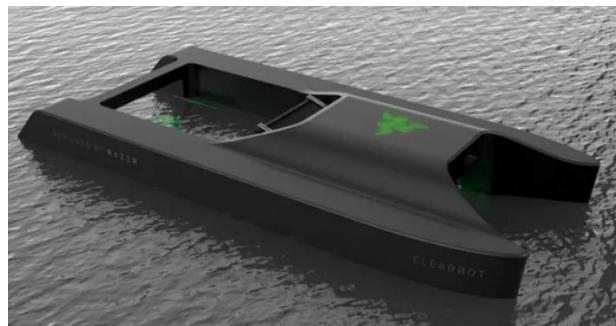


Figura 9: Vehículo Autónomo de Limpieza “ClearBot” por <https://www.forbesargentina.com/>, 2022

Acuerdo entre Razer y ClearBot para crear un robot autónomo que limpie los desechos marinos.

ClearBot, el startup con origen y sede en Hong Kong, ha estado inmersa en la creación de nuevas y futuras embarcaciones autónomas y motricidad eléctrica gracias a la propia energía solar. Estos navíos tienen un propósito muy claro: limpiar y recolectar los diferentes desechos vertidos a nuestros océanos.

¿Y cómo distingue un desecho de algo que no lo es? Los chicos de ClearBot han realizado un intensivo estudio y desarrollo de un sistema que unifica la visión exterior a través de cámaras de alta sensibilidad y la propia inteligencia artificial. Este trabajo mano a mano es capaz de identificar diferentes tipos de desechos, principalmente plásticos, y recopila toda la información sobre ellos en favor de la salvaguarda de la vida submarina.

Puede recolectar hasta 250 kg de basura marítima en una única salida, es decir, llegar a recoger hasta 1 tonelada al día, gracias principalmente a su sistema de descarga rápida de desechos y a la extracción de energía fotovoltaica. Su potente sistema de inteligencia artificial es capaz de detectar residuos plásticos en un radio de 2 metros sobre él mismo.

Ref:[https://www.hibridosyelectricos.com/coches/acuerdo-razer-clearbot-crear-robot-autonomo-que-limpie-desechos-marinos_45907_102.html]

1.3 Planteamiento del Problema

A raíz del antecedente antes mencionado, nos planteamos la pregunta fundamental: ¿Es posible llevar a cabo estas tareas sin depender de la intervención humana? La respuesta es afirmativa. De esta premisa nace la concepción de este proyecto, cuyo propósito central es proporcionar movilidad al dispositivo mencionado previamente.

La movilidad que se busca otorgar a este proyecto no solo representa una mejora significativa desde una perspectiva económica y operativa, sino que también potencia la eficiencia del proceso de control ambiental en los espacios designados. El prototipo desarrollado cuenta con un sistema de movimiento autónomo, el cual incluye una combinación de sensores, motores, controladores y un módulo GPS para la navegación y la determinación precisa de la posición.

Este diseño se ha concebido con la finalidad de detectar obstáculos y evitar posibles colisiones durante su operación. Además, el sistema permite la comunicación con un servidor, lo que posibilita la planificación de rutas óptimas y el almacenamiento de datos recopilados por los diversos sensores integrados en el dispositivo. En conjunto, estos componentes contribuyen a una solución integral que busca automatizar y mejorar significativamente las operaciones de control del prototipo.

El proyecto se divide en dos bloques, que serán denominados de la siguiente manera:

El “Sistema de Tierra Firme” se compone de manera central de un servidor denominado Gateway y una plataforma de visualización de datos. Aunque los detalles pormenorizados de estos elementos se abordarán con mayor profundidad en capítulos subsiguientes de este proyecto, es importante destacar que se proporcionarán imágenes ilustrativas de los elementos mencionados:



Figura 10: Gateway encargado de enviar y recibir datos del vehículo, Edición Propia

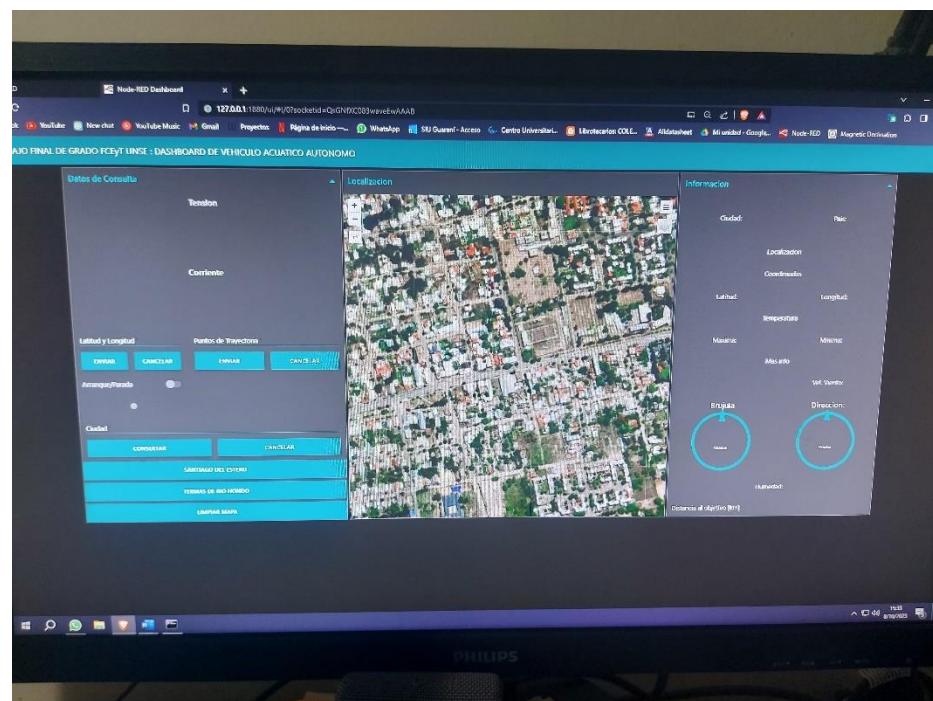


Figura 11: Plataforma de visualización de datos, Edición Propia



Figura 12: Sistema Tierra Firme completo, Edición Propia

El segundo bloque se refiere al "Vehículo", el cual experimentó un proceso de rediseño. Esto fue necesario porque la estructura empleada en el proyecto anterior, en el cual se basa este proyecto actual, no presentaba la forma hidrodinámica requerida para un movimiento fluido sobre el agua. Su diseño anterior generaba una resistencia significativa debido a su forma, lo que motivó la decisión de crear una nueva estructura que cumpliera con las condiciones necesarias para un desplazamiento adecuado. Cabe destacar que esta nueva estructura ha sido construida de manera económica y fabricada de forma casera.

El vehículo tiene la capacidad de desplazarse de forma autónoma, sin requerir intervención humana. Además, puede establecer comunicación con el sistema de tierra firme para llevar a cabo diversas acciones. En las siguientes imágenes, se presentará una

representación visual del vehículo, proporcionando una visión completa de su diseño actualizado.



Figura 13: Vehículo realizado para el proyecto, Edición Propia

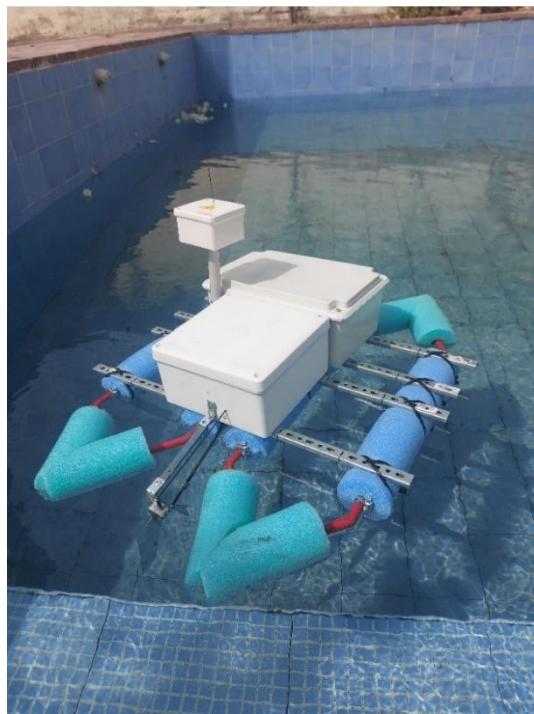


Figura 14: Vehículo realizado para el proyecto, Edición Propia

En el anexo A se podrá observar imágenes del diseño en 3D y el plano del mismo.

2 Capítulo 2: HARDWARE UTILIZADO y PROTOCOLOS DE COMUNICACIÓN

En este capítulo, se analizará en detalle el hardware utilizado tanto en el sistema de tierra firme como en el vehículo. Se describirán cada uno de los módulos empleados en cada caso y los protocolos de comunicación necesarios para garantizar su correcto funcionamiento. Este análisis detallado permitirá comprender cómo cada componente contribuye al funcionamiento integral de los dos bloques mencionados anteriormente, asegurando su operación de manera eficiente y coordinada.

El hardware se compone de dos partes fundamentales. En tierra firme, se dispone de una placa ESP32 que establece comunicación mediante el protocolo MQTT (capítulo 5) con un servidor (Gateway) y, además, utiliza radiofrecuencia para interactuar con el vehículo. Por otro lado, el vehículo cuenta con otra placa ESP32 que desempeña el papel de dispositivo central, encargado de coordinar y controlar los módulos que se verán en este capítulo. Todos estos componentes están interconectados de manera cuidadosa para asegurar un funcionamiento integral y preciso del sistema.

Tanto para el sistema de tierra firme como para el vehículo, se ha implementado un esquema de conexión modular utilizando el siguiente adaptador diseñado específicamente para el ESP32:



Figura 15: Adaptador de ESP32 utilizado en ambos bloques, Edición Propia

Este adaptador facilitará la conexión de los módulos utilizados en cada caso, ofreciendo la ventaja de una sustitución sencilla en caso de fallos en alguno de los componentes. A continuación, se detallan imágenes de las conexiones en ambos bloques:

Bloque Tierra Firme:

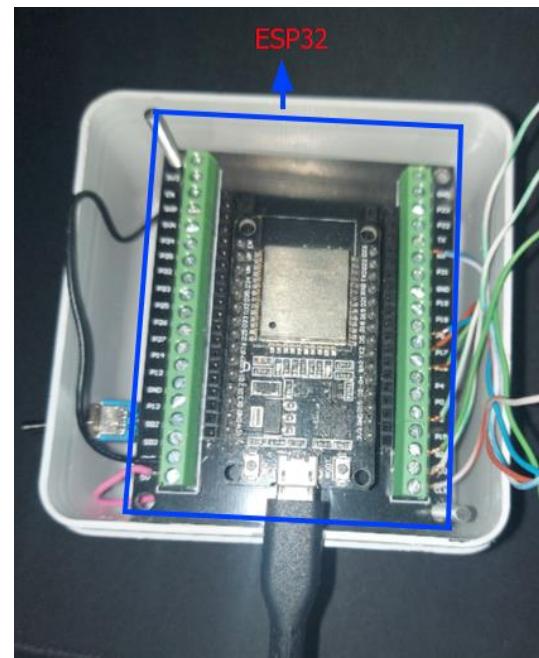


Figura 16: Compartimento estanco de tierra firme con ESP32, Edición Propia

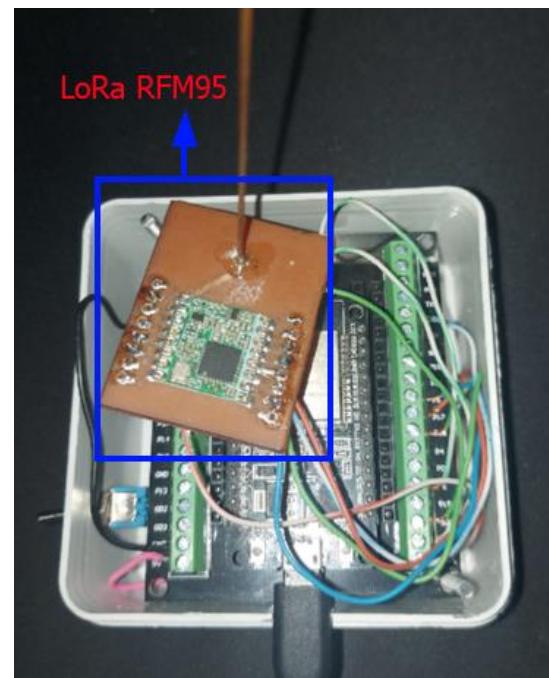


Figura 17: Modulo LoRa del sistema de tierra firme, Edición Propria

Bloque Vehículo:

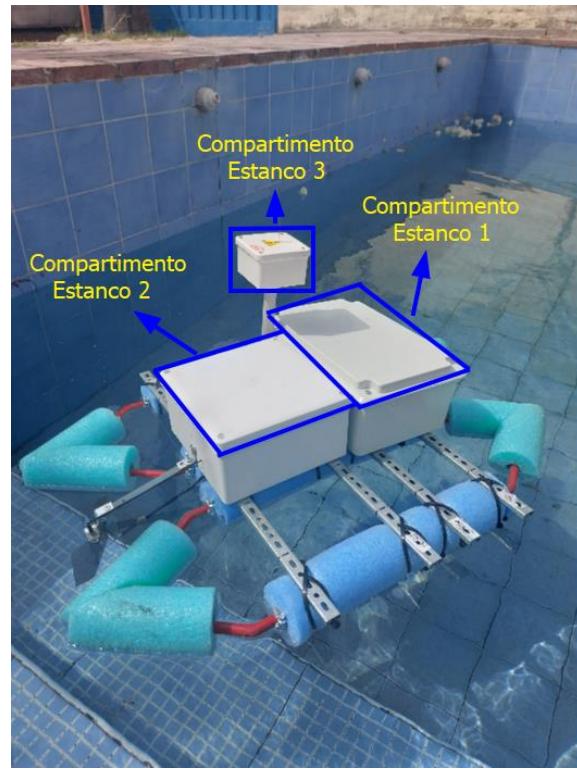


Figura 18: Compartimentos Estancos del Vehículo, Edición Propia

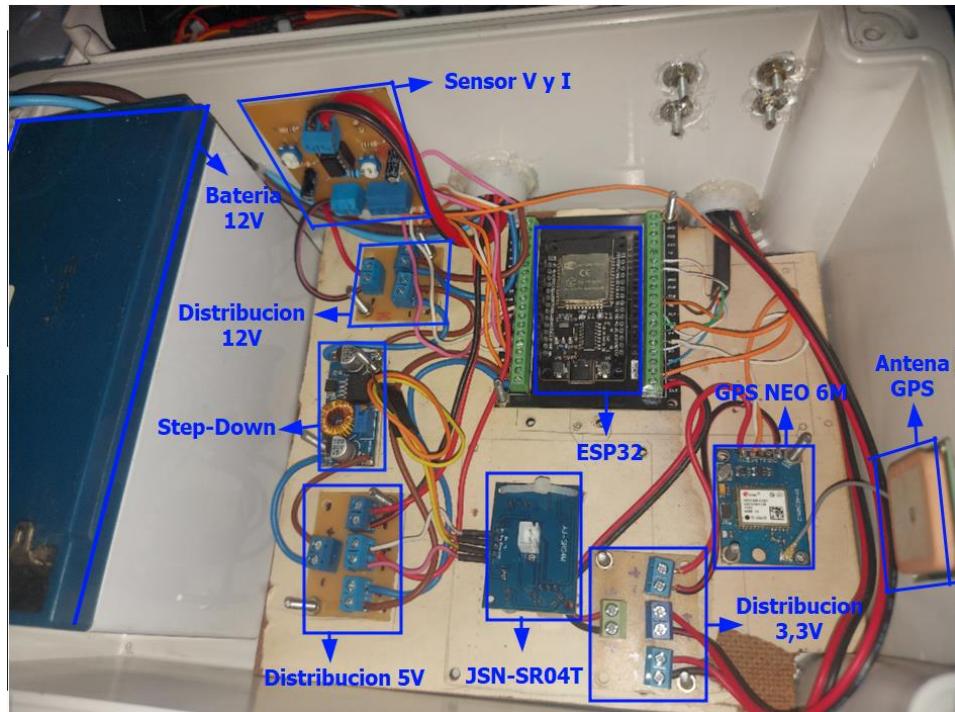


Figura 19:Conexión modular del Vehículo compartimento estanco 1, Edición Propia

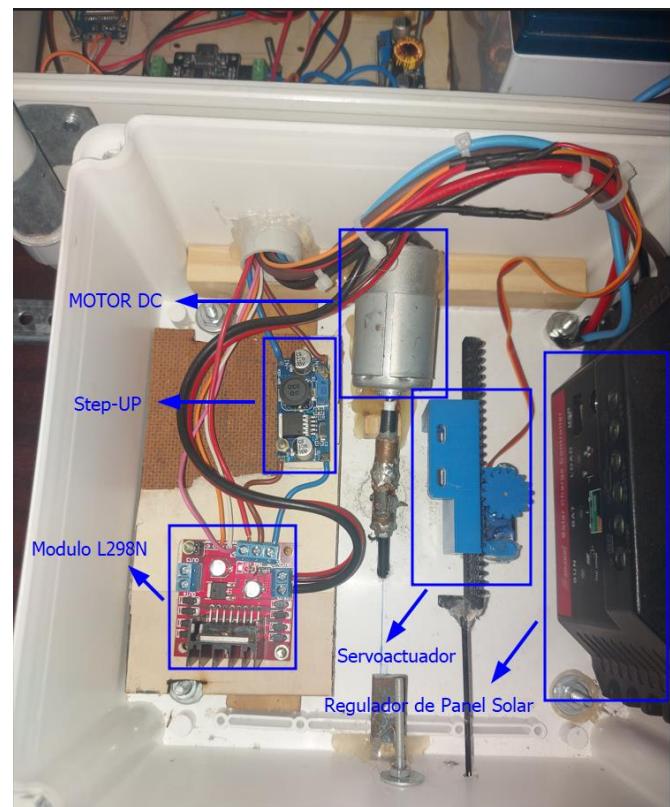


Figura 20:Conexión compartimento estanco 2, Edición Propia

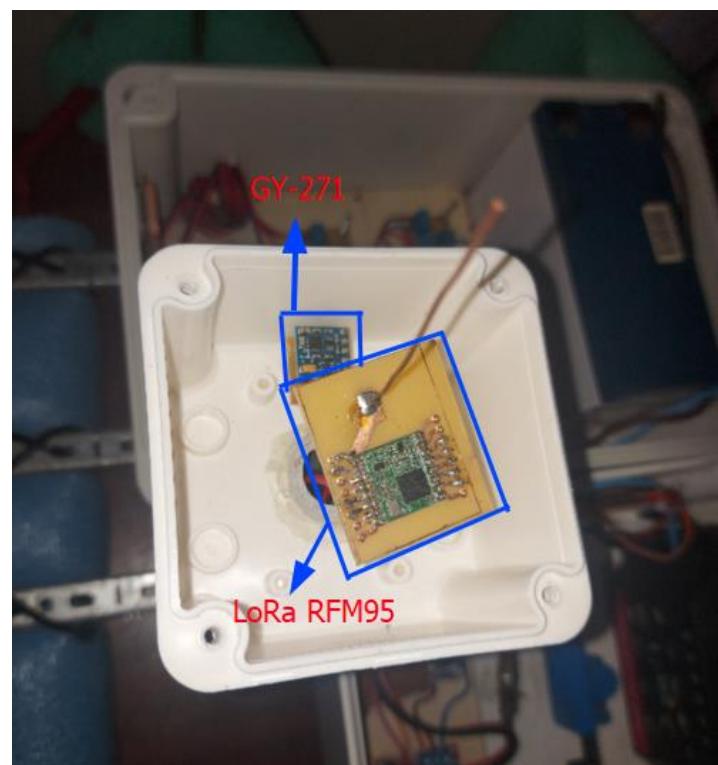


Figura 21:Conexión de Compartimento estanco 3, Edición Propia

Todos los módulos señalados en las imágenes anteriores serán explicados en detalle a continuación.

2.1 Módulos Comunes en el Sistema de Tierra Firme y el Vehículo

Los módulos comunes empleados para ambos bloques fueron los siguientes:

- Modulo ESP32
- Modulo LoRa RFM95

2.1.1 Modulo ESP32

“Creado por Espressif Systems, ESP32 es un sistema de bajo consumo y bajo costo en un chip SoC (System On Chip) con Wi-Fi y modo dual con Bluetooth. Cuenta con un microprocesador Tensilica Xtensa LX6 de doble núcleo o de un solo núcleo con una frecuencia de reloj de hasta 240MHz.”

“ESP32 posee un alto nivel de integración, con switch de antena, balun para RF, amplificador de potencia, amplificador de recepción con bajo nivel de ruido, filtros y módulos de administración de energía, totalmente integrados dentro del mismo chip. Diseñado para dispositivos móviles; tanto en las aplicaciones de electrónica, y las de IoT (Internet de las cosas), ESP32 logran un consumo de energía ultra bajo a través de funciones de ahorro de energía que incluyen sincronización de reloj y múltiples modos de operaciones.”

Ref:[https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf]

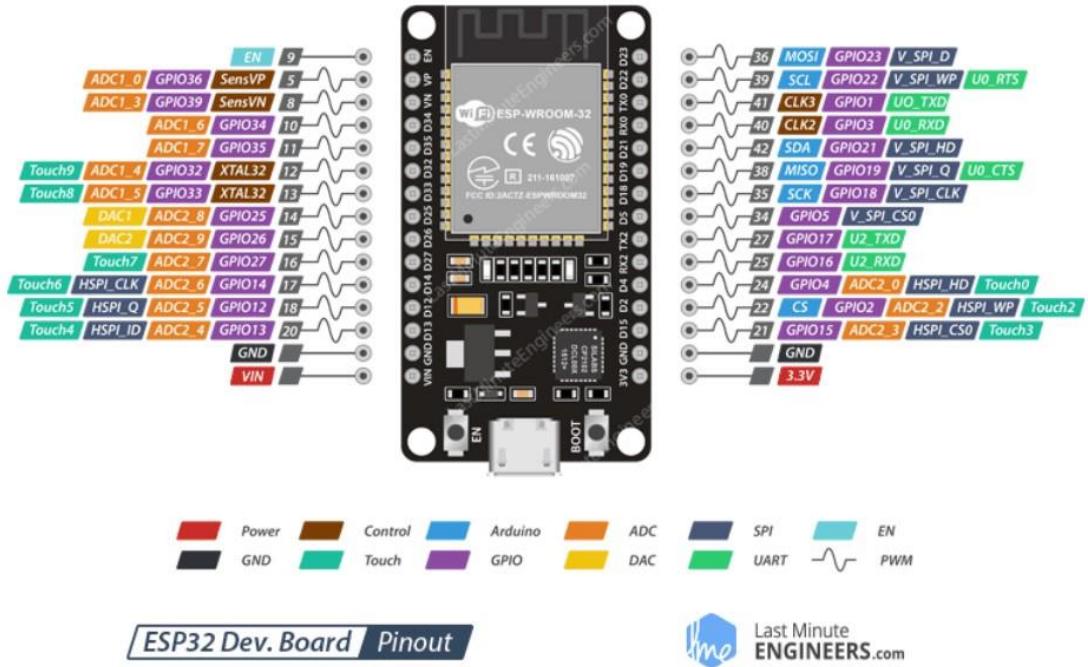


Figura 22: Pin Out de Placa ESP32 Dev Module por <https://descubrearduino.com/>, 2015.

Características principales:

- *Procesador principal: Tensilica Xtensa LX6 de 32 bits.*
- *Wi-Fi: 802.11 b / g / n / e / i (802.11n @ 2.4 GHz hasta 150 Mbit / s).*
- *Bluetooth: v4.2 BR / EDR y Bluetooth Low Energy (BLE).*
- *Frecuencia de Clock: Programable, hasta 240MHz.*
- *Rendimiento: hasta 600DMIPS.*
- *ROM: 448KB, para arranque y funciones básicas. RAM: 520KiB, para datos e instrucciones.*
- *30 pines GPIO (Pines entrada/salida de propósitos generales).*
- *15 canales ADC de 12 bits con rangos seleccionables de 0-1V, 0-1.4V, 0-2V o 0-4V.*
- *2 interfaces UART con control de flujo.*
- *25 pines PWM configurables.*
- *Dos ADC de 8 bits.*

- *Tres interfaces SPI, una I2C y dos I2S para sonido.*
- *9 GPIO con detección táctil capacitiva.*

Ref:https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

Tanto en el sistema de tierra firme como en el vehículo, se optó por utilizar el microcontrolador ESP32 como unidad central de procesamiento. Esta elección se basó en varias consideraciones. En primer lugar, se contaba con disponibilidad de estos microcontroladores, lo que simplificó la implementación del proyecto. Además, el ESP32 se destacó como una elección idónea debido a su capacidad de procesamiento con dos núcleos, lo que permite una distribución eficiente del software, que se detallará en capítulos posteriores.

Otra característica destacada es su conectividad WiFi, que posibilita la comunicación del sistema de tierra firme con la plataforma de visualización de datos. Los detalles de esta plataforma se presentarán con mayor profundidad en capítulos subsiguientes.

2.1.2 Protocolo SPI y modulo LoRa RFM95

SPI, es una interfaz de comunicación en serie de muy baja potencia de cuatro cables diseñado para IC controladores y periféricos para comunicarse entre sí.

El bus SPI es un bus de dúplex completo, que permite la comunicación fluya hacia y desde el dispositivo maestro simultáneamente a velocidades de hasta 10 Mbps. La operación de alta velocidad de SPI generalmente limita que sea utilizado para la comunicación entre componentes en PCB separadas a distancias largas debido al aumento en la capacitancia. La capacitancia PCB también puede limitar la longitud de las líneas de comunicación SPI.

Mientras SPI es un protocolo establecido. Puede conducir a problemas de compatibilidad. Las implementaciones SPI siempre deben ser revisadas entre controladores maestros y periféricos esclavos para asegurar que la combinación no tendrá ningún problema inesperado de comunicación que impactará en el desarrollo de un producto.

Estructura general del protocolo SPI:

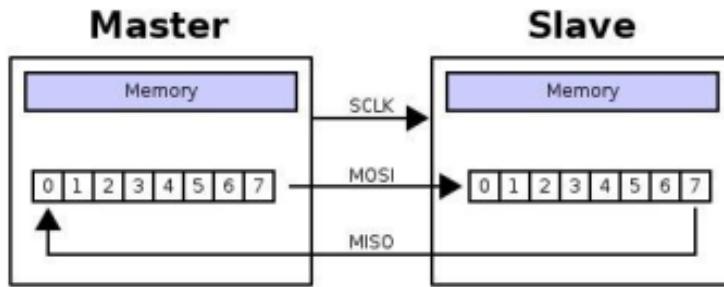


Figura 23: Estructura General de protocolo SPI por <https://panamahitek.com/>, 2014

Dentro de este protocolo se define un maestro que será aquel dispositivo encargado de transmitir información a sus esclavos. Los esclavos serán aquellos dispositivos que se encarguen de recibir y enviar información al maestro. El maestro también puede recibir información de sus esclavos. Para que este proceso se haga realidad es necesaria la existencia de dos registros de desplazamiento, uno para el maestro y uno para el esclavo respectivamente. Los registros de desplazamiento se encargan de almacenar los bits de manera paralela para realizar una conversión paralela a serial para la transmisión de información.

Existen cuatro líneas lógicas encargadas de realizar todo el proceso:

- MOSI (Master Out Slave In): Línea utilizada para llevar los bits que provienen del maestro hacia el esclavo.
- MISO (Master In Slave Out): Línea utilizada para llevar los bits que provienen del esclavo hacia el maestro.
- CLK (Clock): Línea proveniente del maestro encarga de enviar la señal de reloj para sincronizar los dispositivos.
- SS (Slave Select): Línea encargada de seleccionar y a su vez, habilitar un esclavo.

Algunas ventajas y desventajas a tener en cuenta de este protocolo son:

SPI (Serial Peripheral Interface):

Ventajas:

- Velocidad: SPI puede ser más rápido que I2C, ya que no está limitado por la sincronización del reloj como en I2C.
- Modo Maestro-Múltiples Esclavos: El modo maestro en SPI permite una comunicación rápida y eficiente con múltiples dispositivos esclavos.
- Dúplex completo: SPI admite comunicación full-duplex, lo que significa que puede transmitir y recibir datos simultáneamente.
- Flexibilidad de configuración: SPI permite configuraciones flexibles, como diferentes modos de reloj y polaridad de señal.
- Carga de datos simple: La transferencia de datos en SPI es más simple que en I2C, ya que no involucra direcciones de dispositivo.

Desventajas:

- Más líneas requeridas: SPI requiere más líneas de datos (al menos 3) en comparación con I2C y UART.
- Conexiones complejas: Con múltiples dispositivos en un bus SPI, la gestión de las señales de selección (chip select) puede volverse complicada.
- No estándar: A diferencia de I2C y UART, no existe un estándar universal para las conexiones y configuraciones de SPI, lo que puede generar variabilidad.

El ESP32 dispone de tres interfaces SPI, de las cuales se emplea una en ambos componentes, tanto en el sistema de tierra firme como en el vehículo. Esta interfaz se utiliza para establecer la comunicación adecuada con el módulo LoRa RFM95, que habilita tanto la transmisión como la recepción de datos entre los dos bloques fundamentales del prototipo.

2.1.2.1 Modulo LoRa y Protocolo LoRa

LoRa (*Long Range*) emplea espectros de frecuencia de uso público sin licencia ISM (como los utilizados por Bluetooth y WiFi) de banda estrecha (entre 300 y 3.400 hertzios) siendo esta una de sus principales ventajas, utiliza una modulación de espectro ensanchado en la banda menor al GHz lo que permite rangos largos de cobertura mayores a 10 kilómetros, tiene alta capacidad de nodos (hasta 1 millón de nodos), es una comunicación robusta y ofrece capacidad de localización. LoRa es una tecnología inalámbrica desarrollada para crear las redes de área amplia con baja potencia, requeridas para aplicaciones de maquina a máquina (M2M) e Internet de las cosas (IoT). La tecnología ofrece una mezcla muy convincente de largo alcance, bajo consumo de energía y transmisión”.

[Ref: <https://www.semtech.com/lora/what-is-lora>].

Modulo LoRa a utilizar

Modulo RFM95

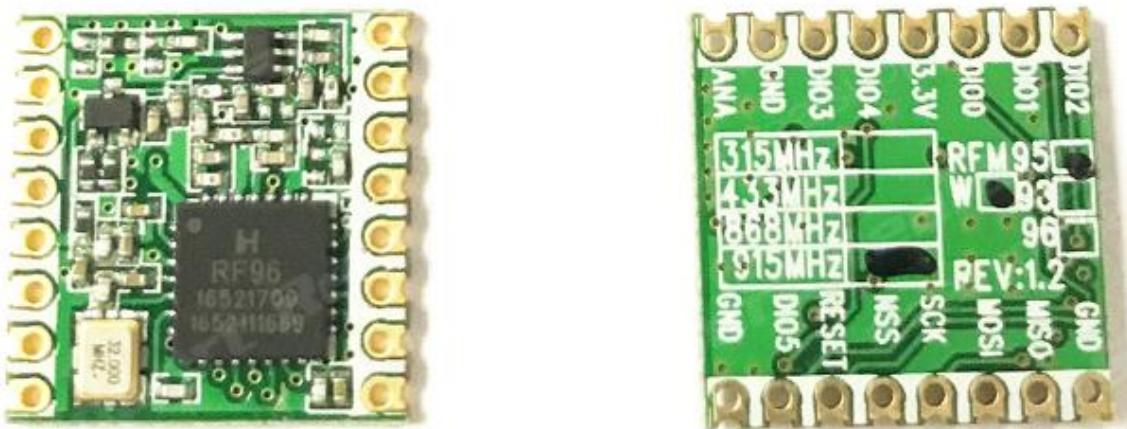


Figura 24:M Modulo LoRa RFM95 por <https://www.aprendiendoarduino.com/>, 2018.

Características del Módulo:

- Es apto para un espectro de frecuencia ISM de 915.0 MHz.
- Compatible con la plataforma ESP32.

- *Voltaje de funcionamiento 3.3V Recomendado, MIN 1.8V – MAX 3.7V, Se puede conectar directamente a microcontroladores que trabajan a 3.3v, para otras plataformas a 5v es necesario utilizar convertidores de nivel lógico.*
- *La configuración y comunicación del módulo se realiza vía Bus SPI 4 hilos, implementada en técnicamente todos los microcontroladores.*
- *Tiene 6 Gpio configurables por software, generalmente son interrupciones ligadas al funcionamiento del RFM95.*
- *Aunque se puede configurar como modem LoRa TM también permite configurarse como modem FSK/OOK y los estándares GFSK, MSK y GMSK.*

[Ref: http://pdacontroles.com/wpcontent/uploads/2018/03/RFM95_96_97_98W.pdf]

Se ha seleccionado el módulo RFM95 para facilitar la comunicación entre el vehículo y el sistema de tierra firme debido a sus características sobresalientes. Este módulo opera mediante radiofrecuencia, lo que posibilita el establecimiento de comunicaciones a larga distancia con un ancho de banda reducido, una característica crucial para lograr una cobertura eficiente en este proyecto. Además, el módulo RFM95 es reconocido por su bajo consumo de energía, lo cual resulta esencial para el funcionamiento sostenible y prolongado del prototipo.

En resumen, el módulo RFM95 habilita la comunicación bidireccional entre el vehículo y la estación de tierra firme. Esto permite tanto la recepción como el envío de mensajes de manera eficaz y eficiente, respaldando así la operación fluida del proyecto. Entre sus funciones clave, se incluye la transmisión de puntos de trayectoria desde tierra firme al vehículo, junto con las coordenadas asociadas a cada uno de estos puntos. Además, el módulo RFM95 facilita la instrucción de arranque y parada del vehículo, detalles que se explorarán con mayor profundidad en el Capítulo 6.

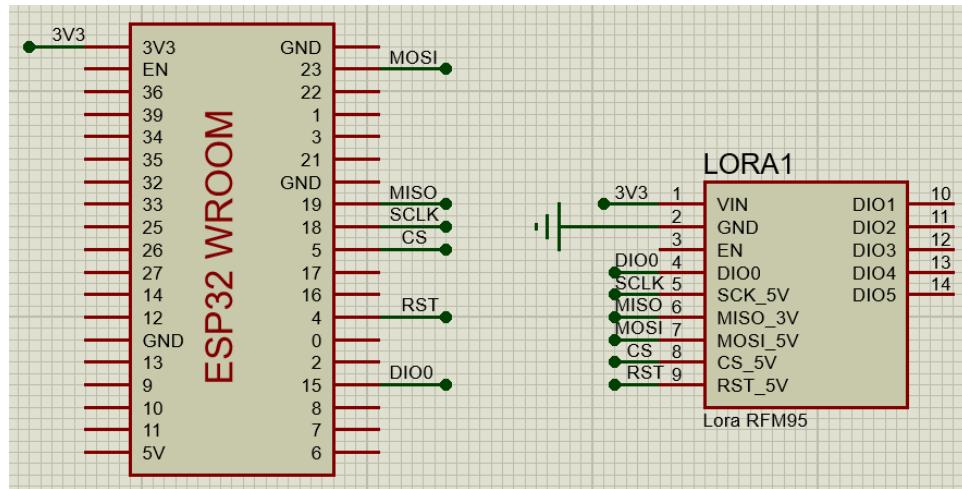


Figura 25: Esquema de conexión de módulo LoRa con ESP32, Edición propia

2.1.2.1.1 Protocolo LoRa

El protocolo de comunicación LoRa permite la conexión de un gran número de nodos (vehículo) a un nodo principal, que en nuestro caso será el servidor en tierra firme, en una topología de red tipo estrella. Esta comunicación de radio frecuencia es a 915Mhz que es la adecuada para Argentina según lo detallado en la Resolución del Ministerio de Modernización N° 581/18 donde se especifican cuales con las bandas que no requieren autorización para su uso dentro del territorio.

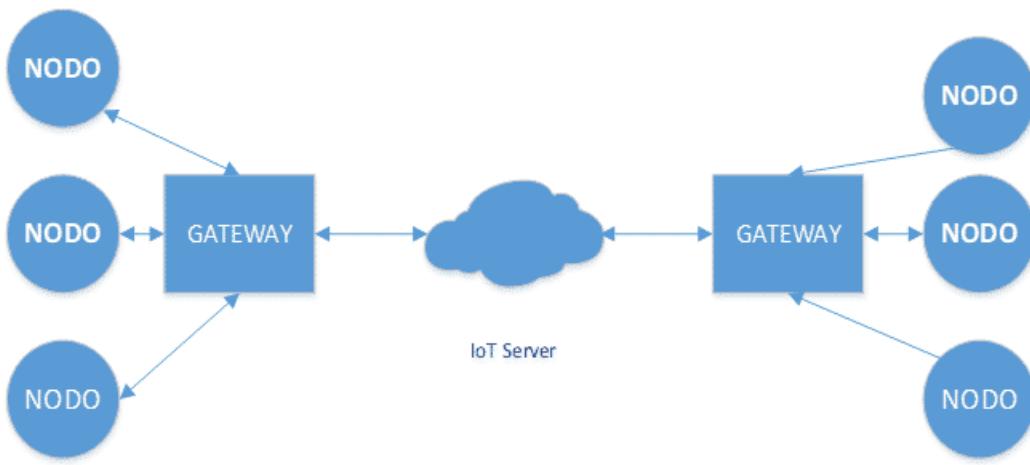


Figura 26: Tipología Estrella de una red LoRa por <https://elbosquedesilicio.es/>

2.2 Módulos Adicionales del Vehículo

Además de los dos módulos mencionados anteriormente, para asegurar el correcto funcionamiento del vehículo, es necesario contar con los siguientes módulos adicionales:

- Módulo HMC5883L (GY-271)
- Módulo GPS NEO-6M
- Sensor JSN-SR04T
- Módulo L298N
- Módulo Step-Down
- Módulo Step-Up

A continuación, se detallan cada uno de los módulos mencionados junto con su protocolo de comunicación en los casos que sean necesarios:

2.2.1 Protocolo I2C y modulo GY-271

I2C es un protocolo de comunicación en serie estándar oficial que sólo requiere dos líneas de señal, fue diseñado para la comunicación entre los chips en un PCB.

I2C fue diseñado originalmente para la comunicación a 100 kbps, pero los modos de transmisión de datos más rápida se han desarrollado en los últimos años para alcanzar velocidades de hasta 3.4Mbps. El protocolo I2C se ha establecido como un estándar oficial, que prevé una buena compatibilidad entre las implementaciones I2C y buena compatibilidad con versiones anteriores.

El bus I2C, es un estándar que facilita la comunicación entre microcontroladores, memorias y otros dispositivos con cierto nivel de "inteligencia", sólo requiere de dos líneas de señal y un común o masa. Fue diseñado a este efecto por Philips y permite el intercambio de información entre muchos dispositivos a una velocidad aceptable.

La metodología de comunicación de datos del bus I2C es en serie y sincrónica. Una de las señales del bus marca el tiempo (pulsos de reloj) y la otra se utiliza para intercambiar datos. Descripción de las señales:

- SCL (System Clock) es la línea de los pulsos de reloj que sincronizan el sistema.
- SDA (System Data) es la línea por la que se mueven los datos entre los dispositivos.

- GND (Masa) común de la interconexión entre todos los dispositivos "enganchados" al bus.

Las líneas SDA y SCL son del tipo drenaje abierto, es decir, un estado similar al de colector abierto, pero asociadas a un transistor de efecto de campo (o FET). Se deben polarizar en estado alto (conectando a la alimentación por medio de resistores "pull-up") lo que define una estructura de bus que permite conectar en paralelo múltiples entradas y salidas.

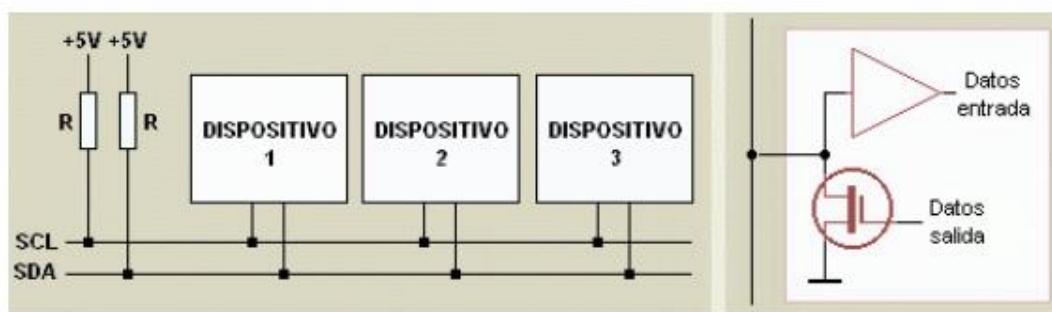


Figura 27: Protocolo I2C conexión en paralelo por <https://robots-argentina.com.ar/>, 2019.

Algunas ventajas y desventajas de este protocolo son:

Ventajas:

- Multi-dispositivo: I2C permite la conexión de múltiples dispositivos en el mismo bus, utilizando solo dos líneas de datos (SDA) y reloj (SCL).
- Direcciones únicas: Cada dispositivo en un bus I2C tiene una dirección única, lo que facilita la selección del dispositivo con el que deseas comunicarte.
- Protocolo Maestro-Esclavo: Permite un control preciso del flujo de datos entre un dispositivo maestro y varios dispositivos esclavos.
- Protocolo sincronizado: El bus I2C es síncrono y permite la transferencia de datos a velocidades moderadas.
- Soporte de lectura/escritura: I2C admite tanto operaciones de lectura como de escritura, lo que permite una comunicación bidireccional.

Desventajas:

- Velocidad limitada: Comparado con SPI, I2C suele ser más lento debido a su naturaleza síncrona.
- Topología limitada: La longitud y la topología del bus I2C pueden ser limitantes en aplicaciones que requieren distancias largas o conexiones complejas.
- Complejidad: La implementación de múltiples dispositivos en un bus I2C puede volverse compleja debido a las direcciones únicas y la gestión de colisiones.

El ESP32 dispone de un único canal I2C para conectar varios módulos en un mismo bus. En esta ocasión, se conectará al bus I2C únicamente el módulo que se detallará a continuación:

2.2.1.1 Modulo GY-271

Este módulo de brújula está diseñado para la detección magnética de campo bajo con una interfaz digital y es perfecto para brindar información de dirección precisa. Este sensor compacto se adapta a proyectos pequeños como UAV y sistemas de navegación robótica. El sensor convierte el campo magnético en una salida de voltaje diferencial en 3 ejes.

Ref [GY-271 HMC5883L Datasheet/ <https://handsontec.com>]

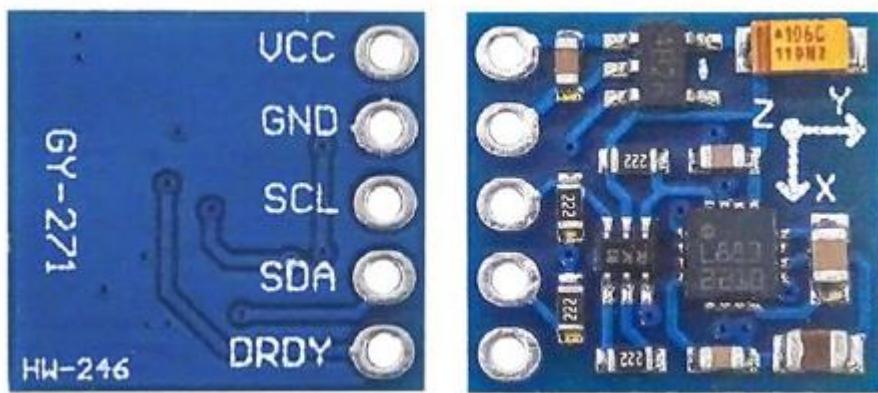


Figura 28: Modulo GY-271 por <https://handsontec.com>

Características del Módulo:

- *Número de ejes 3.*
 - *ADC de 16 bits con sensores AMR de bajo ruido.*
 - *Chip sensor: HMC5883L.*
 - *Fuente de alimentación: 3-5 V.*
 - *Comunicación: I2C.*
 - *Rango de Medición: +- 8 Gauss*

Este componente permite determinar la orientación del vehículo con respecto al norte geográfico, proporcionando datos esenciales para la navegación autónoma del vehículo que se verán en el capítulo 7.

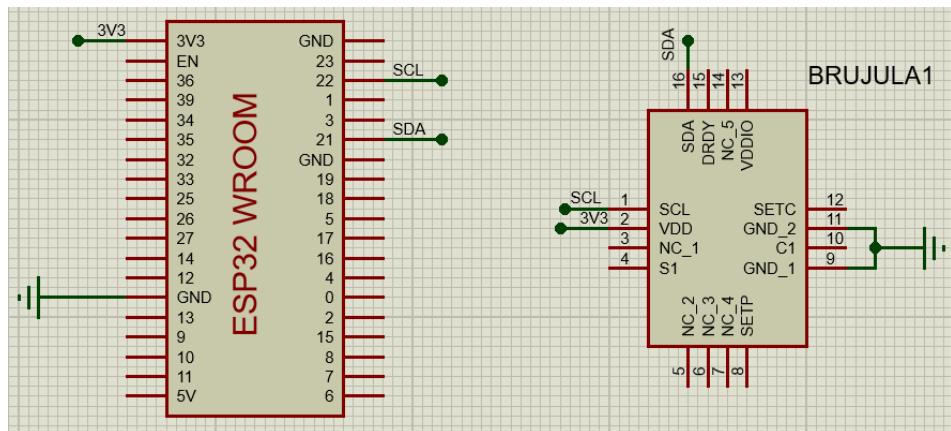


Figura 29: Esquema de conexión del Módulo GY-271 con ESP32, Edición Propia

2.2.2 Protocolo UART, modulo GPS NEO 6M y Sensor JSN-SR04T

UART significa "Universal Asynchronous Receiver-Transmitter" (Receptor-Transmisor Universal Asincrónico), y es un protocolo de comunicación utilizado para transmitir datos entre dispositivos electrónicos. Es ampliamente utilizado para establecer conexiones seriales punto a punto entre dispositivos, como microcontroladores, sensores, módulos de comunicación y otros componentes electrónicos.

El UART opera de manera asíncrona, lo que significa que no está sincronizado por un reloj central. En su lugar, utiliza señales de inicio y parada para delimitar cada byte de datos transmitido.

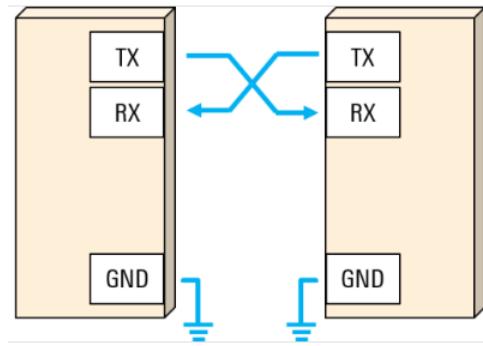


Figura 30: Conexión entre dos dispositivos UART por <https://www.rohde-schwarz.com/>

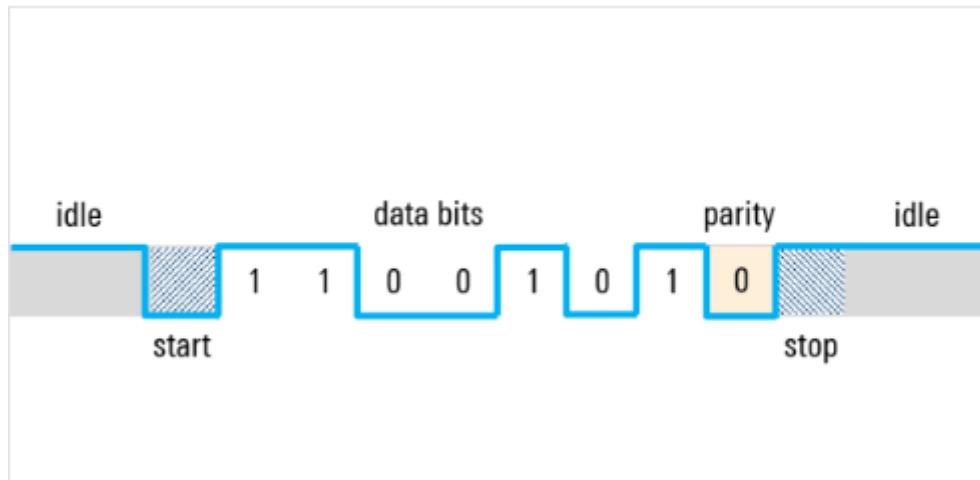


Figura 31: Formato de trama de datos UART por <https://www.rohde-schwarz.com/>

- Bits de datos: Cada byte de datos se divide en bits individuales (generalmente 8 bits), y se transmiten uno tras otro.
- Bit de inicio: Antes de transmitir un byte, se envía un bit de inicio, que siempre es un nivel bajo (0). Esto indica al receptor que se está iniciando una nueva transmisión y le permite sincronizarse.
- Bits de datos: A continuación, se envían los bits de datos, comenzando por el bit menos significativo (LSB) y terminando en el bit más significativo (MSB). Cada bit se transmite durante un período de tiempo fijo.

- Bits de paridad (opcional): Algunas configuraciones de UART incluyen un bit de paridad que se utiliza para verificar la integridad de los datos transmitidos. La paridad puede ser par o impar, y ayuda a detectar errores de transmisión.
- Bits de parada: Despues de transmitir los bits de datos y, si corresponde, el bit de paridad, se envían uno o más bits de parada. Estos son niveles altos (1) que indican al receptor que la transmisión ha finalizado.
- Intervalo entre caracteres: Entre transmisiones sucesivas, hay un intervalo de tiempo durante el cual la línea de datos permanece en el estado de alto (1). Esto ayuda a separar los bytes transmitidos.

Es importante destacar que tanto el transmisor como el receptor deben estar configurados con la misma velocidad de transmisión (baud rate) para que la comunicación sea exitosa. El baud rate determina la velocidad a la que se transmiten los bits y se mide en bits por segundo (bps). Valores comunes para el baud rate son 9600, 115200, etc.

Se detallan algunas ventajas y desventajas a tener en cuenta para este protocolo:

Ventajas:

- Simpleza: El protocolo UART es simple y fácil de implementar, ideal para aplicaciones básicas de comunicación punto a punto.
- Conexión punto a punto: UART establece una conexión directa entre dos dispositivos, lo que puede ser útil en escenarios donde se requiere una comunicación simple y directa.
- Ampliamente compatible: La mayoría de los microcontroladores y dispositivos electrónicos incluyen soporte nativo para UART.
- Velocidad variable: Puedes ajustar la velocidad de transmisión (baud rate) según tus necesidades.

Desventajas:

- Unidireccionalidad: A diferencia de I2C y SPI, UART no admite una comunicación simultánea de lectura y escritura sin configuraciones adicionales.
- Menos funcionalidades: UART carece de características avanzadas de detección y corrección de errores presentes en I2C y SPI.
- No es adecuado para redes complejas: Debido a su naturaleza punto a punto, UART no es la mejor opción para configuraciones con múltiples dispositivos.

El ESP32 ofrece dos interfaces UART, en este proyecto se utilizan ambas. Estas interfases están configuradas a una velocidad de 9600 baudios, lo que permite la obtención de los datos necesarios de los módulos que se describen a continuación.

2.2.2.1 Modulo GPS NEO 6M

La serie de módulos NEO-6 es una familia de receptores GPS independientes que cuentan con el motor de posicionamiento u-blox 6 de alto rendimiento. Estos receptores flexibles y rentables ofrecen numerosas opciones de conectividad en un paquete en miniatura de 16 x 12,2 x 2,4mm. Su arquitectura compacta y sus opciones de alimentación y memoria hacen que los módulos NEO-6 sean ideales para dispositivos móviles que funcionan con baterías con limitaciones de espacio y costo muy estrictas.

Ref [NEO-6_Datasheet / <https://handsontec.com>]

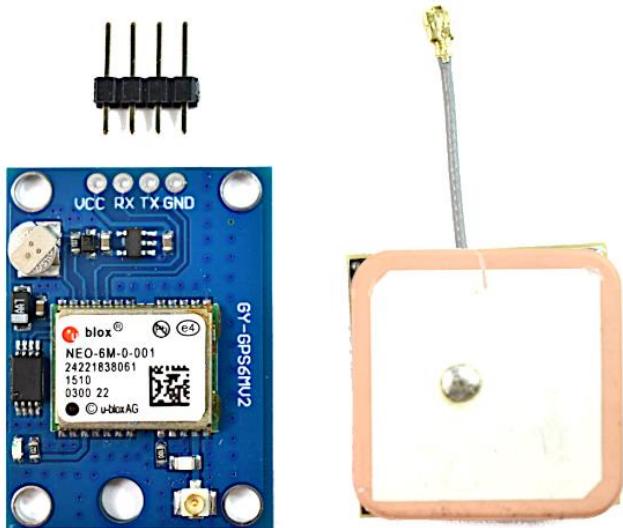


Figura 32: Modulo GY-GPS6MV2 por <https://handsontec.com>

Características del Módulo:

- *Alimentación: 2.7 V – 3.6 V.*
- *Comunicación: UART.*
- *Tiempo hasta la primera obtención de datos: 27 segundos.*
- *Tasa máxima de actualización de navegación: 5Hz.*
- *Precisión de la posición Horizontal: 2.5 metros.*
- *Precisión de Velocidad: 0.1 m/s.*

Este módulo es fundamental para adquirir las coordenadas geográficas del vehículo, incluyendo latitud y longitud. Estos datos posibilitan el seguimiento en tiempo real de la ubicación del vehículo y proporcionan la información esencial para su navegación. Además de rastrear la posición del vehículo, estas coordenadas tienen un papel crucial en la obtención de otros datos fundamentales, que se detallarán más adelante en el informe.

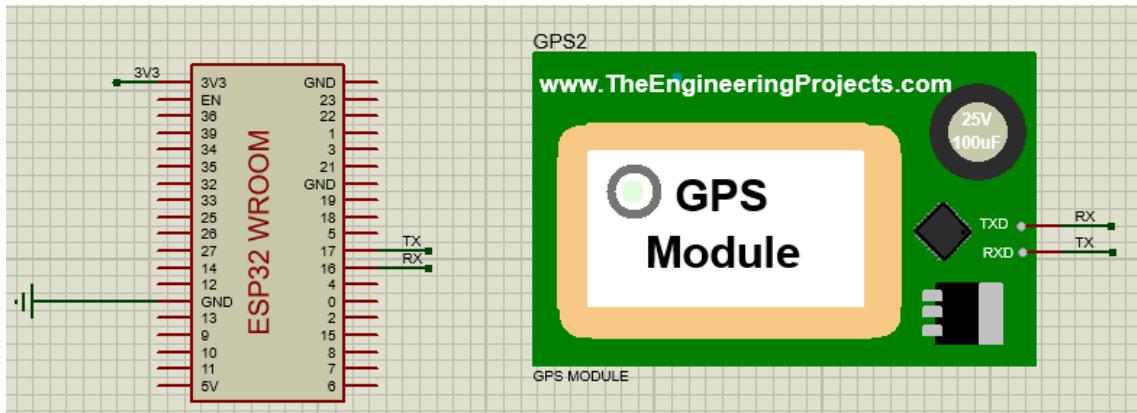


Figura 33: Esquema de conexión del módulo GPS NEO 6M con ESP32, Edición Propia

2.2.2.2 Sensor JSN-SR04T

El sensor ultrasonido JSN-SR04T es un sensor de distancia que utiliza ultrasonido (sonar) para determinar la distancia de un objeto en un rango de 25 a 450 cm. Destaca especialmente por su resistencia al agua, pequeño tamaño, bajo consumo energético y buena precisión. Perfecto para aplicaciones donde el sensor estará expuesto a la intemperie/humedad/agua, utilizado en automóviles para medir distancia de colisión/parqueo.

El sensor trabaja con ultrasonido y contiene toda la electrónica encargada de hacer la medición. El funcionamiento del sensor es el siguiente: se emite un pulso de sonido (TRIG), se mide la anchura del pulso de retorno (ECHO), se calcula la distancia a partir de las diferencias de tiempos entre el TRIG y ECHO. El funcionamiento no se ve afectado por la luz solar o material negro (aunque los materiales blandos acústicamente como tela o lana pueden ser difícil de detectar).

La distancia se puede calcular utilizando la siguiente formula:

$$\text{Distancia}(m) = \{(Tiempo\ del\ pulso\ ECO) * (\text{Velocidad}\ del\ sonido=340m/s)\}/2$$

Ref:[<https://naylampmechatronics.com/sensores-proximidad/326-sensor-ultrasonido-jsn-sr04t.html>]



Figura 34: Sensor JSN-SR04T por <https://naylampmechatronics.com/>

Características del Sensor:

- *Voltaje de alimentación: 5V DC*
- *Corriente de trabajo: 30mA*
- *Rango de detección: 25cm - 450cm*
- *Precisión: +-0.3mm*
- *Frecuencia de emisión acústica: 40KHz*
- *Duración mínima del pulso de disparo (nivel TTL): 10μS*
- *Tiempo mínimo de espera entre una medida y el inicio de otra 20mS*
- *Ángulo de detección: menor a 50º*
- *A prueba de agua (parte delantera)*
- *Dimensiones tarjeta: 41*28.5 mm*
- *Dimensiones transductor: D25*L19 mm*
- *Temperatura de trabajo: -10ºC hasta 70ºC*

Este sensor se utiliza para detectar posibles obstáculos y, en consecuencia, tomar las acciones necesarias para que el vehículo pueda superarlos con éxito y continuar su trayectoria hacia el punto de destino. La elección de este sensor se basa en su capacidad

de resistencia al agua, una característica fundamental para este proyecto, dado que se desenvolverá en entornos acuáticos.

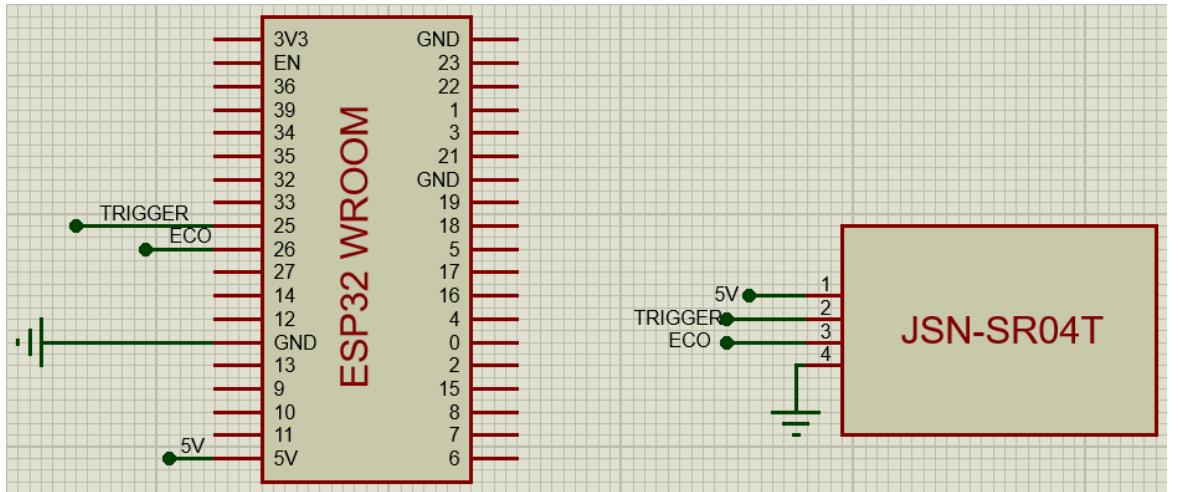


Figura 35: Esquema de conexión del módulo JSN-SR04T con ESP32, Edición Propia

2.2.3 Modulo L298N

Este controlador de motor bidireccional dual se basa en el muy popular circuito integrado de controlador de motor de puente H dual L298. El circuito le permitirá controlar de manera fácil e independiente dos motores de hasta 2 A cada uno en ambas direcciones. Es ideal para aplicaciones robóticas y muy adecuado para la conexión a un microcontrolador que requiere solo un par de líneas de control por motor. También se puede interconectar con interruptores manuales simples, puertas lógicas TTL, relés, etc. Esta placa está equipada con indicadores LED de alimentación, regulador de +5V integrado y diodos de protección.

Ref [L298N- Datasheet / www.handsontec.com]

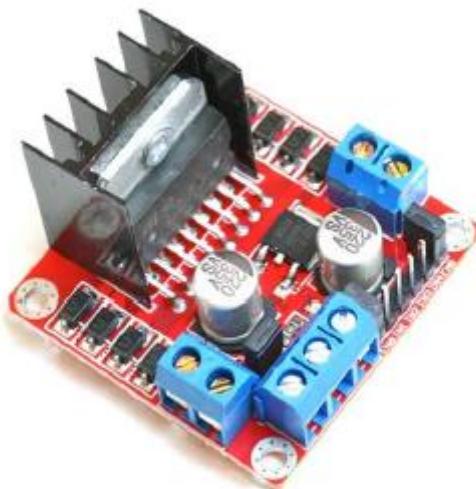


Figura 36: Modulo L298N por www.handsontec.com

Características del Módulo:

- *Voltaje de entrada: 3,2 V ~ 40 V CC.*
- *Controlador: controlador de motor de CC de puente doble H L298N*
- *Fuente de alimentación: CC 5 V - 35 V*
- *Corriente máxima: 2 amperios*
- *Rango de corriente de funcionamiento: 0 ~ 36mA*
- *Rango de voltaje de entrada de la señal de control:*
 - *Bajo: $-0.3V \leq Vin \leq 1,5 V$.*
 - *Alto: $2.3V \leq Vin \leq vs$.*
- *Habilitar rango de voltaje de entrada de señal:*
 - *Bajo: $-0.3 \leq Vin \leq 1,5 V$ (la señal de control no es válida).*
 - *Alto: $2.3V \leq Vin \leq Vss$ (señal de control activa)*
- *Consumo máximo de energía: 20W (cuando la temperatura $T = 75^{\circ}C$).*
- *Temperatura de almacenamiento: $-25^{\circ}C \sim +130^{\circ}C$.*
- *Suministro de salida regulado de +5 V incorporado.*

- Tamaño: 3,4 cm x 4,3 cm x 2,7 cm.

Este módulo, basado en un puente H, se utiliza para controlar el motor DC que es responsable del movimiento del vehículo. Emplea señales digitales, incluyendo PWM, para controlar tanto la velocidad como la dirección de giro. No requiere la implementación de un protocolo de comunicación específico; en su lugar, simplemente se conectan tres pines digitales del ESP32 para gestionar su funcionamiento.

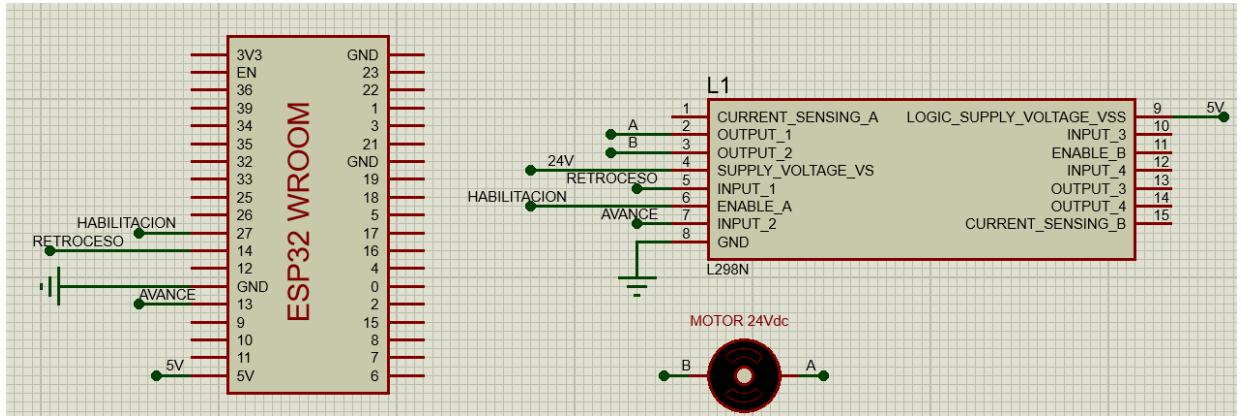


Figura 37: Esquema de conexión del módulo L298N con ESP32, Edición Propia

2.2.4 Modulo Step-Down XL4005

Este es un convertidor de voltaje Step-Down DC-DC tipo Buck, con un potenciómetro multivuelta para regular el voltaje deseado. Es capaz de manejar una carga de hasta 5A con una alta eficiencia. Cuando se emplee para una corriente mayor a 2.5A se recomienda utilizarlo con un disipador de calor.

Este circuito te permite tener un voltaje regulado a partir de una fuente de alimentación con un voltaje mayor, por ejemplo, si tienes una fuente de 12V puedes regularlos a 5V, 3.3V, 2.2V, etc, para el uso con microcontroladores, Arduino, PICs, Raspberry Pi, fuentes variables, drivers para leds, etc.

Este módulo está basado en el Regulador DC-DC Step Down XL4005 que es un circuito integrado monolítico adecuado para el diseño fácil y conveniente de una fuente de conmutación tipo buck. Es capaz de conducir una corriente de hasta 5A. Maneja una carga con excelente regulación de línea y bajo voltaje de rizado. Este dispositivo está disponible con voltaje de salida ajustable. El módulo reduce al mínimo el uso de componentes externos para simplificar el diseño de fuentes de alimentación.

El módulo convertidor XL4005 es un regulador de tipo conmutado, así que su eficiencia es significativamente mayor en comparación con los populares reguladores lineales de tres terminales, especialmente con tensiones de entrada superiores.

Ref:[<https://www.todomicro.com.ar/investigacion-desarrollo-yprototipado/352-modulo-step-down-xl4005.html>]

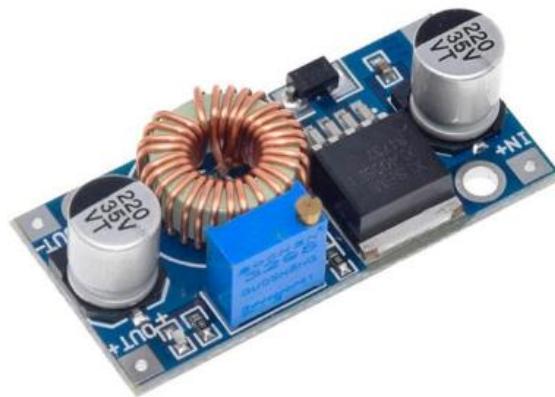


Figura 38: Modulo Step-Down con CI XL4005 por <https://www.dled.com.ar/>

Este módulo desempeña un papel crucial al convertir un voltaje de entrada de 12Vdc, suministrado por la batería, en un voltaje de salida de 5Vdc. Este último voltaje es necesario para alimentar los módulos que operan a esta tensión.

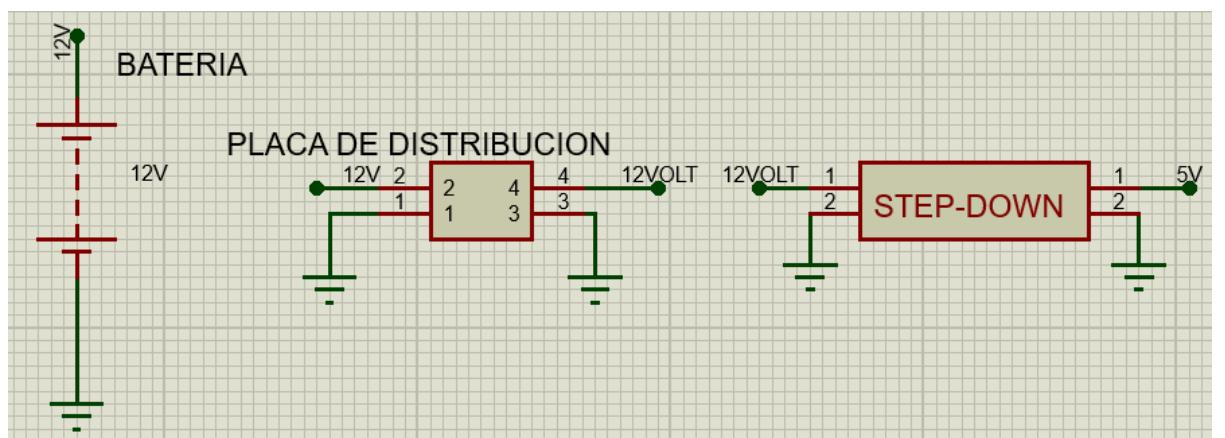


Figura 39: Esquema de conexión de modulo Step-Down con Batería, Edición propia

2.2.5 Módulo Step-Up XL6009

El convertidor DC-DC XL6009 es un regulador de tipo conmutado elevador (Step-Up o Boost) con una alta eficiencia de conversión, excelente regulación de línea y bajo voltaje de rizado. El módulo reduce al mínimo el uso de componentes externos para simplificar el diseño de fuentes de alimentación. Permite obtener un voltaje regulado a partir de una fuente con un voltaje inferior, por ejemplo: obtener 5V o 12V a partir de una batería de litio de 3.7V. Es capaz de manejar una carga de hasta 2.5A o 10W máx.

Ref:[<https://www.componentesmerlo.com.ar/MLA-758043915-fuente-step-up-xl6009-xl-6009-elevador-125v-35v-4a-arduino- JM>]



Figura 40: Módulo Step-Up con CI XL6009 por <https://www.todomicro.com.ar/>

Características del Módulo:

- Convertidor DC-DC Boost: XL6009
- Voltaje de entrada: 5V a 32V DC
- Voltaje de salida: 7V a 35V DC
- V. Salida ajustable (Regulable por trimmer)
- Corriente de salida: máx. 2.5A (usar disipador para corrientes mayores a 2A)
- Potencia de salida: 10W
- Eficiencia de conversión: 94% máx.
- Regulación de carga: $S(I) = 0.5\%$.

- *Regulación de voltaje: $S(u) = 0.5\%$.*
- *Frecuencia de Trabajo: 400KHz*
- *Protección de sobre-temperatura: SI (apaga la salida)*
- *Protección de corto circuito: NO*
- *Protección limitadora de corriente: SI (4A)*
- *Protección frente a inversión de polaridad: NO*
- *Dimensiones: 43mm*20mm*14mm*

Este módulo cumple la función de elevar la tensión de entrada de 12Vdc, suministrada por la batería, a un voltaje de salida de 24Vdc. Esta elevación de voltaje es esencial para alimentar y hacer funcionar la parte mecánica del vehículo de manera efectiva.

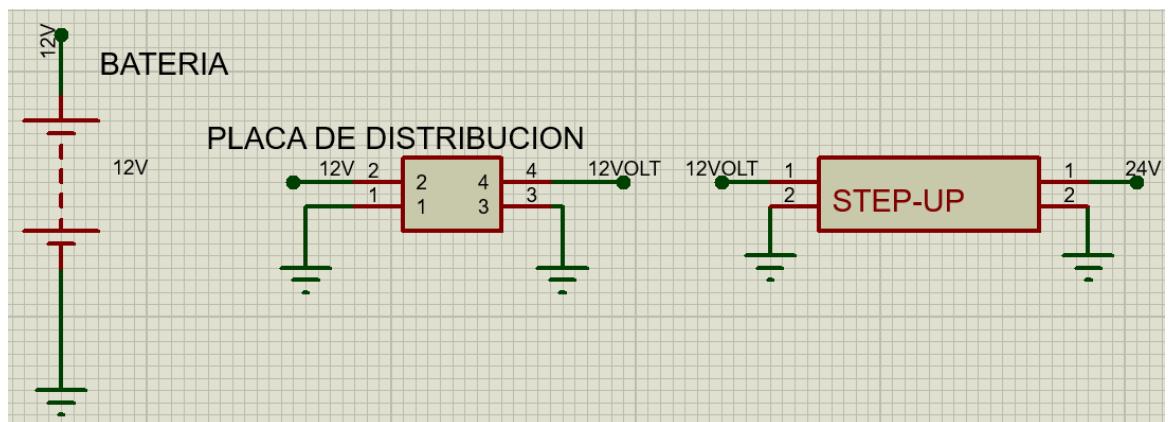


Figura 41: Esquema de conexión de modulo Step-Up con Batería, Edición Propia

3 Capítulo 3: SISTEMA MECANICO

Hasta el momento, hemos explorado en detalle los componentes electrónicos ubicados en los compartimentos estancos uno, dos y tres del vehículo acuático autónomo. Sin embargo, aún no hemos abordado la parte mecánica que compone el compartimento estanco 2. En este capítulo, profundizaremos en la construcción y funcionamiento de esta parte fundamental del vehículo, destacando los componentes clave que la componen y su contribución al conjunto del prototipo.

3.1 Motor DC RS-455PA

El motor DC es la fuente de energía principal del vehículo. Este motor convierte la energía eléctrica en energía mecánica al girar su eje. La velocidad y la dirección de giro del motor DC pueden controlarse variando la polaridad de la corriente eléctrica que fluye a través de él (L298N). Esto permite al vehículo avanzar hacia adelante o hacia atrás, así como controlar su velocidad.

El Motor utilizado fue el siguiente:

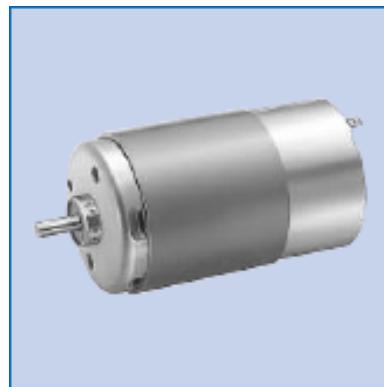


Figura 42: Motor DC RS-455PA- 15200 por Mabuchi Motor

Sus características son las siguientes:

MODEL	VOLTAGE		NO LOAD		AT MAXIMUM EFFICIENCY					STALL		
	OPERATING RANGE	NOMINAL	SPEED	CURRENT	SPEED	CURRENT	TORQUE		OUTPUT	TORQUE		CURRENT
			r/min	A			r/min	A		mN·m	g·cm	
RS-455PA-15200	12~42	*42V CONSTANT	6300	0.058	5350	0.33	15.8	161	8.83	105	1070	1.85
RS-455PA-18140	12~30	21V CONSTANT	4500	0.078	3730	0.38	13.0	132	5.06	76.0	775	1.84

Figura 43: Características del Motor DC RS-455PA

3.2 Eje acoplado al Motor

Para habilitar el movimiento de la hélice del vehículo, se ha implementado un sistema cardán-eje. El cardán, es un ingenioso sistema mecánico conceptualizado por Girolamo

Cardano, que permite la conexión de dos ejes que no son coaxiales, lo que significa que no están alineados en una misma línea recta. Su principal objetivo es transmitir el movimiento de rotación desde un eje conductor a otro eje conducido, a pesar de que estos no se encuentren en posición colineal.

En este caso, se ha utilizado una varilla roscada de 3 mm como eje conducido, que se conecta al cardán en uno de sus extremos y, posteriormente, a la hélice que ha sido diseñada específicamente para este propósito. A continuación, se presentan imágenes que ilustran este sistema para una comprensión visual más completa.

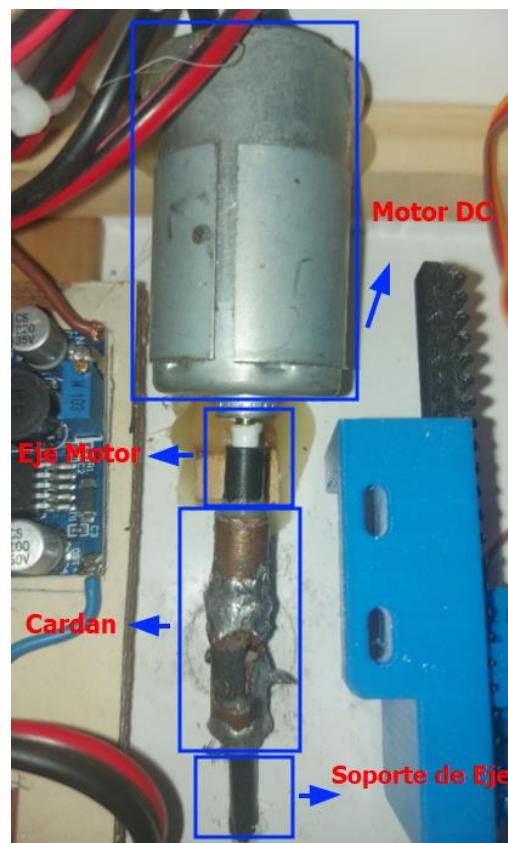


Figura 44: Conjunto Motor-Cardan-Eje, Edición Propia



Figura 45: Soporte de Eje conducido, Edición Propia

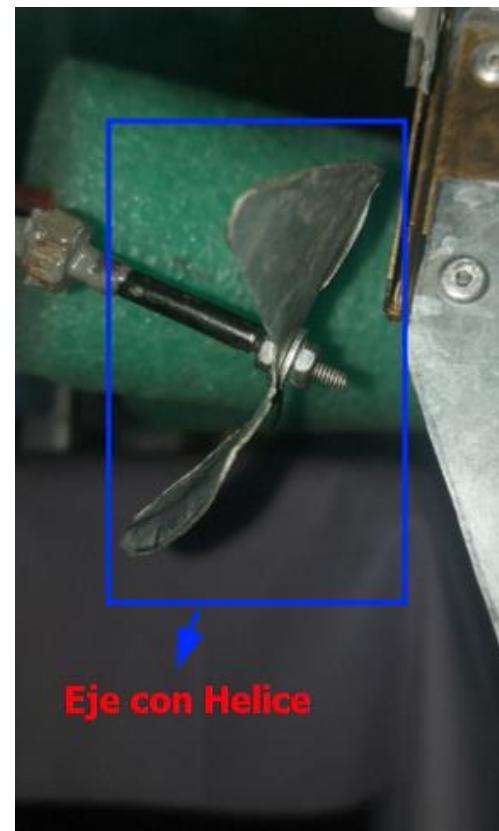


Figura 46: Eje conducido con hélice acoplada, Edición Propia

3.3 Hélice

La hélice es un elemento formado por una serie de dispositivos que se denominan palas o álabes, dispuestos de forma concéntrica sobre un eje y que giran alrededor de éste en un mismo plano. Gracias a sus álabes pueden transmitir su energía cinética, creada al girar los álabes, a un fluido de manera que se crea una fuerza de tracción.

El funcionamiento de la hélice se debe al diferente flujo que se produce entre las caras activa y pasiva, debido a la forma de las secciones de la pala y a su ángulo de ataque (paso de la hélice). Este diferente flujo provoca un empuje, por las diferencias de presión que aparecen, de manera similar a como ocurre en las alas de un avión. El resultado es que cada pala produce un empuje en la dirección del avance del barco, gracias al movimiento rotatorio del conjunto.

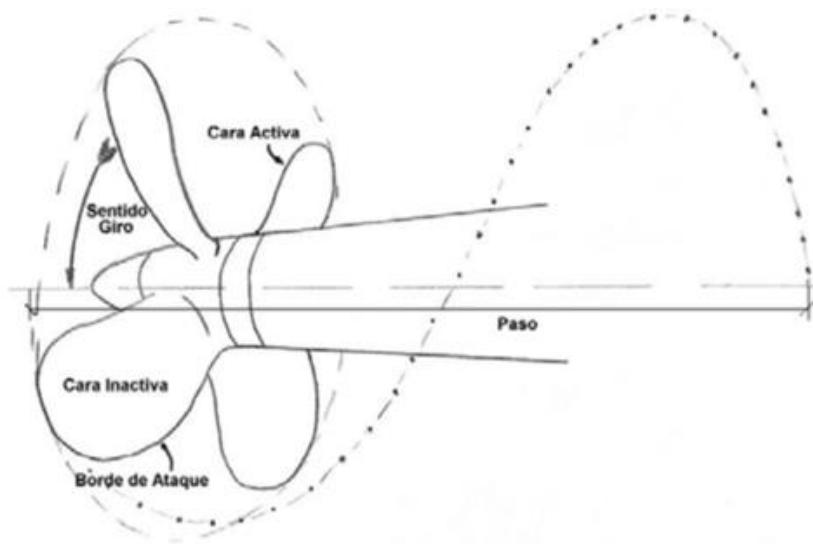


Figura 47: Funcionamiento de una Hélice por Trabajo final de carrera para la obtención de la Diplomatura en Navegación Marítima de Rubén Heras Zurita

Características

- Paso: avance cuando da una vuelta completa.
- Retroceso: diferencia entre el avance hipotético y el avance real.
- Diámetro: circunferencia circunscrita a los extremos de las palas.
- Sentido de giro: dextrógiras (paso a la derecha mirando desde proa) o levógiras (paso a la izquierda).

En este proyecto, se ha diseñado una hélice de tres palas dispuestas a intervalos de 120 grados entre sí, un enfoque comúnmente utilizado en el ámbito de los prototipos y vehículos acuáticos. Un aspecto importante a destacar es que esta hélice tiene un paso fijo, también conocido como "paso constante". Esto significa que el valor del paso permanece invariable en toda la superficie de las palas, con la excepción de sus ángulos. Esta configuración permite que, al girar en sentido de las agujas del reloj, el vehículo avance, mientras que, al girar en sentido contrario, retroceda de manera eficiente.

A continuación, se presentarán imágenes de la hélice diseñada para una mejor comprensión visual.



Figura 48: Hélice realizada para el vehículo, vista de frente desde el motor, Edición Propia



Figura 49: Vista de frente desde la parte trasera, Edición Propia

Ref:[https://www.youtube.com/watch?v=XTI-1fAyBX8&t=1187s&ab_channel=Carttthum]

3.4 Servo Motor y Aleta como Timón

El servo motor es un dispositivo que se utiliza para controlar la posición de la aleta que actúa como timón. El servo motor es especialmente adecuado para aplicaciones de control de posición precisa, como la dirección del vehículo. Puede girar el timón a un ángulo específico según las señales de control que recibe. Se implementó un diseño 3D “servoactuador” para transformar el movimiento rotatorio del servo en un movimiento lineal para tener una mejor precisión. El cual fue realizado por “protentprintables”, el diseño es el siguiente:



Figura 50: Servo Motor SG90 utilizado, Edicion Propia

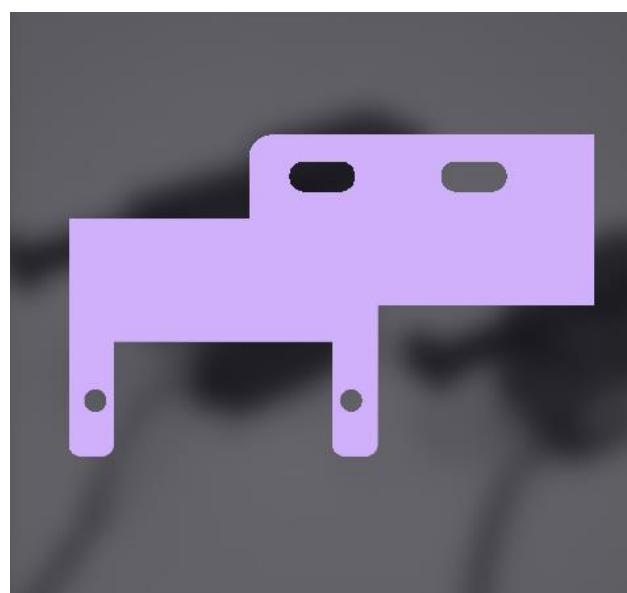


Figura 51: Vista superior soporte del servoactuador lineal por <https://cults3d.com/>

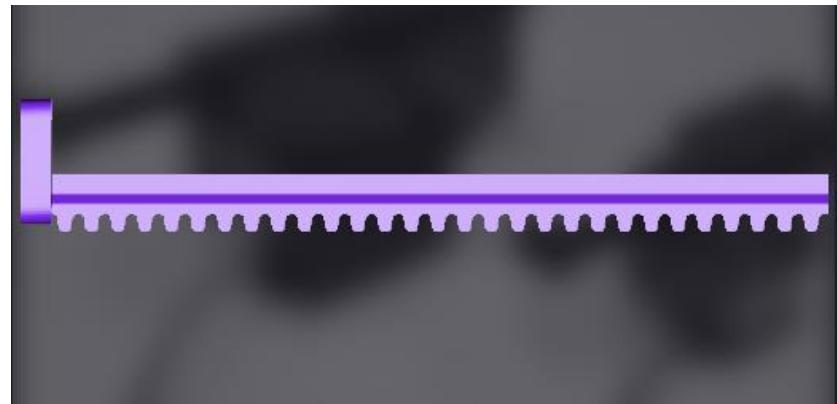


Figura 52: Brazo “pusher” del servoactuador para el movimiento lineal por <https://cults3d.com/>

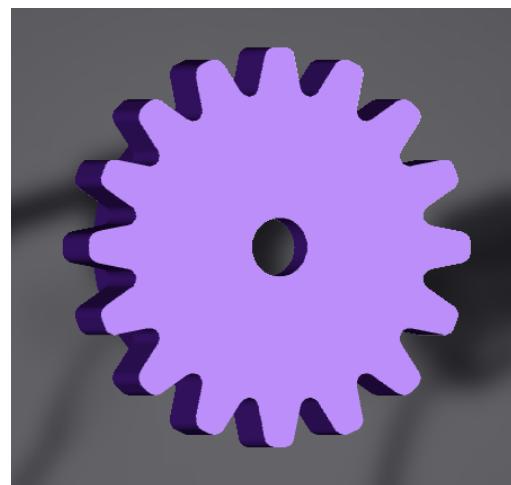


Figura 53: Engranaje de piñón del servoactuador por <https://cults3d.com/>



Figura 54: Diseño final acoplado al servomotor por <https://cults3d.com/>

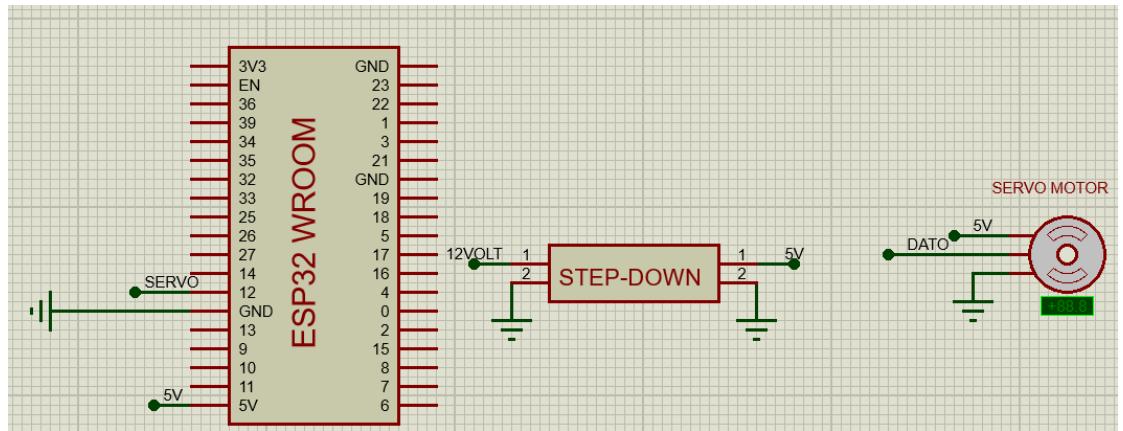


Figura 55: Esquema de conexión del servomotor con ESP32, Edición Propia

3.4.1 Aleta como Timón

La aleta funciona como el timón del vehículo. Se encuentra en la parte trasera del vehículo y gira alrededor de un eje vertical. El servo motor controla el ángulo de giro de la aleta, lo que permite cambiar la dirección del vehículo en el agua. Cuando la aleta gira hacia la derecha, el vehículo se moverá hacia la derecha, y viceversa.

Para sujetar la aleta, se diseñó un brazo que está conectado al compartimento estanco dos. En el extremo de este brazo, se ha instalado una bisagra que permite el movimiento rotatorio de la aleta. La bisagra también incluye un codo a 90 grados que tiene la función de facilitar el movimiento de la aleta mediante el servoactuador mencionado anteriormente. Para una comprensión más clara de este sistema, se proporcionarán imágenes ilustrativas.

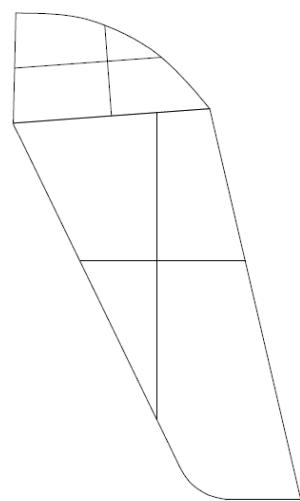


Figura 56: Plano de aleta realizada por Estudio, diseño, automatización y construcción de una embarcación RC
impresa en 3D Trabajo realizado por: Roger Martín Valls.

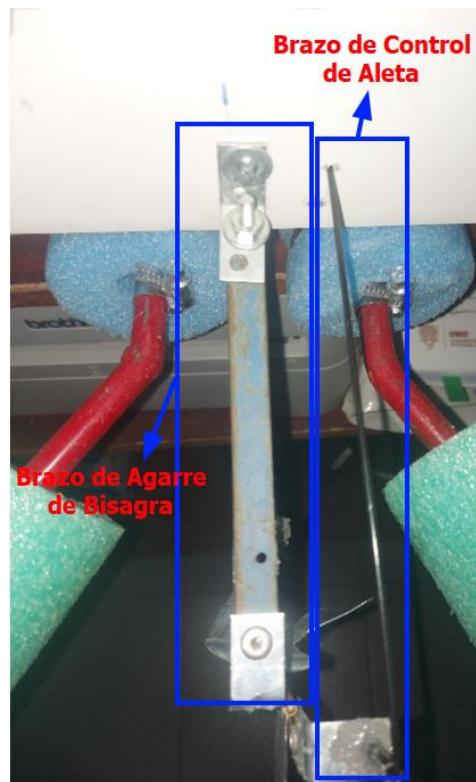


Figura 57: Brazo de agarre de bisagra unido al compartimento estanco tres y brazo para control de movimiento
de aleta, Edición Propia

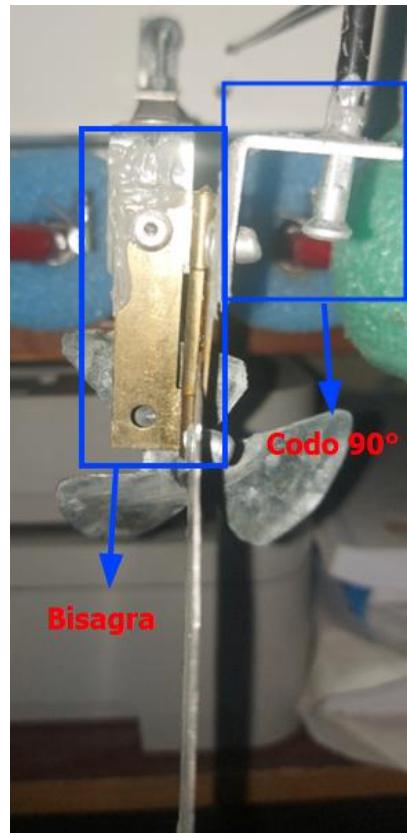


Figura 58: Conjunto bisagra y codo 90°, Edición Propia

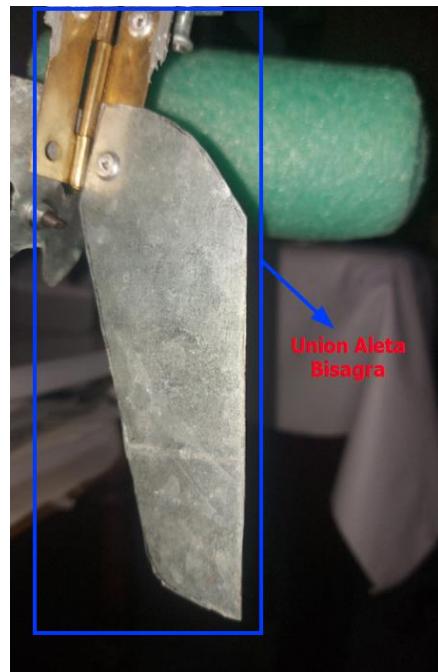


Figura 59: Unión de Aleta con Bisagra para movimiento de aleta, Edición Propia



Figura 60: Brazo de control de aleta unido a servoactuador lineal, Edición Propia

4 Capítulo 4: SISTEMA DE ALIMENTACION AUTONOMA

En este capítulo, nos adentraremos en el sistema de alimentación crucial para el funcionamiento óptimo del bloque del vehículo. Este sistema engloba componentes como un panel solar, un regulador y una batería de gel. Esto nos permitirá comprender cómo se gestiona la energía en el proyecto y cómo se garantiza un suministro continuo para las diversas funciones del vehículo.

4.1 Componentes de un sistema de alimentación fotovoltaico

Un sistema fotovoltaico aprovecha la radiación solar mediante paneles solares para generar electricidad. Esta electricidad se almacena en baterías o se utiliza directamente para alimentar dispositivos y electrodomésticos a través de un inversor. El sistema es una forma sostenible y eficiente de obtener energía eléctrica, reduciendo la dependencia de fuentes de energía no renovables.

Sus componentes principales son:

- **Paneles Solares (Módulos Fotovoltaicos):** Son dispositivos que contienen células solares que absorben la luz solar y la convierten en electricidad. Los paneles solares están compuestos por varias células conectadas en serie o en paralelo.
- **Regulador de Carga (Controlador Solar):** Este dispositivo controla la cantidad de energía que fluye desde los paneles solares a la batería. Evita la sobrecarga de la batería y garantiza su larga vida útil.
- **Batería (Acumulador):** Almacena la electricidad generada por los paneles solares para su uso posterior. Las baterías son esenciales para garantizar un suministro constante de electricidad incluso cuando el sol no brilla.
- **Inversor:** Convierte la electricidad continua (CC) generada por los paneles solares y almacenada en la batería en electricidad alterna (CA), que es la forma de electricidad utilizada en la mayoría de los dispositivos y electrodomésticos.

- Cableado y Conexiones: Los cables y conexiones eléctricas conectan todos los componentes del sistema y permiten que la electricidad fluya de manera segura y eficiente.

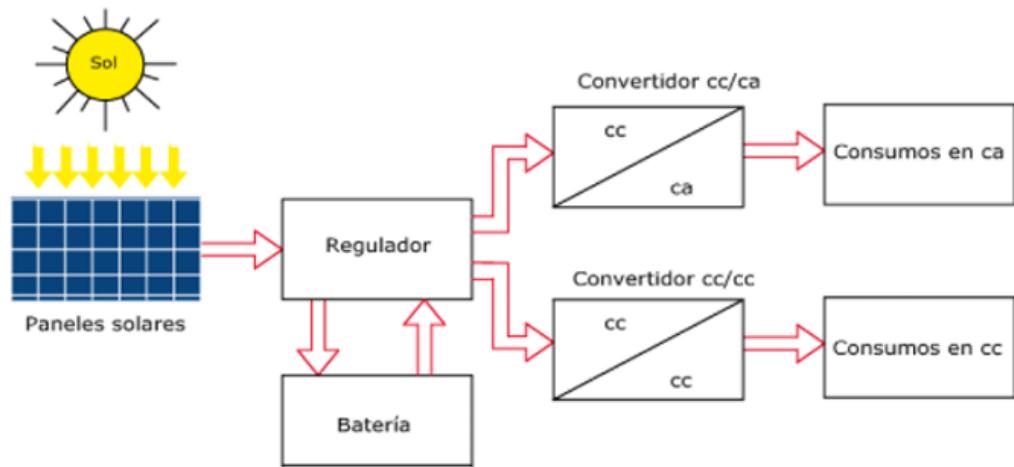


Figura 61: Elementos de una instalación de energía fotovoltaica por Tobajas, 2018

Como se puede observar, después de pasar por el regulador, la energía solar fotovoltaica puede ser utilizada para alimentar cargas tanto en corriente continua como en corriente alterna. En el caso de este proyecto, se emplea para alimentar cargas en corriente continua. Sin embargo, la tensión entregada a la salida del regulador es de 12Vdc. Por lo tanto, se utilizan los reguladores de corriente continua a corriente continua (cc/cc) mencionados anteriormente, tanto para reducir la tensión a 5Vdc como para elevarla a 24Vdc, lo que garantiza el correcto funcionamiento de las diferentes partes del bloque del vehículo. El esquema de conexión en el vehículo se observa a continuación:

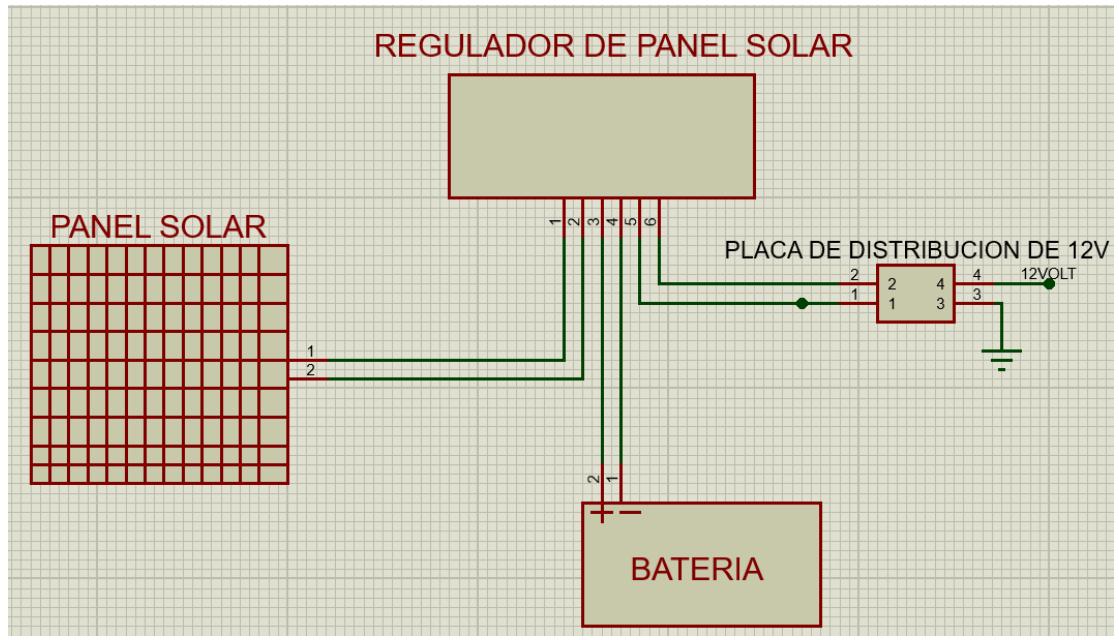


Figura 62: Esquema de conexión del sistema de la alimentación del vehículo, Edición Propia

4.2 Principio de Funcionamiento del Panel Solar

Los rayos del sol golpean los paneles solares. Las células de los paneles fotovoltaicos están formadas por semiconductores (silicio) que, bajo el efecto de los rayos solares, crean un campo eléctrico en su interior. El movimiento de electrones entre las 2 capas del semiconductor crea una corriente eléctrica de origen fotovoltaico.

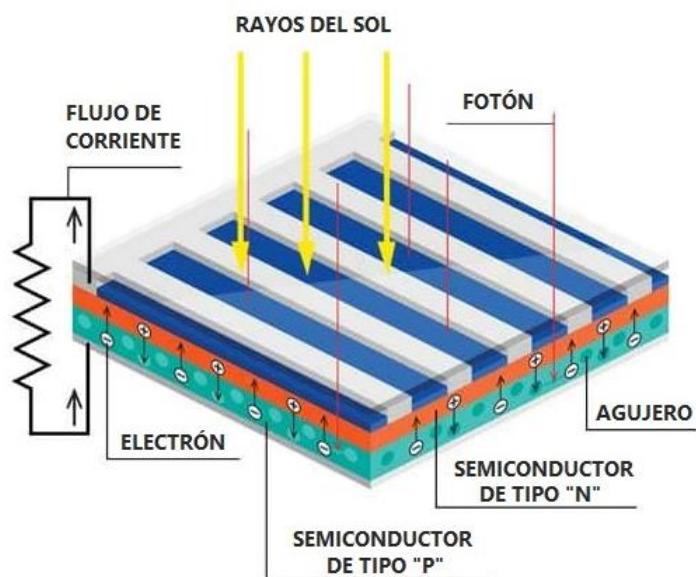


Figura 63: Semiconductores con impurezas de tipo n y p por <https://www.mikitsolar.es/>

La eficiencia de conversión de la célula que traduce la transformación de energía solar en corriente eléctrica determina el rendimiento del panel fotovoltaico. Por lo tanto, es importante entender que el panel solar funciona gracias a la radiación del sol y no gracias a su calor.

La nubosidad puede reducir la eficiencia de las células, pero no impide que el panel produzca energía y por lo tanto electricidad.

La corriente así producida por los paneles es corriente continua (DC). Es el inversor que transformará esta corriente en corriente alterna o continua según sea el caso.

4.3 Tipos de Paneles Solares

En el mercado se pueden encontrar diferentes tipos de paneles solares:

- **Paneles Solares de Silicio Monocristalino:** Estos paneles están hechos de silicio monocristalino de alta pureza y tienen una alta eficiencia de conversión de energía solar en electricidad. Son conocidos por su aspecto uniforme y su alta eficiencia, lo que significa que ocupan menos espacio. Sin embargo, son más caros de producir.
- **Paneles Solares de Silicio Policristalino:** Estos paneles están hechos de silicio policristalino y son menos costosos de producir que los monocristalinos, aunque también son un poco menos eficientes. Son reconocibles por su aspecto moteado.
- **Paneles especiales:** dentro de esta categoría podemos encontrar paneles de sulfuro de cadmio y sulfuro de cobre, de arsénico de galio y de di seleniuro de cobre en indio.

4.3.1 Panel solar LTC Electronics

Para el proyecto realizado se utilizó un panel solar de 10W, de tipo monocristalino y con una superficie menor a un metro cuadrado. Se lo puedo observar en las siguientes figuras:



Figura 64: Panel Solar LTC Electrónico de 10W, Edición Propia



Figura 65: Características del panel definidas por el fabricante, Edición Propia



Figura 66: Prueba de tensión del panel Solar, Edición Propia



Figura 67: Prueba de corriente del panel solar, Edición Propia

Las imágenes previamente mostradas confirman el adecuado funcionamiento del panel solar, proporcionando una tensión de 21,6V y una corriente cercana a los 260mA. Estos valores se encuentran dentro de los parámetros especificados por el fabricante. Es relevante destacar que las mediciones se llevaron a cabo a una temperatura ambiente de 30°C alrededor de las 13:00 horas.

4.4 Reguladores de Carga

En el sistema de alimentación fotovoltaica, el regulador desempeña un papel crucial al controlar los voltajes y corrientes suministrados a la batería. Su función principal es garantizar una vida útil más prolongada de la batería al evitar sobrecargas y descargas excesivas. Además, el regulador impide que la corriente fluya desde la batería hacia el panel solar cuando este no genera energía debido a la falta de luz solar.

La mayoría de los reguladores están equipados con características importantes, como un amperímetro para medir la corriente, un voltímetro para medir el voltaje, indicadores de baja tensión para advertir sobre condiciones peligrosas, un sensor de temperatura para ajustar el rendimiento según las variaciones de temperatura y un diodo de bloqueo para prevenir el flujo de corriente inversa.

Esta combinación de características garantiza un control eficiente y seguro de la carga y descarga de la batería en un sistema fotovoltaico.

Existen dos tipos de reguladores:

Regulador Shunt o Paralelo:

- Funcionamiento: En un regulador shunt o paralelo, el flujo de corriente desde los paneles solares hacia la batería y otros dispositivos se divide en dos caminos. Una parte de la corriente fluye directamente hacia la batería para cargarla, mientras que la otra parte se "deriva" o "desvía" hacia una resistencia, conocida como "shunt" o "derivación". La corriente que pasa por el shunt se desperdicia en forma de calor.
- Control de Carga: Estos reguladores controlan el estado de carga de la batería monitorizando la tensión de la misma. Cuando la batería está completamente

cargada y alcanza su voltaje máximo, el regulador shunt corta el flujo de corriente desde los paneles solares hacia la batería para evitar la sobrecarga.

- Eficiencia: Tienden a ser menos eficientes que los reguladores serie ya que parte de la corriente se disipa como calor en el shunt.

Regulador Serie:

- Funcionamiento: En un regulador serie, el flujo de corriente desde los paneles solares hacia la batería y otros dispositivos se controla mediante un interruptor electrónico (transistor). Este interruptor se abre y cierra en función del estado de carga de la batería.
- Control de Carga: Los reguladores serie utilizan una técnica de modulación por ancho de pulso (PWM) o un seguimiento del punto de máxima potencia (MPPT) para controlar la carga de la batería. Estos sistemas ajustan la corriente y tensión entregada a la batería para maximizar la eficiencia de carga.
- Eficiencia: Suelen ser más eficientes que los reguladores shunt, ya que no se desperdicia corriente como calor.

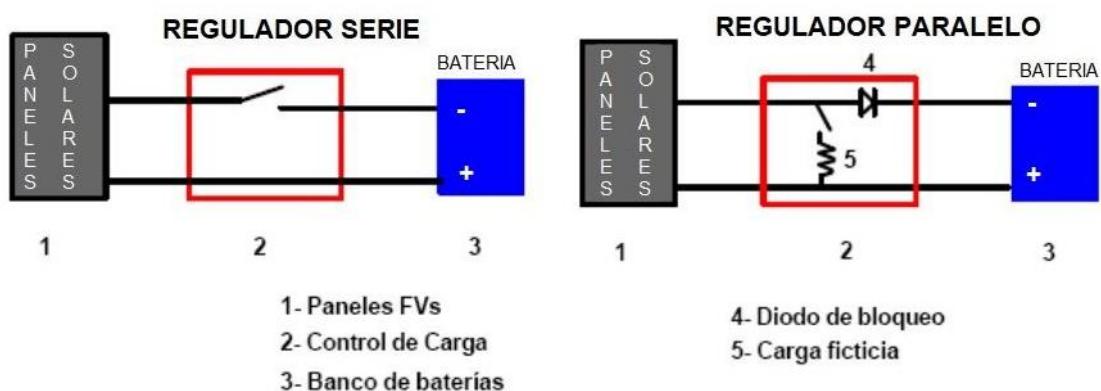


Figura 68: Tipos de reguladores serie y paralelo por <https://www.areatecnologia.com/>

4.4.1 Regulador LTC Electronics SC1024A/U

Este regulador es el que fue utilizado en este proyecto, el mismo se puede ver a continuación:



Figura 69:Regulador de Carga LTC Electronics SC1024A/U, Edición Propia

Es un regulador paralelo que cuenta con las siguientes características:

- 6 bornes de conexión: Entrada del panel solar, entrada de batería y salida de 12V para conexión de carga
- Un puerto USB donde puede programarse el regulador para 12V y 24V.
- Una salida de 12V o 24V tipo Jack
- Indicador de celda solar con 4 estados
 - Encendido: carga completa.
 - Intermitente rápido: carga en progreso.
 - Intermitente lento: carga de flote.
 - Apagado.
- Indicador de batería de 4 estados:
 - Rojo intermitente: batería baja, indicación de que se debe desconectar la carga de salida
 - Rojo: batería al mínimo nivel de carga

- Verde: batería completamente cargada
- Apagado.
- Indicador de carga de salida: solo tiene dos estados
 - Prendido: Indica carga conectada
 - Apagados: Indica que la carga esta desconectada

4.5 Acumuladores de Energía

En un sistema fotovoltaico, un acumulador de energía se refiere a una batería o conjunto de baterías que almacenan la electricidad generada por los paneles solares. Estas baterías son una parte esencial del sistema, ya que permiten que la energía solar generada durante el día se almacene y utilice en momentos en que no hay suficiente luz solar disponible, como durante la noche o en días nublados.

Existen muchos tipos de batería actualmente disponibles en el mercado, las baterías de plomo-antimonio, plomo-calcio, níquel-cadmio, plomo-acido, electrolito de gel, son algunos ejemplos. En este trabajo analizaremos las baterías de gel o de electrolito de gel por ser la usada en el proyecto.

4.5.1 Batería de gel

Las baterías de gel, también conocidas como baterías de electrolito de gel o baterías de gel sellado, son un tipo de batería recargable utilizada en una variedad de aplicaciones, incluyendo sistemas fotovoltaicos, sistemas de respaldo de energía, vehículos eléctricos y más. Estas baterías se caracterizan por su diseño especial que utiliza un electrolito en forma de gel en lugar de un electrolito líquido como en las baterías convencionales de plomo-acido. Aquí hay una explicación más detallada de las baterías de gel:

1. Electrolito de Gel: La característica distintiva de las baterías de gel es el electrolito en forma de gel que se utiliza en lugar del líquido presente en las baterías de plomo-ácido convencionales. El gel es un material espeso y viscoso que inmoviliza el electrolito, lo que evita derrames y fugas, incluso si la batería se instala en diferentes posiciones.

2. Sellado Hermético: Las baterías de gel están selladas herméticamente, lo que significa que no requieren mantenimiento y no es necesario agregar agua destilada para reponer el electrolito, como en algunas baterías de plomo-ácido convencionales. Esto hace que las baterías de gel sean prácticamente libres de mantenimiento.

3. Durabilidad y Vida Útil: Las baterías de gel son conocidas por ser duraderas y tener una vida útil más larga en comparación con las baterías de plomo-ácido convencionales. Pueden soportar ciclos de carga y descarga profundos sin sufrir daños significativos.

4. Seguridad: El electrolito en forma de gel es menos propenso a fugas y derrames, lo que hace que las baterías de gel sean más seguras y fáciles de manejar. También son menos susceptibles a la corrosión.

5. Descarga Lenta: Las baterías de gel son ideales para aplicaciones que requieren una descarga lenta y constante de energía, como sistemas de respaldo de energía, sistemas solares fotovoltaicos y aplicaciones marinas.

6. Menos Capacidad de Corriente: En comparación con las baterías de plomo-ácido convencionales, las baterías de gel tienden a tener una capacidad de corriente menor, lo que significa que no son adecuadas para aplicaciones que requieren corrientes muy altas en un corto período de tiempo.



Figura 70: Batería de electrolito de gel utilizada, Edición Propia

5 Capítulo 5: PROTOCOLO MQTT y BROKER MQTT

En este capítulo, abordaremos en detalle el Protocolo MQTT y el Broker MQTT. Estos elementos son fundamentales para el correcto funcionamiento de la comunicación entre el sistema de tierra firme y la interfaz de visualización de datos que será detallada en el capítulo 9. Además, en un contexto general, el Protocolo MQTT se utiliza ampliamente en aplicaciones de Internet de las cosas (IoT) y sistemas de mensajería para permitir una comunicación eficiente y escalable entre dispositivos distribuidos.

5.1 Protocolo MQTT

El protocolo MQTT (Message Queuing Telemetry Transport) es un protocolo de comunicación ligero y eficiente diseñado para la transmisión de mensajes entre dispositivos en redes con ancho de banda limitado o conexiones poco confiables. MQTT se utiliza comúnmente en aplicaciones de Internet de las cosas (IoT) y en sistemas de mensajería en tiempo real.

Algunas de las características clave de MQTT son:

- Ligero: MQTT es un protocolo de comunicación ligero que minimiza la sobrecarga de datos en los mensajes y en el intercambio de información entre dispositivos. Esto lo hace adecuado para aplicaciones con ancho de banda limitado o conexiones de red poco confiables.
- Basado en el modelo de publicación/suscripción: MQTT sigue un modelo de publicación/suscripción, donde los dispositivos pueden publicar mensajes en "temas" (topics) y suscribirse a temas específicos para recibir los mensajes relevantes. Esto permite una comunicación flexible y escalable entre dispositivos.
- Calidad de servicio (QoS): MQTT admite varios niveles de calidad de servicio para garantizar la entrega confiable de mensajes. Los niveles de QoS van desde 0 (entrega sin garantía) hasta 2 (entrega garantizada).

- Retención de mensajes: MQTT permite que los mensajes se retengan en el servidor hasta que un suscriptor se conecte y los reciba. Esto es útil para asegurarse de que los suscriptores no se pierdan mensajes importantes cuando están desconectados.
- Amplia adopción en IoT: MQTT es ampliamente utilizado en aplicaciones de Internet de las cosas debido a su eficiencia y flexibilidad. Permite la comunicación eficaz entre una amplia variedad de dispositivos IoT, desde sensores y actuadores hasta servidores y aplicaciones en la nube.

5.2 Broker MQTT

Un broker MQTT es un componente fundamental en el protocolo MQTT. Es un servidor o servicio centralizado que actúa como intermediario entre los dispositivos que publican información (publicadores) y los dispositivos que desean recibir esa información (suscriptores) en una red MQTT. El broker MQTT es responsable de enrutar los mensajes publicados a los suscriptores adecuados, lo que permite una comunicación eficiente y escalable en aplicaciones de Internet de las cosas (IoT) y otros sistemas de mensajería.

Estos funcionan como centro de almacenamiento y organización de la información, que tienen las siguientes funciones:

- Conocer a sus Publicadores y Suscriptores dándoles a cada uno una ID unívoco llamado Token.
- Conocer cuáles son los mensajes que llegan de sus diferentes servidores y guardarlos en los topics a los que estos estan suscriptos.
- Mandar los mensajes a todos los Suscriptores que se encuentren anotados en topics que recibieron mensajes.
- Administrar la información de modo correcto entre clientes y suscriptores.

Estas funciones del Broker hacen que:

- Los dispositivos no se comuniquen unos con otros sino a través del broker.

- Los dispositivos desconozcan la existencia de otros en la misma red.
- Tantos los Suscriptores como los Publicadores puedan conectarse a uno a más topics del Broker.

El Broker MQTT cumple la función de ser el elemento central de la comunicación MQTT y actúa como una pasarela de datos. Cuando se publica un dato en un tópico específico, el Broker recibe ese mensaje y lo envía a todos los suscriptores que están registrados en ese mismo tópico. Esto permite una distribución eficiente de la información entre los dispositivos y sistemas que utilizan MQTT para la comunicación, asegurando que los datos lleguen a quienes están interesados en ellos.

6 Capítulo 6: SISTEMA DE TIERRA FIRME

En este capítulo, nos sumergiremos en la comprensión del sistema de tierra firme y el software que impulsa su correcto funcionamiento. Sin embargo, antes de adentrarnos en estos conceptos, es esencial adquirir una comprensión sólida del funcionamiento del Gateway. Una vez que hayamos abordado de manera exhaustiva este componente crucial, procederemos a explorar en detalle el software que opera en el sistema de tierra firme. Esta secuencia de aprendizaje nos permitirá apreciar mejor cómo se integran los distintos elementos para garantizar un control efectivo del vehículo en su entorno acuático.

6.1 Gateway

Para el sistema de tierra firme se desarrolló un Gateway LoRa hecho con un módulo LoRa y un ESP32 que es un dispositivo que actúa como un puente de comunicación entre dispositivos LoRa de baja potencia y una red de conectividad más amplia, como Internet. Para entender mejor esta configuración, a continuación, se detallarán sus partes paso por paso:

- Módulo LoRa: Sus características fueron definidas en el capítulo 2.
- ESP32: Sus características fueron definidas en el capítulo 2.
- Gateway LoRa: El ESP32 se utiliza como la unidad central de procesamiento de un Gateway LoRa. La función principal de este Gateway es recibir datos de dispositivos LoRa y transmitirlos a una red de mayor alcance, como Internet. Su funcionamiento se detalla a continuación:
 - a. Recepción de datos: El módulo LoRa recibe datos desde dispositivos LoRa cercanos. Estos dispositivos pueden ser sensores, actuadores u otros dispositivos que utilicen LoRa para la comunicación.
 - b. Procesamiento de datos: El ESP32 se encarga de procesar los datos recibidos, lo que puede incluir la validación, el almacenamiento temporal y la conversión de datos según sea necesario.
 - c. Transmisión de datos: Una vez que los datos se han procesado, el ESP32 utiliza su conectividad Wi-Fi o Ethernet para enviar estos datos a un servidor o plataforma en la nube.

nube a través de Internet. Esto permite la visualización, el análisis y el control remoto de los dispositivos LoRa.

El Gateway LoRa está configurado para interactuar con servicios en la nube, bases de datos o aplicaciones específicas.

Definido todo esto, la tarea del Gateway LoRa será vincularse a un servidor mediante el protocolo MQTT con una aplicación diseñada en Node-RED (capítulo 9) desde la cual se administrarán y visualizarán los datos del vehículo y a su vez se realizarán acciones de control.

Dentro de la estructura de comunicación mencionada, el vehículo será un nodo, en donde el Gateway cumple la función de ser el puente entre el nodo y el servidor.

Antes de sumergirnos en la descripción de la composición del software del sistema de tierra firme y del vehículo, es importante establecer una base sólida al comprender qué es el software y cuál fue la tecnología utilizada para desarrollar este software.

6.2 Software

Para el diseño del Software de control tanto para el módulo ESP32 que se encuentra en tierra firme y el del vehículo, se utilizó el entorno de desarrollo Arduino cuyo lenguaje de programación es C++ y además es un entorno con el cual estábamos familiarizados, pero la desventaja que tiene es que no se puede sacar el máximo provecho del módulo ESP32, pero sirvió para la aplicación que se plantea en este proyecto.

Sistema Operativo (SO)

Un sistema operativo es un software fundamental que actúa como intermediario entre el hardware de una computadora y las aplicaciones de usuario. Su función principal es administrar los recursos de hardware y proporcionar servicios esenciales para que las aplicaciones se ejecuten de manera eficiente y sin problemas. Esto incluye, por ejemplo, la gestión de la memoria, la administración de procesos, el control de dispositivos, etc.

Sistema Operativo en Tiempo Real (RTOS)

Este tipo especializado de sistema operativo está diseñado para aplicaciones que requieren respuestas rápidas y predecibles a eventos que suceden en tiempo real. A diferencia de los sistemas operativos convencionales, donde la prioridad suele ser la eficiencia general del sistema, un RTOS se centra en garantizar que las tareas críticas se

completan dentro de plazos estrictos y predecibles. Se suele utilizar mucho en sistemas como controladores industriales, sistemas de control de vuelo, dispositivos médicos, etc.

FreeRTOS

De sus siglas en inglés “Real-Time Operating System” es un sistema operativo en tiempo real de código abierto que se utiliza comúnmente en sistemas embebidos y aplicaciones en tiempo real. Proporciona características esenciales de un RTOS, pero se esfuerza para mantener poco uso de los recursos para que sea adecuado para sistemas con restricción de los mismos. Se utiliza mucho en aplicaciones para el Internet de las cosas (IoT), además de las mencionadas anteriormente, algunas de sus características son:

- Es soportado por más de 30 arquitecturas de microcontroladores diferentes.
- Soportado por más de 20 compiladores.
- Portable: el código es fácilmente portable de un microcontrolador a otro.
- Mayormente escrito en código C.
- El kernel compilado solo ocupa 6KB de memoria flash y unos pocos cientos de bytes de RAM.
- Permite crear tareas y establecerles una prioridad, también permite asignar el núcleo que la ejecutará, lo cual es una característica importante a la hora de utilizar un ESP32, ya que posee dos núcleos.

6.3 Software del Servidor

El software que se diseñó para el ESP32 del servidor, está formado por un conjunto de archivos, dentro de los cuales se encuentran los siguientes:

- `ESP32_Servidor.ino`: es el archivo principal en un proyecto de Arduino, este contiene el código fuente principal del programa que se ejecutara en el microcontrolador, dentro de él se encuentra la función “`setup ()`” y las funciones “`loop ()`” del núcleo 1.

- Config.h: Este es un archivo de encabezado en el cual se realizan las configuraciones necesarias para el correcto funcionamiento del programa principal, como la inicialización de la comunicación MQTT, la inicialización del puerto SPI para el módulo LoRa, la inicialización de la comunicación Wifi.
- Funciones.h: En este archivo se configuran las funciones que serán utilizadas en el código principal para llevar a cabo las distintas tareas, para un correcto funcionamiento del servidor.
- Variables.h: Se decidió crear este archivo para declarar todas las variables que deberán ser inicializadas de manera global, a modo de no contaminar el archivo principal del programa, y se puede recurrir de manera más sencilla y ordenada ante una posible modificación.
- Instancias.h: Este archivo se realizó para la declaración de pines de los distintos componentes externos y la declaración de objetos que hacen referencia a las diferentes librerías.

En el caso del programa del servidor no hizo falta la utilización de ambos núcleos del ESP32, debido a que las tareas realizadas por el mismo bastan con uno solo de los núcleos.

Dentro del “setup()” del programa se procede a inicializar la comunicación MQTT, la comunicación Lora, suscribirse a los diferentes tópicos, y establecer la conexión Wifi.

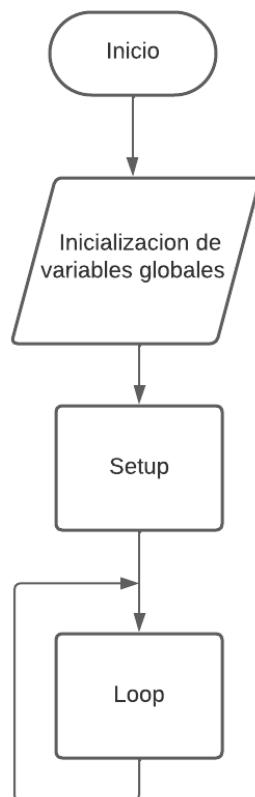


Figura 71: Diagrama de Flujo del Programa Principal, Edición Propia

Las funciones que se realizan en el “loop()” del programa principal son las siguientes:

```

void loop() {

  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  recepcionLora();
}

```

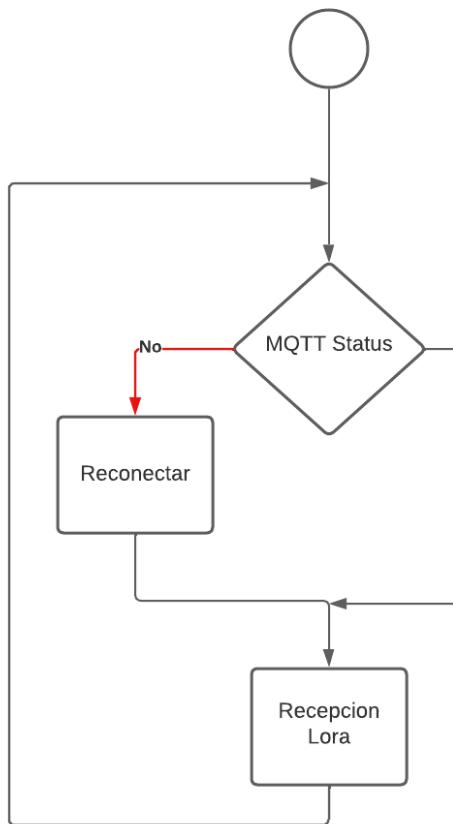


Figura 72: Diagrama de Flujo de la Función Loop, Edición Propia

`!client.connected()`: Esta línea verifica si la variable `client` (probablemente una instancia de la biblioteca MQTT o similar) no está conectada al servidor. La función `connected()` generalmente devuelve `true` si la conexión está establecida y `false` si no lo está.

`reconnect()`: Si la condición `!client.connected()` es verdadera, se llama a la función `reconnect()`. Esto implica que, si no se encuentra conectado al servidor, el código intentará volver a conectarse al servidor utilizando la función `reconnect()`.

`client.loop()`: Despues de verificar la conexión y, si es necesario, intentar reconectarse, el programa entra en el bucle `client.loop()`. Este bucle es típico en bibliotecas como MQTT y se utiliza para mantener la comunicación con el servidor y procesar mensajes entrantes.

`receptionLora ()`: Esta es una función que se diseñó de manera particular para recibir los datos por medio de la comunicación LoRa con el único nodo (vehículo). A fines de

sintetizar esta función, se hizo que todos los mensajes provenientes del vehículo tengan una estructura determinada:

“encabezado: dato”

Donde “encabezado” es el nombre del dato que recibirá el Gateway y “dato” es el dato en sí. Esta función está configurada para recibir mensajes con encabezados, dentro de los cuales se encuentran todos los datos necesarios cuyo nombre ya está establecido, de tal forma que al recibir los mensajes LoRa, el programa filtra los datos contenidos en los mensajes, eliminando todo hasta encontrar “.” y guarda cada dato dentro de una variable diferente la cual posteriormente será publicada en un tópico MQTT según corresponda.

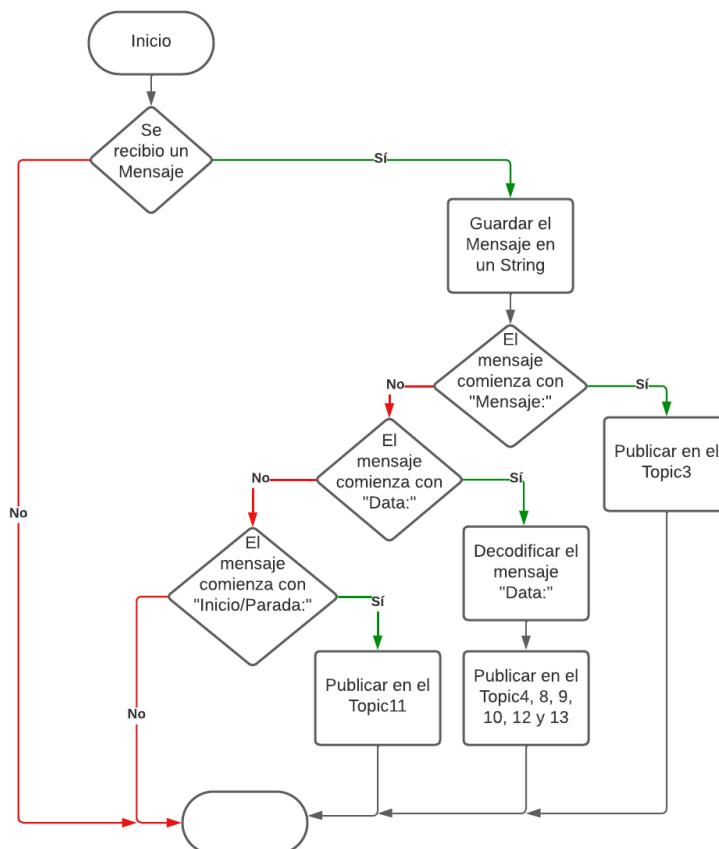


Figura 73: Diagrama de Flujo de la Función Recepción Lora, Edición Propia

Tabla 1: Correspondencia de Tópicos con Encabezados en función “recepccionLora”

Tópico	Encabezado
mensajes	Mensajes:
indicador	Inicio/Parada:
brujula	Data:
haversine	Data:
s_bateria_i	Data:
s_bateria_v	Data:
coor_b	Data:

Otra de las funciones principales del programa es la función “callback”, la cual se pone en funcionamiento cuando se recibe un mensaje MQTT. Los argumentos de esta función son, el tópico con el cual se publicó el mensaje, el payload o el contenido del mensaje y la longitud del mensaje.

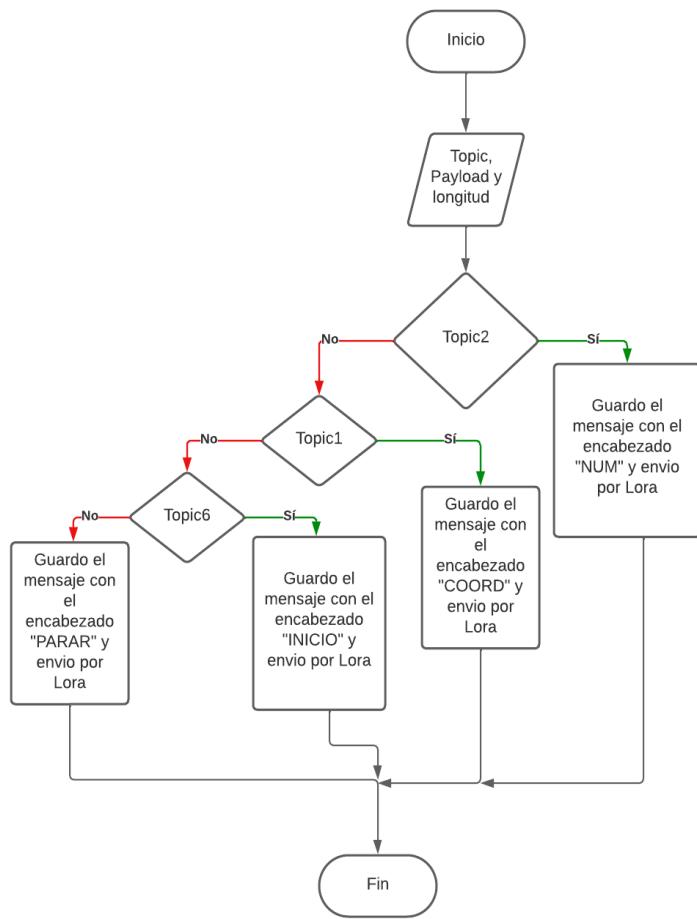


Figura 74: Diagrama de Flujo de la Función Callback, Edición Propia

Como se mencionó en el punto anterior cada tópico tiene un propósito distinto dentro de la estructura de control, por lo cual es importante reconocer bajo qué tópico se publicó el mensaje para poder desarrollar una acción de control consecuente. Por esto, se decidió utilizar la misma estructura de encabezados-datos para que el vehículo pueda diferenciar los comandos que se le envían desde el servidor.

Tabla 2: Correspondencia de Tópicos con Encabezados en función “callback”

Tópico	Encabezado
coord_dir	COORD:
cant_puntos_tray	NUM:
inicio	INICIO:
parar	PARAR:

De esta forma una vez concatenados los mensajes recibidos con sus respectivos encabezados se procede a enviar el mensaje por medio de la comunicación LoRa hacia el vehículo.

7 Capítulo 7: SISTEMA DE NAVEGACION

En este capítulo, se presentarán en detalle los conceptos esenciales necesarios para comprender el funcionamiento del software del sistema de navegación empleado para garantizar el adecuado desempeño del vehículo acuático autónomo. Entre estos conceptos, se destacan el sistema de posicionamiento global (GPS), la fórmula de Haversine y la fórmula del azimut. Estos elementos son fundamentales para la correcta orientación, seguimiento y control del vehículo durante sus operaciones en entornos acuáticos.

Ref[<https://www.tecbolivia.com/index.php/articulos-y-tutoriales-microcontroladores/100-carro-robot-guiado-por-gps>]

7.1 Sistema de Posicionamiento Global

“El Sistema de Posicionamiento Global (GPS) es un sistema de radionavegación de los Estados Unidos de América, basado en el espacio, que proporciona servicios fiables de posicionamiento, navegación, y cronometría gratuita e ininterrumpidamente a usuarios civiles en todo el mundo. A todo el que cuente con un receptor del GPS, el sistema le proporcionará su localización y la hora exacta en cualesquiera condiciones atmosféricas, de día o de noche, en cualquier lugar del mundo y sin límite al número de usuarios simultáneos”

Ref[<https://www.gps.gov/spanish.php>]

7.1.1 Componentes del Sistema GPS

Son tres los componentes fundamentales de este sistema:

1. El segmento espacial: está conformado por 24 satélites que giran en unas direcciones prefijadas a 20200 km de la superficie terrestre cubriendo todo el planeta para poder brindar el servicio de posicionamiento eficientemente. La repartición de satélites se hace en 6 planos orbitales de 4 unidades cada uno. La información proporcionada por los satélites se hace las 24 hs del día sin importar las condiciones de tiempo. Para llevar a cabo la transmisión se emiten dos señales de radiofrecuencia, L1 con una frecuencia de 1575.43 MHz y L2 con frecuencia de 1227.6 MHz. La señal L1 se modula con dos códigos de ruido pseudoaleatorios (Pseudo Radio Noise, PRN). Para la

mayoría de las aplicaciones que no son de las fuerzas armadas, el código que se utiliza es el C/A (Coarse/Adquisition) conocido como servicio Estándar de Posicionamiento.

Para poder estar funcionando las 24hs del día, los satélites se valen de energía eléctrica obtenida de sus paneles solares, los cuales llevan montados lateralmente y tambien de una pila de baterías de Ni-cad (Giménez Rodríguez & Ros Bernabeu, 2010).

Otro componente importante de los satélites es su reloj atómico, estos brindan la hora con un grado de precisión mucho mayor que los relojes usados por los dispositivos móviles, tienen errores de precisión del orden de los nanosegundos.

2. El segmento de control: se refiere a una serie de estaciones terrestres que rastrean los satélites GPS, monitorean sus transmisiones, realizan análisis de información y envían comandos y datos a la constelación. (Oficina Nacional de Coordinación para Posicionamiento, Navegación y Temporización (PNT),2020). Las instalaciones de control pueden dividirse en 4 grupos: Estación de control maestra, Estación de control alternativa, Antenas de comando, dieciséis sitios de monitoreo.
3. Segmento del usuario: Esto hace referencia a los dispositivos y módulos que, con diferentes prestaciones, logran hacer uso de la tecnología satelital para brindar posición, velocidad y tiempo al usuario.

7.1.2 Funcionamiento del GPS

El sistema GPS funciona a través de una técnica denominada trilateración, que permite calcular la elevación, ubicación y velocidad. La trilateración reúne la señal de diversos satélites que envían información de ubicación, midiendo la distancia entre un objeto o persona y otro objeto o persona.

Los distintos satélites que orbitan la Tierra envían señales para que un dispositivo con sistema GPS, ubicado en la superficie terrestre, pueda interpretarlas. Para que esto suceda, un dispositivo GPS tiene que leer las señales de, al menos, cuatro satélites distintos. Los satélites que componen la red completan dos veces por día la órbita a la Tierra, enviando señal e información. Un GPS puede leer las señales de seis o más satélites.

Cuando un satélite envía una señal, se crea un círculo con un radio medido desde el GPS al satélite. Al agregar un segundo satélite, se crea un segundo círculo, y la ubicación del objeto o persona solicitado se reduce a uno de los dos puntos donde los círculos confluyen. Si agregamos un tercer satélite, la ubicación se puede determinar: se encuentra en la intersección de los tres círculos.

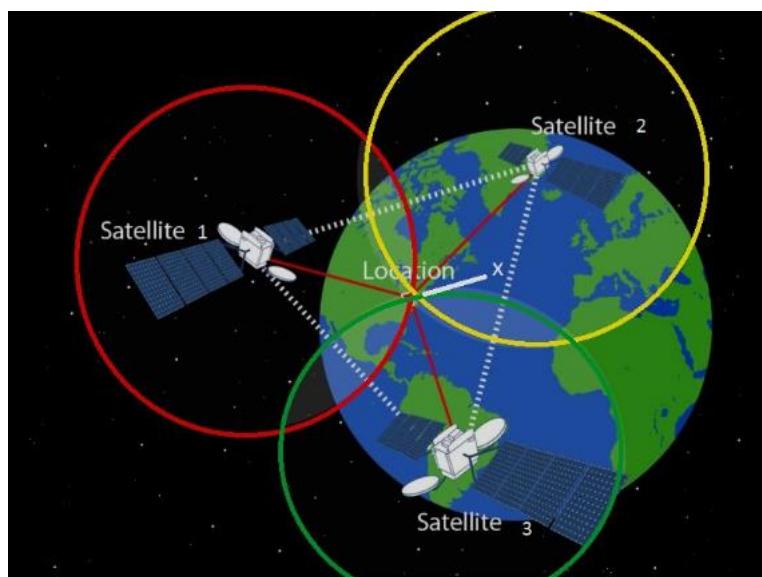


Figura 75: Funcionamiento del GPS por <https://www.xataka.com/>, 2018

A pesar de la precisión en las medidas del GPS, estos incurren en errores que provienen de diferentes fuentes. La principal de estas es el satélite mismo. Los errores propios del satélite están asociados a su forma geométrica. Se definen, por tanto, unos parámetros que se llaman DPG (Dilación en la precisión geométrica, por sus siglas en inglés). En ellos se reflejan los errores surgidos de la forma geométrica del satélite que afecta la estimación de posición y tiempo. Podemos encontrar:

- PDOP (Dilación de precisión en posición, de sus siglas en inglés).
- HDOP (Dilación de precisión en el plano horizontal, de sus siglas en inglés).
- VDOP (Dilación de precisión vertical, de sus siglas en inglés).

- TDOP (Dilación de precisión de tiempo, de sus siglas en inglés).

7.1.3 Códigos NMEA

El "código NMEA" a menudo se refiere a las sentencias o tramas NMEA que son parte del estándar de comunicación NMEA (National Marine Electronics Association) utilizado en la navegación marítima y sistemas de posicionamiento global (GPS). No es un código en el sentido de un programa de software, sino más bien un conjunto de sentencias o tramas de datos que siguen un formato específico.

Cada sentencia o trama NMEA tiene un formato particular y contiene información sobre diversos aspectos de la navegación, como la posición, velocidad, rumbo y otros datos relacionados con la ubicación y el tiempo. Estas sentencias NMEA son utilizadas por dispositivos electrónicos marinos, como receptores GPS, sistemas de navegación y otros equipos de navegación, para comunicarse entre sí de manera estandarizada.

Algunos ejemplos de sentencias NMEA comunes incluyen:

- \$GPGGA: Proporciona información sobre la posición y la hora en formato UTC.
- \$GPRMC: Contiene información sobre la posición, velocidad y rumbo.
- \$GPGLL: Suministra información de latitud y longitud.
- \$GPVTG: Ofrece información sobre la velocidad y el rumbo sobre el suelo.

Cada sentencia NMEA comienza con el símbolo "\$" seguido de una serie de caracteres que identifican el tipo de sentencia y luego está seguida por datos específicos separados por comas.

Por ejemplo, la sentencia:

\$GPGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.9,M,,*47

Contiene información sobre la posición GPS en formato GGA, incluyendo la hora, la latitud y longitud y otros datos relacionados.

7.2 Formula de Haversine

"La fórmula del semiverseno es una importante ecuación para la navegación astronómica, en cuanto al cálculo distancia de círculo máximo entre dos puntos de un globo sabiendo sus longitud y latitud. Es un caso especial de una fórmula más general de trigonometría esférica, la ley de los semiversenos, que relaciona los lados y los ángulos de "triángulos esféricos"

Ref.[<https://es-academic.com/dic.nsf/eswiki/1288404>]

Dicha formula se utiliza para obtener la distancia entre el vehículo y los puntos establecidos en la ruta a seguir.

$$\text{hav}\left(\frac{d}{r}\right) = \text{hav}(\varphi_2 - \varphi_1) + \cos(\varphi_1) \cdot \cos(\varphi_2) \cdot \text{hav}(\lambda_2 - \lambda_1) \quad (1)$$

$$\text{hav}(\theta) = \sin^2\left(\frac{\theta}{2}\right) = \frac{1 - \cos(\theta)}{2} \quad (2)$$

$$a = \sin^2\left(\frac{\Delta\varphi}{2}\right) + \cos(\varphi_1) \cdot \cos(\varphi_2) \cdot \sin^2\left(\frac{\Delta\lambda}{2}\right) \quad (3)$$

$$c = 2 \cdot \arctan 2(\sqrt{a}, \sqrt{1-a}) \quad (4)$$

$$d = r \cdot c \quad (5)$$

φ_1 : point 1 latitude,	λ_1 : point 1 longitude
φ_2 : point 2 latitude,	λ_2 : point 2 longitude
d : arc (distance),	r : Earth's radius
$\Delta\varphi = \varphi_2 - \varphi_1;$	$\Delta\lambda = \lambda_2 - \lambda_1$

Figura 76: Formula de Haversine utilizada en el software de navegación

7.3 Formula del Azimut

*"El azimut de una línea es el ángulo horizontal medido **en el sentido de las manecillas del reloj** a partir de un meridiano de referencia. Lo más usual es medir el azimut desde el Norte (sea verdadero, magnético o arbitrario), pero en ocasiones se usa el Sur como referencia"*

Ref [<https://doblevia.wordpress.com/2007/07/25/direccion-de-una-linea-rumbo-y-azimut/>].

Esta fórmula es necesaria para obtener la orientación del vehículo con respecto al Norte Geográfico, y así se pueda dirigir de forma correcta hacia el punto establecido en la ruta.

$$\theta = \text{atan2}(\sin \Delta\lambda \cdot \cos \varphi_2, \cos \varphi_1 \cdot \sin \varphi_2 - \sin \varphi_1 \cdot \cos \varphi_2 \cdot \cos \Delta\lambda) \quad (6)$$

φ_1 : point 1 latitude, φ_2 : point 2 latitude

θ : waypoint angle or azimuth

$$\Delta\lambda = \lambda_1 - \lambda_2$$

Figura 77: Formula del Azimut utilizada en el software de navegación

7.4 Software del sistema de navegación

El software que se diseñó para el ESP32 del vehículo, está formado por un conjunto de archivos, dentro de los cuales se encuentran los siguientes:

- `ESP32_balsa.ino`: es el archivo principal en un proyecto de Arduino, este contiene el código fuente principal del programa que se ejecutara en el microcontrolador, dentro de él se encuentra la función “`setup ()`” y los funciones “`loop2()`” del núcleo 0 y “`loop ()`” del núcleo 1.
- `Config.h`: Este es un archivo de encabezado en el cual se realizan las configuraciones necesarias para el correcto funcionamiento del programa principal, como la inicialización del puerto serial para la lectura del GPS, la inicialización del puerto SPI para el módulo LoRa, la inicialización del módulo GY-271(brújula), definición de pines digitales para el correcto funcionamiento del módulo L298N y el sensor JSN-SR04T.
- `Funciones.h`: En este archivo se configuran las funciones que serán utilizadas en el código principal para llevar a cabo las distintas tareas, para un correcto funcionamiento del vehículo.
- `Variables.h`: Se decidió crear este archivo para declarar todas las variables que deberán ser inicializadas de manera global, a modo de no contaminar el archivo

principal del programa, y se puede recurrir de manera más sencilla y ordenada ante una posible modificación.

- Instancias.h: Este archivo se realizó para la declaración de pines de los distintos componentes externos y la declaración de objetos que hacen referencia a las diferentes librerías.

Las funciones definidas en el programa principal mencionado anteriormente están formadas por dos tareas, que corren en paralelo mediante la utilización de ambos núcleos del ESP32 (núcleo 0 y 1), todo esto es posible gracias al FreeRTOS, en la siguiente figura se puede observar el flujo del programa principal.

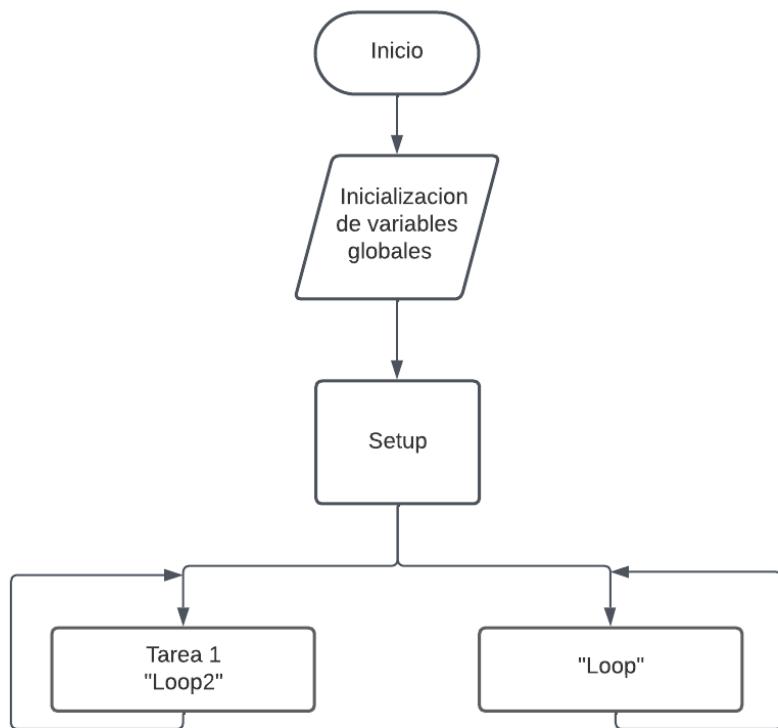


Figura 78: Diagrama de Flujo del Programa Principal, Edición Propia

El núcleo 0 va a ser el encargado de la ejecución de diferentes funciones como, Comunicación LoRa, lectura y procesamiento de los datos de los módulos (GY-271, GPS NEO 6-M), cálculos (azimut y haversine), lectura y cálculos del sensor de tensión y corriente (apéndice 10.1). La primera función mantiene una comunicación continua con el módulo

RFM95 por medio del protocolo SPI. Esta función es la que se encarga de detectar los mensajes enviado desde tierra firme, como son los puntos de trayectoria, las coordenadas de cada punto, el inicio/parada del vehículo y el envío de mensajes de confirmación a tierra firme. La segunda función es la que se encarga de leer y procesar los datos de los módulos para realizar de forma correcta la siguiente función, la de cálculos, que se encarga de determinar los valores de azimut y haversine con los datos proporcionados por la función anterior.

El núcleo 1 solo ejecuta la tarea del movimiento automático del vehículo. Esta tarea es la que realiza todo el control tanto del servo motor, como la velocidad y el sentido de giro del motor DC, basado en la información que recolecto y calculo el núcleo 0.

Tarea Núcleo 0 “loop2”

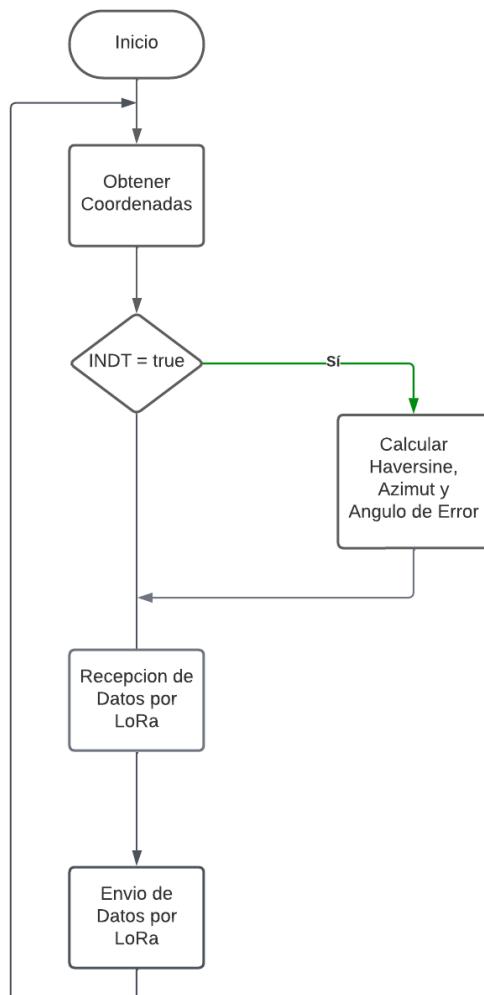


Figura 79: Diagrama de Flujo del Núcleo 0, Edición Propia

Lo primero que se debe hacer antes de realizar el código de la tarea es crear la misma. Lo cual se hace de la siguiente manera:

```
1  xTaskCreatePinnedToCore(  
2  |  loop2,      //nombre del loop creado  
3  |  "loop2",    //nombre de la función creada  
4  |  100000,     //Tamaño de la pila  
5  |  NULL,       //Parametro casi siempre nulo  
6  |  1,          //Prioridad de la tarea  
7  |  &Sensado,   //nombre de la tarea  
8  |  0);        //Núcleo donde se ejecuta
```

Figura 80: Código para la creación de una Tarea en un núcleo determinado en el IDE de Arduino Edición Propia

En la línea 1 tenemos el nombre de la función que crea una tarea. Entre paréntesis, y separando por comas se deben ingresar 7 argumentos o parámetros.

1. El primero argumento es la función de la tarea que se desea ejecutar. Es la función que va a contener el código que será ejecutado cuando se inicie la tarea.
2. El segundo es el nombre de la tarea. Es un identificador legible que se usa para identificar la tarea. Es útil especialmente cuando se tiene múltiples tareas en un programa.
3. El tercer argumento es el tamaño de pila de la tarea en palabras. La pila es donde se almacenan las variables locales y la información de la tarea.
4. El cuarto argumento sirve para pasar parámetros a la tarea.
5. El quinto argumento es la prioridad de la tarea. El sistema de planificación asigna tiempo de CPU en función de esta prioridad. Un valor más alto significa una prioridad más alta.
6. El sexto argumento es el identificador de la tarea.
7. El séptimo argumento se determina el núcleo donde se va a realizar la tarea.

El ESP32 tiene dos núcleos 0 y 1.

El código que se realizara dentro de la tarea creada, debe escribirse dentro de la función a la que se hizo referencia en el primer argumento. En este caso se le dio el nombre de “loop2”. Todas las funciones que se van a realizar dentro de “loop2” se harán en un bucle infinito, es decir, es un código que se ejecuta siempre, una y otra vez a lo largo del tiempo.

Otra definición importante que se debe realizar es la siguiente:

21 TaskHandle_t Sensado;

Esta línea de código, declara una variable llamada “Sensado”, del tipo “TaskHandle_t”. Este tipo de variable se utiliza para identificar una tarea que se crea a través de FreeRTOS y se utiliza para representar o identificar una tarea que será ejecutada dentro de un determinado núcleo.

La estructura de la función que representa una tarea es la siguiente:

```
1 void loop2 (void *parameter){ //Definicion de la funcion de la tarea
2
3     for (;;){ //Definicion e inicio del bucle infinito
4
5         //Escrivira del codigo que realiza la tarea
6
7     } //Fin del bucle infinito
8
9
10 } //Terminacion de la tarea
11
```

Figura 81: Estructura de tarea para bucle infinito, Edición Propia

En el fragmento de código proporcionado se define la función “loop2” que representa a la tarea (Línea 1).

En la línea 3 se crea un bucle infinito mediante la función “for()”. En lenguaje C una función “for()” sin ningún parámetro en su interior se comporta como tal. Dentro de este bucle se debe realizar las funciones que debe ejecutar la tarea, luego con “{}” se determina el comienzo y el fin de cada función.

Una vez creado el bucle infinito en la tarea, se van a realizar las siguientes funciones dentro del mismo:

- Obtener coordenadas del módulo GPS NEO 6M.

- Realizar cálculos de distancia entre dos puntos de la tierra (Haversine), además del cálculo necesario para obtener el azimut.
- Obtener el valor del error de ángulo utilizando la orientación del vehículo obtenida mediante el módulo GY-271 y el azimut calculado anteriormente.
- Enviar mensajes de confirmación a tierra firme cada vez que se recibe un mensaje desde allí.
- Enviar datos de: Las coordenadas del vehículo, la orientación con respecto al norte geográfico, la distancia entre el vehículo y el punto a donde debe desplazarse, confirmación de mensajes recibidos y valores de tensión y corriente medidos por el sensor.

Obtención de coordenadas geográficas del Vehículo

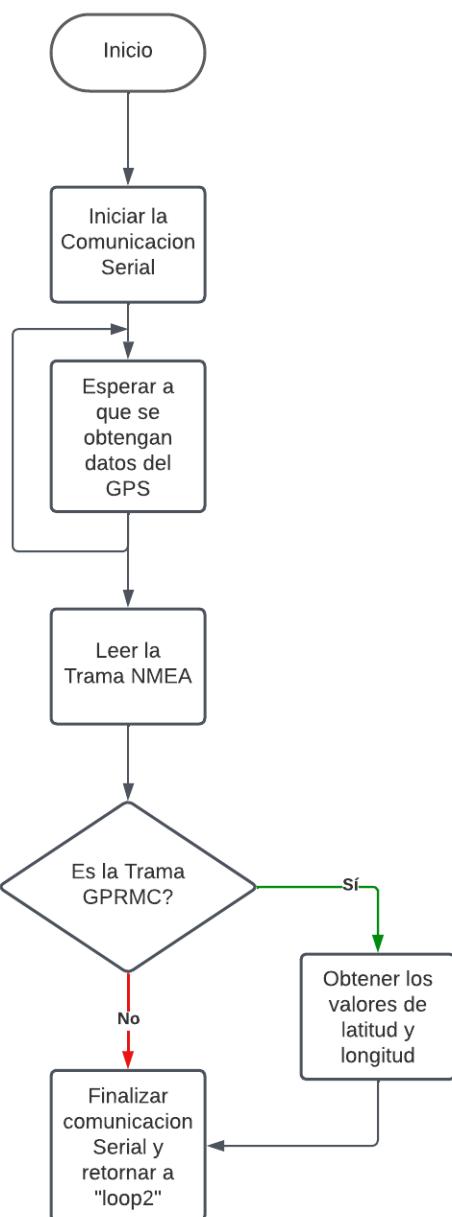


Figura 82: Diagrama de Flujo de la Función coord_GPS, Edición Propia

```

void loop2 (void *parameter){
    for(;;){
        ///////////////////////////////////////////////////
        coord_GPS();
    }
}

```

Figura 83: Función coord_GPS dentro del bucle infinito de la tarea “loop2”, Edición Propia

La función coord_GPS(); que se encuentra dentro de la tarea “loop2” es la que se realizó para poder obtener los valores de Latitud y Longitud del vehículo. Esta información es importante porque nos permite saber en cada momento el punto en el cual se encuentra el vehículo, además de ser necesario para poder calcular la distancia entre su posición inicial y la posición final a la cual debe llegar.

Los datos se obtienen mediante la trama \$GPRMC del conjunto de tramas NMEA que nos brinda el módulo GPS NEO 6M, a partir de esta trama obtenemos la posición del vehículo, además con esta trama se obtienen los datos de la hora (horas: minutos: segundos), de la fecha (día/mes/año), la velocidad sobre el suelo en nudos, el rumbo en grados verdaderos. Pero estos datos tienen el inconveniente que se encuentran sometidos a un error por lo tanto no se los utiliza y solo se obtienen las coordenadas del vehículo.

Cálculo de distancia entre dos puntos de la tierra y del azimut.

A continuación, se muestra el llamado a la función haversine para obtener la distancia entre dos puntos:

Llamado a la función:

```
distancia=haversine(coordenadas[0],coordenadas[1],trayectoria[indTCopia].lat,
trayectoria[indTCopia].lon);
```

Tener en cuenta la figura 76 que se dio en la explicación de la fórmula de haversine, donde:

$$\varphi_1 = \text{latitud del vehículo} = \text{coordenadas}[0]$$

$$\varphi_2 = \text{longitud del vehículo} = \text{coordenadas}[1]$$

$$\lambda_1 = \text{latitud del punto de trayectoria} = \text{trayectoria}[indTCopia].lat$$

$$\lambda_2 = \text{longitud del punto de trayectoria} = \text{trayectoria}[indTCopia].lon$$

$$r = \text{radio de la tierra en km}$$

Cuando se hace referencia a **coordenadas**, nos referimos a un vector con datos de tipo “double” de dos componentes en el cual se almacenan las coordenadas actuales del vehículo.

Cuando hacemos referencia a **trayectoria**, nos referimos a un vector con datos del tipo “coor”, donde “coor” es una estructura la cual se forma con dos valores del tipo “double”, en este caso se utiliza para establecer un dato el cual contiene dos atributos llamados “lat” y “lon” los cuales corresponden a la latitud y longitud respectivamente. Este vector contendrá como máximo diez datos de tipo “coor”, lo cual significa que se puede tener como máximo diez puntos para establecer una trayectoria. En la función “haversine” definida anteriormente, para poder utilizar los valores contenidos en esta estructura debemos utilizar el nombre del vector, la posición del valor buscado dentro del vector y el atributo que buscamos utilizar. Al finalizar el cálculo dentro de la función, esta nos devuelve la distancia en km desde el vehículo hasta el punto objetivo.

Llamado a la función azimut:

```
azimuth = calculoAzimuth(radians(trayectoria[indTCopia].lat),
                           radians(trayectoria[indTCopia].lon), radians (coordenadas[0]),
                           radians(coordenadas[1]));
```

Tener en cuenta la figura 77 que se dio en la definición de la formula del azimut, donde:

$$\varphi_1 = \text{latitud del vehículo} = \text{trayectoria}[indTCopia].lat$$

$$\varphi_2 = \text{longitud del vehículo} = \text{trayectoria}[indTCopia].lon$$

$$\lambda_1 = \text{latitud del punto de trayectoria} = \text{coordenadas}[0]$$

$$\lambda_2 = \text{longitud del punto de trayectoria} = \text{coordenadas}[1]$$

El argumento de la función “calculoAzimuth” utiliza los mismos parámetros que se definieron en la función “haversine”.

Obtención del error de ángulo

Llamado a la función error:

```
error = anguloError(brujula(),azimuth);
```

Función anguloError:

```
double anguloError(double theta1, double theta2) {
    double result = (theta1 + theta2) - 360 * floor((theta1 + theta2)/360);

    if (result > 180) {
        result = 360 - result;
    }
    return result;
}
```

Esta función se desarrolló con el fin de obtener un valor de ángulo que resulta de la diferencia entre la orientación del vehículo (ángulo obtenido por medio de la brújula, que se da en sentido horario desde el norte geográfico) y el azimuth (definido anteriormente en el apéndice 7.3).

Este valor nos indica que tanto esta desviada la dirección del vehículo con respecto a la dirección del objetivo, como se puede observar en la siguiente imagen.

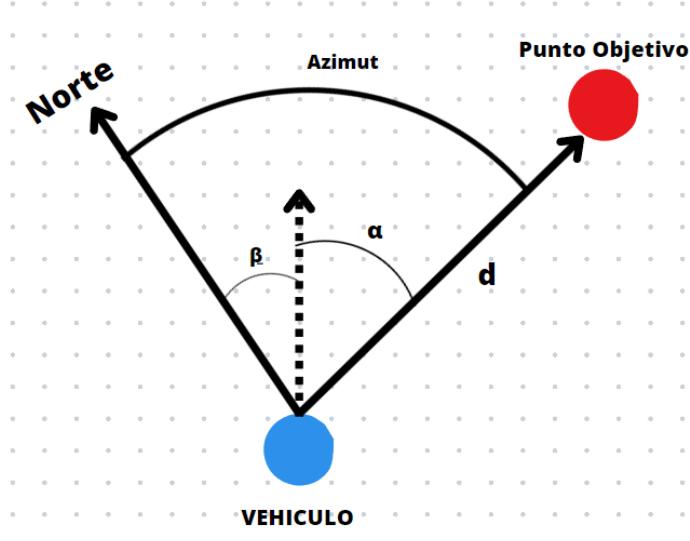


Figura 84: Orientación del vehículo, Azimut y ángulo de error, Edición Propia

La línea a trazos es donde esta “mirando” el vehículo, el ángulo “beta” es el ángulo dado por la brújula, y el azimuth es calculado mediante la formula ya explicada, a partir de estos valores sacamos el ángulo “alfa” que es el “anguloError” mencionado anteriormente. A partir de este resultado se realizan diferentes acciones, que se explicaran con mayor lujo de detalles más adelante.

Recepción de Mensaje mediante la comunicación LoRa

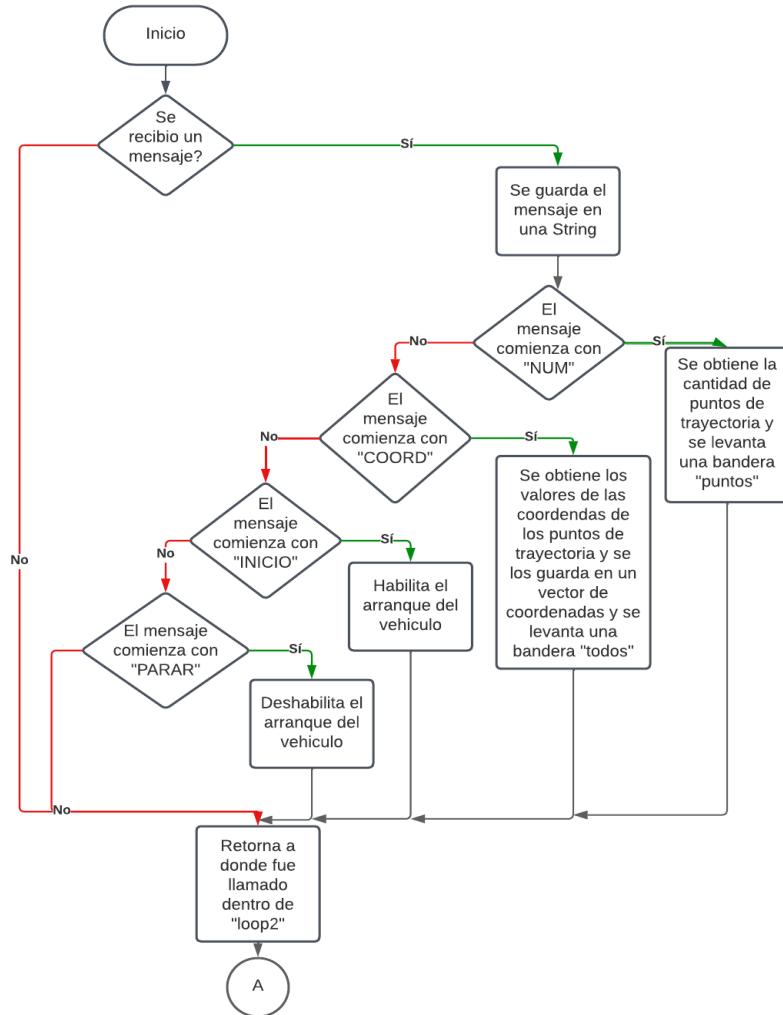


Figura 85: Diagrama de Flujo de la Función Recepción Lora, Edición Propia

Como se vio anteriormente en el software del Gateway los mensajes recibidos tienen un encabezado alusivo al dato que portan. Por lo tanto, de manera homologa, se decodifican los mensajes sustrayendo los encabezados y guardando los datos recibidos en una variable, la cual posteriormente será utilizada para realizar las acciones correspondientes.

Envío de Datos mediante la comunicación LoRa

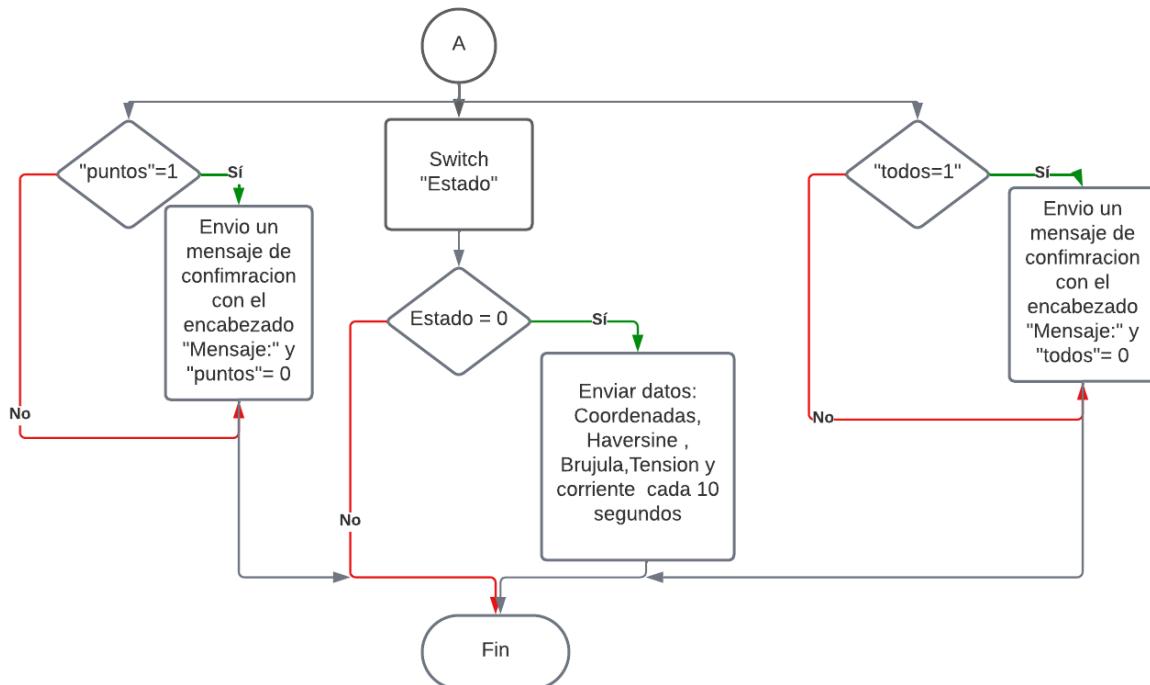


Figura 86: Diagrama de Flujo para el Envío de Datos, Edición Propia

Para poder enviar datos al servidor se implementó una estructura Switch/case con temporización, de tal forma, que los datos se envíen de manera secuencial y con un tiempo prudencial para no crear un conflicto de mensaje a la hora de utilizar la comunicación LoRa. Los mensajes tendrán el mismo formato que se viene utilizando (encabezado: dato), para que estos sean mostrados según corresponda en el dashboard.

Para enviar la confirmación de mensajes se utilizan dos banderas, una para los puntos de trayectoria y otra para las coordenadas de los puntos de trayectoria, es decir, cuando se recibe un mensaje que contiene el encabezado "NUM:" se levanta una bandera que habilita el envío de un mensaje de confirmación al Gateway, lo mismo ocurre cuando se recibe un mensaje con el encabezado "COORD:".

Tarea Núcleo 1 “loop()”

Dentro de este núcleo del ESP32 se realizó la función "marcha" que es la que se encargara de realizar el movimiento del vehículo al punto objetivo establecido, también se implementó el código necesario para las acciones que se deben tomar al detectar un obstáculo con el sensor ultrasónico.

Para poder realizar el movimiento se necesitará de los valores que se obtuvieron en el núcleo 0 (“azimuth”, “brújula” y “error”), mediante una estructura de comparación entre estos valores, se realizan diferentes marchas de acuerdo a la posición relativa del vehículo con respecto al punto objetivo.

Para iniciar debemos definir algunos parámetros, como:

- Valores Obtenidos: Este parámetro es una bandera que se habilita cuando se obtienen todos los datos necesarios del núcleo 0, y permite el comienzo de la función del núcleo 1.
- Inicio: Es una bandera que indica cuando puede iniciar a funcionar el vehículo y cuando debe detenerse. Esta cambiara su valor entre true/false cuando el usuario lo determine, a través de un comando que se enviara desde el servidor.
- Tolerancia: Es un valor constante que define cuando el vehículo llego al objetivo especificado. En este caso se decidió optar por un valor de 10 m o 0,01 km, debido a que el módulo GPS introduce un error de precisión de aproximadamente 5 m.

Cuando todos estos parámetros obtienen valores con los cuales cumplen las condiciones para poder iniciar el funcionamiento del vehículo, se procede a detallar una estructura de control utilizando las lecturas y los cálculos que se obtuvieron en el “loop2()”.

Direccionamiento del Vehículo

Para esta estructura de control se decidió optar por un control proporcional en el cual, mediante la comparación de tres parámetros (azimut, ángulo de error, orientación del vehículo), se realizan diferentes acciones. Estas, fueron definidas a partir de pruebas en las cuales se vio cómo reacciona mecánicamente el vehículo y se concluyó que, en un principio, debe conocerse como está orientado el vehículo con respecto a la dirección del punto geográfico (objetivo) al cual se busca llegar.

De manera general se puede decir, que las acciones que realizara el vehículo son:

- Avance en la dirección del objetivo utilizando como referencia de giro, el ángulo de error.
- Avance con giro total hacia la izquierda o derecha del vehículo.
- Retroceso con giro total hacia la izquierda o derecha del vehículo.

Estas tienen como objetivo, hacer que el vehículo tome la decisión más sencilla para orientarse con dirección al destino.

Para tener una compresión más clara de lo hablado anteriormente se realiza el siguiente grafico:

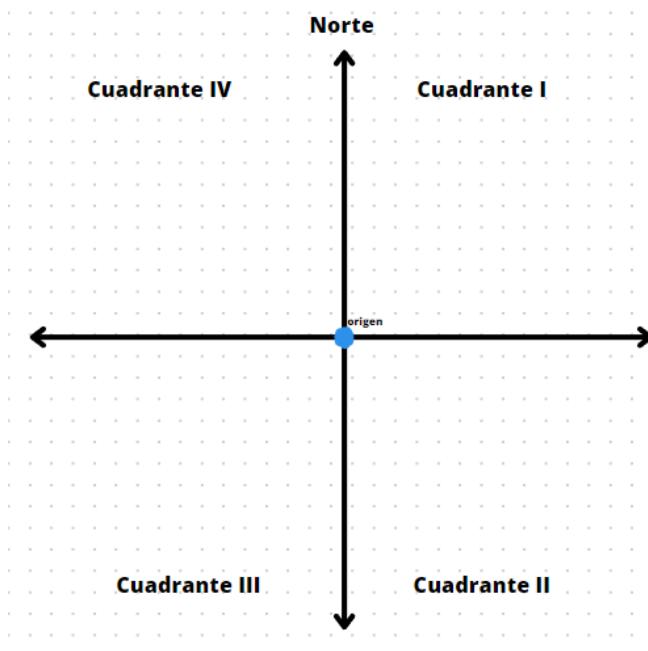


Figura 87: Ejes para representar diferentes casos. Edición Propia

En donde el origen de coordenadas es la posición inicial del vehículo (punto azul), si suponemos el siguiente caso:

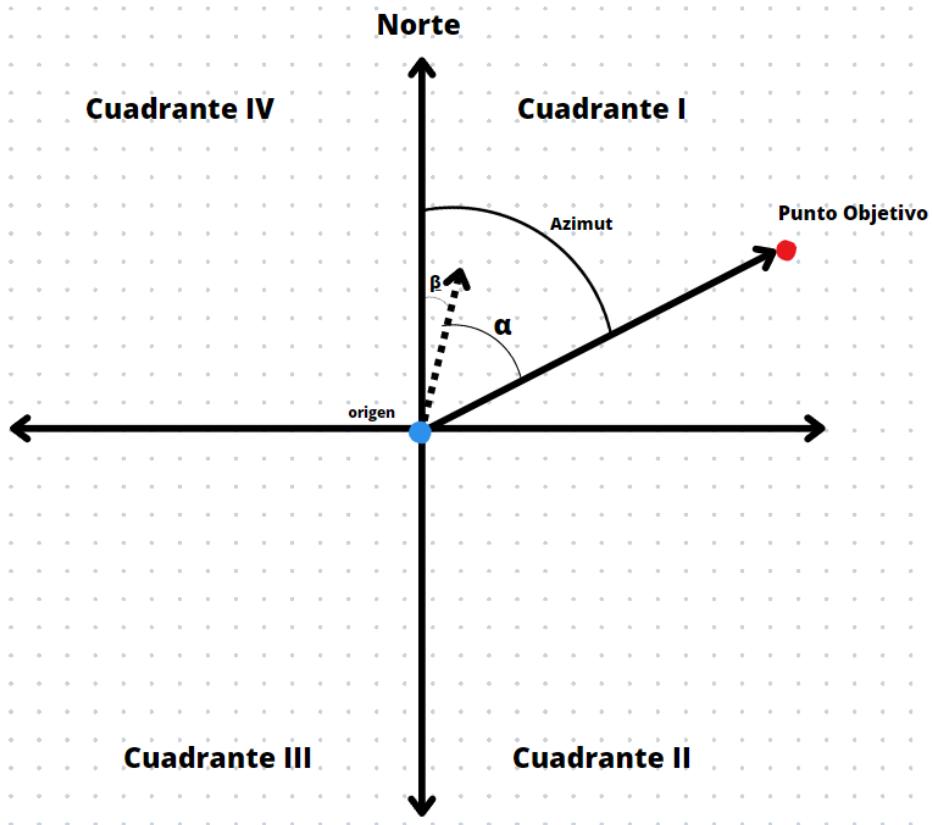


Figura 88: Caso uno, ángulo alfa menor a 50° y azimut mayor al ángulo de orientación del vehículo beta.

Edición Propia

Donde el vector a trazos negro indica la orientación del vehículo con respecto al norte, y el ángulo formado entre el norte geográfico y el vector negro indicara el azimut. Supongamos que el ángulo de error (alfa) tiene un valor menor o igual a 50° y, además, como se observa en la gráfica, el azimut es mayor al ángulo de orientación del vehículo con respecto al norte (beta), entonces será conveniente hacer un giro hacia la derecha (sentido horario) con un ángulo igual al ángulo de error para lograr que el vehículo se oriente hacia la posición del punto objetivo.

Ahora supongamos otro caso:

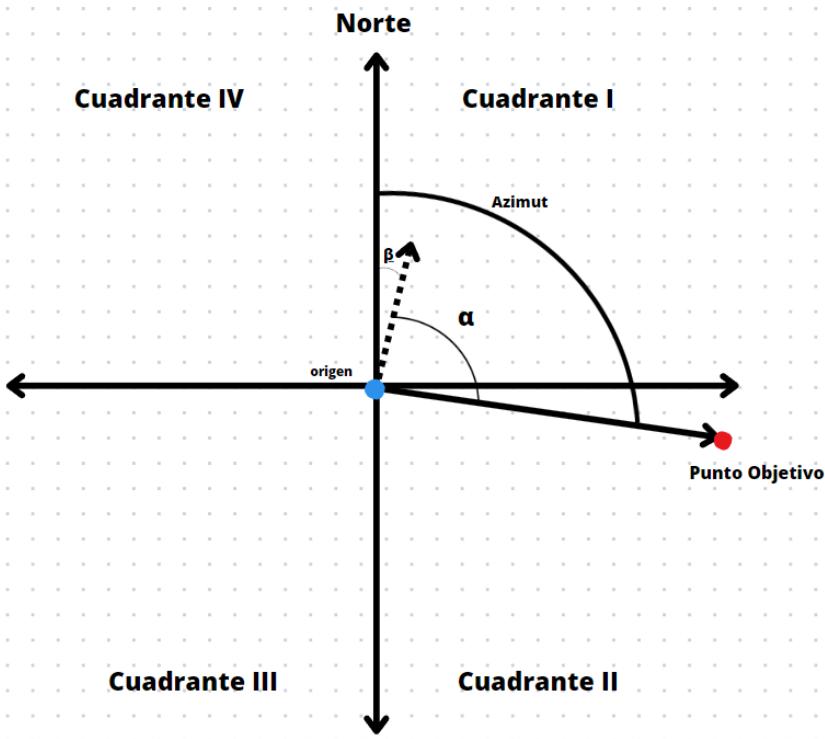


Figura 89: Caso dos, ángulo alfa mayor a 50° y menor o igual a 90° , Edición Propia

En este podemos ver un caso similar, pero con un ángulo de error (alfa) que es mayor que 50° y menor o igual a 90° , en esta situación lo conveniente es que el vehículo decida girar totalmente hacia la derecha (sentido horario) hasta que el ángulo de error (alfa) se reduzca a un valor menor de 50° y se retoma el caso anterior.

Hasta este momento se dio dos claros ejemplos en los cuales es conveniente avanzar hacia el objetivo girando hacia un sentido u otro, pero esto no siempre es lo más conveniente. Para plantear esto se muestra el siguiente caso:

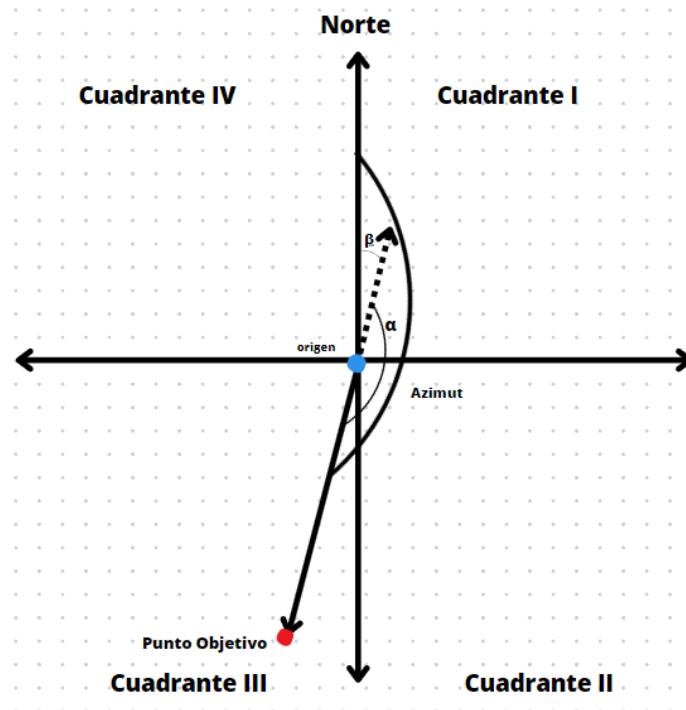


Figura 90: Caso tres, ángulo alfa aproximadamente 180°

Podemos observar que ahora el valor del ángulo de error es aproximadamente 180° . Si planteamos una solución similar a las anteriores, el vehículo deberá avanzar con un giro total a la derecha hasta que en algún momento llegue a orientarse hacia el punto objetivo, pero al hacer esto deberá hacer un recorrido muy pronunciado. Para este caso, la solución más sencilla es hacer que el vehículo retroceda con un giro total hacia la izquierda hasta que logre orientarse con la dirección del punto objetivo, y el ángulo de error se reduzca a un valor menor de 50° y se vuelva a encontrar en alguno de los casos anteriores.

Siguiendo esta lógica se plantean las distintas situaciones que se pueden llegar a producir de acuerdo a las orientaciones tanto del vehículo como del punto objetivo.

Velocidad del vehículo

Otro parámetro importante a tener en cuenta es la velocidad con la cual el vehículo se desplazará hacia el punto objetivo. Debido a esta necesidad se planteó la utilización de un motor DC ya que este puede controlarse de manera sencilla a través de un driver (L298N) y una modulación por ancho de pulsos (conocido por sus siglas en inglés como PWM, Pulse Width Modulation).

Dado esto se plantearon las siguientes condiciones:

- Cuando el vehículo este en la orientación correcta es decir que el ángulo de error sea menor a 10° , el ciclo útil de la modulación (duty cycle), será del 50% en avance (giro de la hélice en sentido horario).
- Cuando se desvíe de su correcta orientación (con un ángulo de error como máximo de 90°), el ciclo útil irá aumentando de manera proporcional al aumento del ángulo de error en avance. Esto se hace para garantizar que el vehículo se oriente de manera rápida en la dirección correcta y además se pueda corregir la dirección en caso de agentes externos (viento u olas).
- En los casos en donde el vehículo deba retroceder, se plantea un ciclo útil al 75%, además de tener en cuenta que, por ejemplo, si el vehículo se encuentra avanzando y tiene que realizar posteriormente una acción de retroceso, este primero detendrá el motor por un segundo y luego habilitara el giro en sentido contrario para retroceder, esto se realiza para que no haya cambios bruscos en el sentido de la corriente que fluye por el motor.

Almacenamiento de los puntos de coordenadas

Se estuvo hablando del “punto objetivo”, el cual puede ser uno o más de acuerdo a cada necesidad, para este proyecto se decidió que la cantidad máxima de puntos de trayectoria que puede realizar el vehículo es igual 10.

Estos puntos que se envían desde el servidor se guardan en un arreglo del siguiente tipo:

```
coor trayectoria[MAX_COORDENADAS];
```

Donde **trayectoria** es el tipo de dato que se almacena en el arreglo “coord” y este es una estructura de datos la cual contiene los valores de latitud y longitud:

```
struct coor {
    double lat;
    double lon;
};
```

“MAX_COORDENADAS” es un valor constante igual a 10, que define el tamaño del arreglo.

Esto funciona de la siguiente manera, por ejemplo, cuando se envía desde el servidor tres puntos de trayectoria al vehículo, estos se almacenan en el arreglo ocupando tres espacios respectivamente, una vez el vehículo llega al primer punto objetivo, se debe pasar al segundo, esto se hace incrementando la posición del arreglo es decir pasar de la posición 0 a la posición 1 para establecer el nuevo punto objetivo al cual debe dirigirse. Una vez se llega a todos los puntos objetivos establecidos el vehículo se detiene y espera nuevas órdenes.

8 Capítulo 8: CARACTERISTICAS ADICIONALES DEL VEHICULO

En los grandes reservorios de agua, es común encontrar una variedad de obstáculos sobre los cuales las personas no tienen control directo. Debido a esta consideración, se planteó la necesidad de implementar un sistema de detección de obstáculos para el bloque del vehículo. Es importante destacar que se ha incorporado el sensor JSN-SR04t, el cual ofrece la ventaja de ser resistente al agua, como se describió anteriormente en sus características técnicas. En este capítulo, se proporcionará una descripción detallada de cómo se llevó a cabo la detección de obstáculos utilizando este sensor.

8.1 Consideraciones y funcionamiento

Dentro de las características del sensor JSN-SR04T se destaca un alcance de 4.5 metros. Basándonos en esta especificación, se tomó la decisión de configurar la detección de obstáculos para que el sensor identificara cualquier obstrucción situada a una distancia de 3 metros o menos desde su posición en el frente del compartimento estanco 1, que corresponde a la parte delantera del vehículo.

Cuando el sensor detecta un obstáculo a una distancia igual o menor a 3 metros, se ejecuta una serie de acciones automatizadas. En primer lugar, el sistema mecánico se detiene. Luego, el vehículo retrocede y gira hacia la derecha durante un período de 10 segundos. Este giro provoca un cambio en la dirección del frente del vehículo. Transcurridos los 10 segundos, el sistema mecánico se apaga durante 1 segundos antes de avanzar nuevamente durante otros 10 segundos. Este ciclo de acciones se repite de manera continua hasta que el sensor deje de detectar un obstáculo a una distancia menor o igual a 3 metros.

Este enfoque garantiza que el vehículo pueda sortear obstáculos de manera efectiva y evitar colisiones mientras se encuentra en funcionamiento en un entorno acuático.

Se adjuntarán imágenes para una mejor comprensión

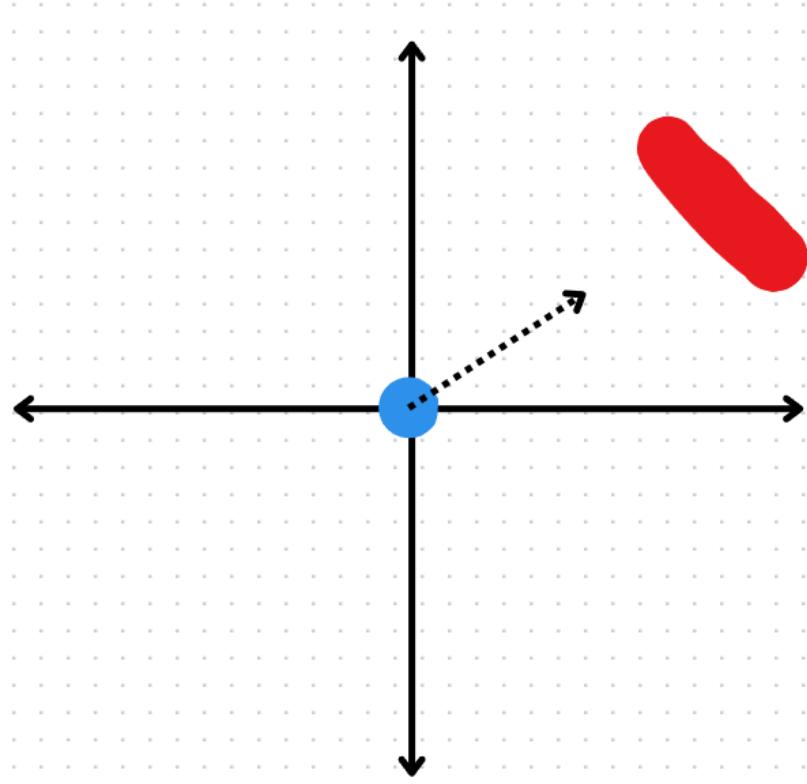


Figura 91: Obstáculo frente del vehículo, Edición Propia

En la figura anterior, se representa una situación en la que el vehículo detecta un obstáculo situado a una distancia de 3 metros o menos, tal como se indica por la línea punteada que representa la dirección de detección del vehículo. En esta circunstancia, el vehículo procederá a ejecutar las acciones previamente descritas para sortear el obstáculo de manera efectiva.

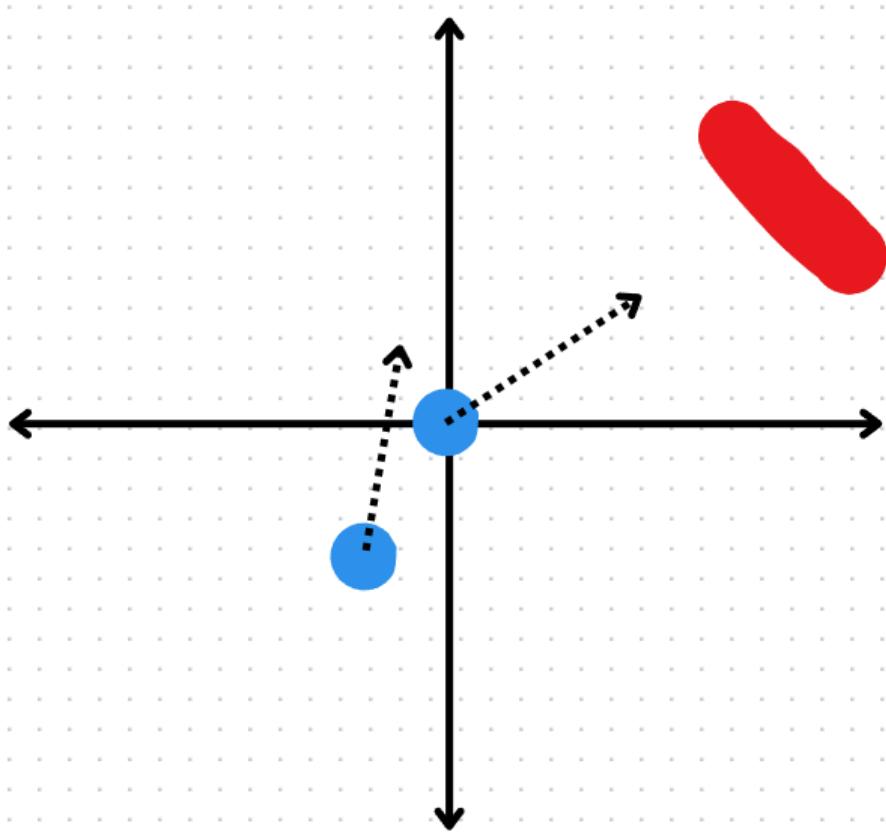


Figura 92: Posición del vehículo 10 segundos después, Edición Propia

En la figura 92, se puede observar cómo el vehículo, 10 segundos después de detectar el obstáculo, retrocede una cierta distancia, modificando la orientación de su sensor para evitar el obstáculo. En situaciones donde el obstáculo sea más grande, este proceso se repetirá hasta que el vehículo lo haya esquivado por completo y pueda retomar su trayectoria original. Para una comprensión más detallada de esta detección y maniobra, se adjuntará un video de la prueba correspondiente junto con el código fuente implementado en el CD que acompaña este documento.

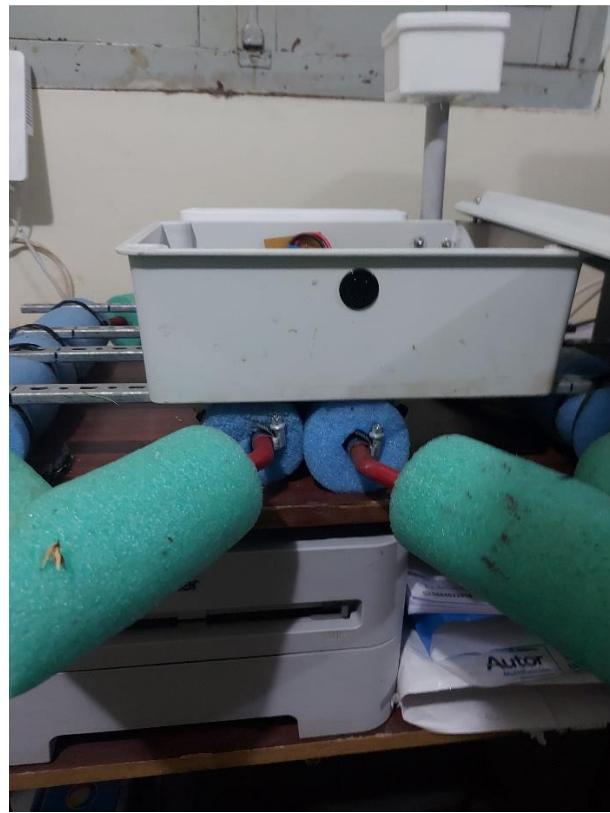


Figura 93: Vehículo con Sensor JSN-SR04T, parte frontal, Edición Propia

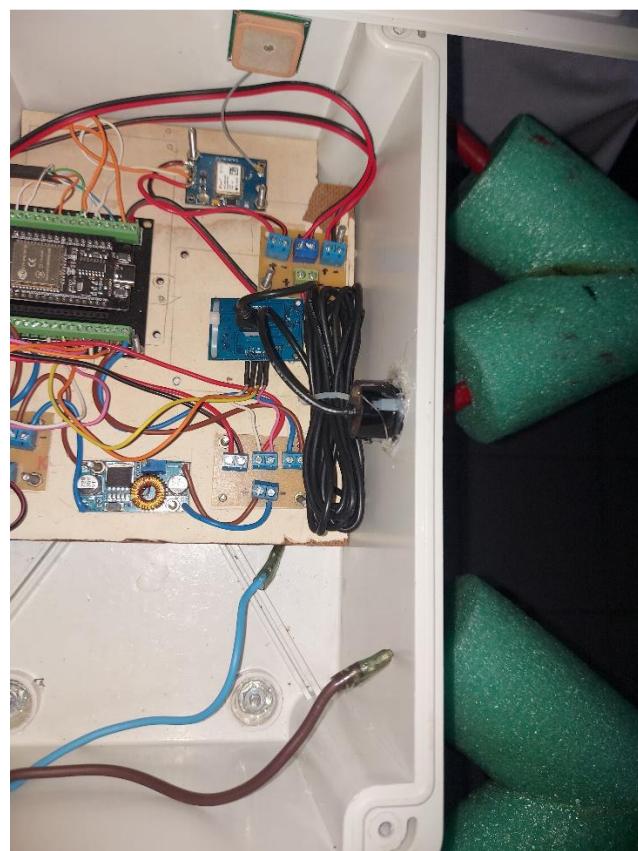


Figura 94: Vehículo con Sensor JSN-SR04T, Edición Propia

9 Capítulo 9: INTERFAZ DE VISUALIZACION DE DATOS

En este capítulo, se abordarán tanto el software utilizado para la visualización de datos obtenidos por el vehículo como la programación del mismo. La información recopilada por el vehículo se somete a un proceso que incluye la recepción, el almacenamiento y el procesamiento por parte del Gateway. Posteriormente, estos datos se transmiten mediante el protocolo MQTT a una plataforma de visualización denominada Node-RED. En esta plataforma, se ha diseñado un dashboard específico para presentar y analizar los datos de manera efectiva. A lo largo de este capítulo, profundizaremos en estos conceptos y su implementación.

9.1 Node-RED y Dashboard

Node-RED es una plataforma de programación visual basada en flujos que se utiliza para conectar dispositivos, servicios y API en el contexto de la Internet de las cosas (IoT) y la automatización de tareas. Fue desarrollado inicialmente por IBM Emerging Technology y se ha convertido en un proyecto de código abierto ampliamente adoptado.

Sus capacidades son las siguientes:

- Programación Visual basada en Flujos: Node-RED proporciona una interfaz gráfica basada en la web que permite a los usuarios crear aplicaciones conectando nodos (bloques de construcción) en un flujo. Los nodos representan diferentes componentes y servicios, y los usuarios pueden conectarlos arrastrando y soltando líneas entre ellos para definir cómo fluyen los datos y las acciones en la aplicación.
- Soporte para Internet de las Cosas (IoT): Node-RED está diseñado específicamente para facilitar la creación de aplicaciones IoT. Ofrece una amplia gama de nodos predefinidos que permiten interactuar con dispositivos IoT, sensores, actuadores y servicios en la nube. Esto lo convierte en una herramienta valiosa para la integración y automatización en entornos IoT.
- Extensibilidad: Node-RED es altamente extensible y permite a los usuarios crear sus propios nodos personalizados en JavaScript para adaptarse a sus

necesidades específicas. Además, hay una comunidad activa que contribuye con nodos adicionales que pueden extender aún más las capacidades de la plataforma.

- Amplia Integración: Node-RED puede integrarse con una amplia gama de servicios y protocolos, incluyendo MQTT, HTTP, TCP/UDP, bases de datos, servicios en la nube, redes sociales y más. Esto facilita la conexión y la automatización de datos y servicios de diversas fuentes.
- Flujos de Datos en Tiempo Real: Node-RED permite crear flujos de datos en tiempo real, lo que significa que se pueden tomar decisiones y realizar acciones de manera instantánea en función de la información que fluye a través del sistema.
- Facilidad de Uso: Node-RED es conocido por su simplicidad y facilidad de uso, lo que lo hace accesible incluso para usuarios sin experiencia en programación.

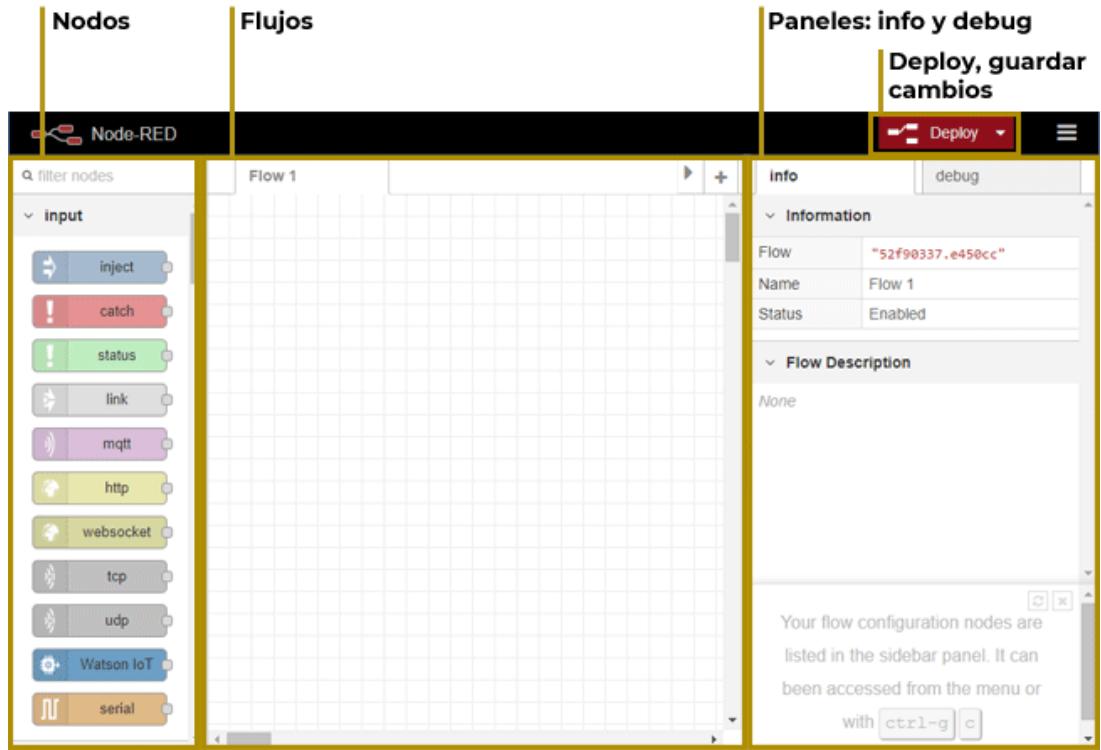


Figura 95: Entorno de desarrollo de Node-Red por <https://programarfacil.com/>

Sector “Nodos” aquí se encuentran los diferentes nodos que se pueden utilizar en el entorno de programación, estos son los elementos que realizan las acciones en el diagrama de flujo.

Sector “Flujos” es el espacio de trabajo y se lo denomina Flow, es donde se colocan los diferentes nodos y se realizan las conexiones entre ellos a través de hilos para formar el flujo del programa.

Sector “Paneles”, aquí se puede encontrar múltiples funciones del entorno, como obtener información de los nodos, utilizar la ventana de depuración (debug) que sirve para observar los mensajes de error que se pueden tener en el programa. Otra función importante es que se puede configurar el aspecto de la aplicación, como ser, tamaño, tipo y color de la fuente, el color de fondo de la aplicación, el aspecto y tamaño de todos los elementos que aparecen en pantalla. Se puede acceder también al instalador de librerías para sumar nuevos nodos al sector de “Nodos”.

Dashboard

Un dashboard en Node-RED es una interfaz gráfica de usuario que permite mostrar información de manera visual y accesible, así como interactuar con sistemas y datos en

tiempo real. Node-RED es una plataforma de código abierto, y el dashboard es una de sus funcionalidades más destacadas.

Un dashboard en Node-RED se compone de varios elementos, como widgets y paneles, que se pueden personalizar y organizar según las necesidades del proyecto. Estos elementos pueden mostrar información en forma de gráficos, tablas, medidores, botones y más. Aquí hay algunas características clave de un dashboard de Node-RED:

- Widgets Personalizables: Puedes agregar widgets como medidores, gráficos de barras, botones, campos de texto y más para mostrar información específica de tus datos.
- Conexión en Tiempo Real: Los datos se actualizan en tiempo real en la interfaz, lo que permite monitorear y responder a cambios instantáneamente.
- Interacción del Usuario: Los usuarios pueden interactuar con el dashboard, por ejemplo, para cambiar configuraciones, activar acciones o filtrar datos.
- Diseño Flexible: Puedes organizar los widgets y paneles de manera personalizada, ajustando su tamaño y posición en la interfaz.
- Integración con Flujos de Node-RED: El dashboard se integra de manera nativa con los flujos de trabajo de Node-RED, lo que facilita la conexión con diversas fuentes de datos y sistemas.
- Seguridad: Node-RED ofrece opciones para asegurar el acceso al dashboard mediante autenticación y autorización de usuarios.

9.2 Programación realizada en Node-RED

El programa realizado consta de un único Flow al cual se le puso el nombre "Dashboard". Dentro del cual se realizó todo el diagrama de flujo correspondiente al Dashboard que se realizó.

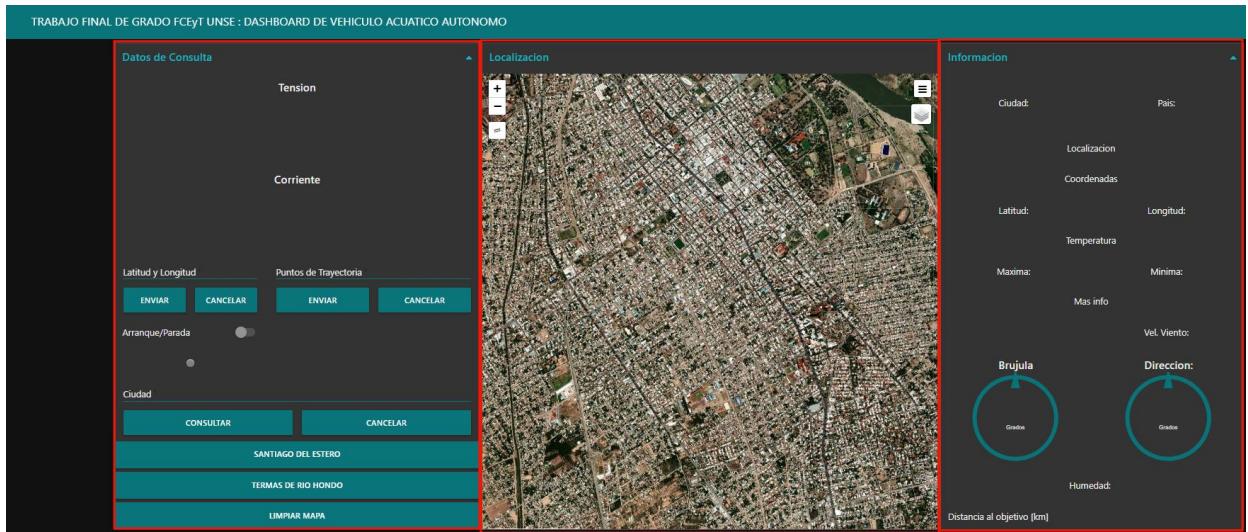


Figura 96: Dashboard realizado en Node-Red, Edición Propia

Podemos observar que el Dashboard consta de tres ventanas o grupos de datos:

- Ventana 1: Llamada “Datos de Consulta” en la misma encontramos gráficos, botones y casillas para el envío de datos.
- Ventana 2: Llamada “Localización” podemos observar un mapa interactivo, el cual nos permitirá ver en tiempo real la posición del vehículo, además de poder obtener la latitud y longitud de cualquier punto que se desee
- Ventana 3: Llamada “Información” en donde se pueden observar diferentes textos los cuales muestran información de todo tipo, como la orientación del vehículo, la distancia del vehículo al punto objetivo, la velocidad del viento, etc.

Se presentará un fragmento del diagrama del entorno de desarrollo, para dar una explicación de cómo funciona el flujo y la funcionalidad de algunos nodos, ya que el diagrama completo se podrá observar con mayor lujo de detalles en los archivos del CD que acompaña este documento.

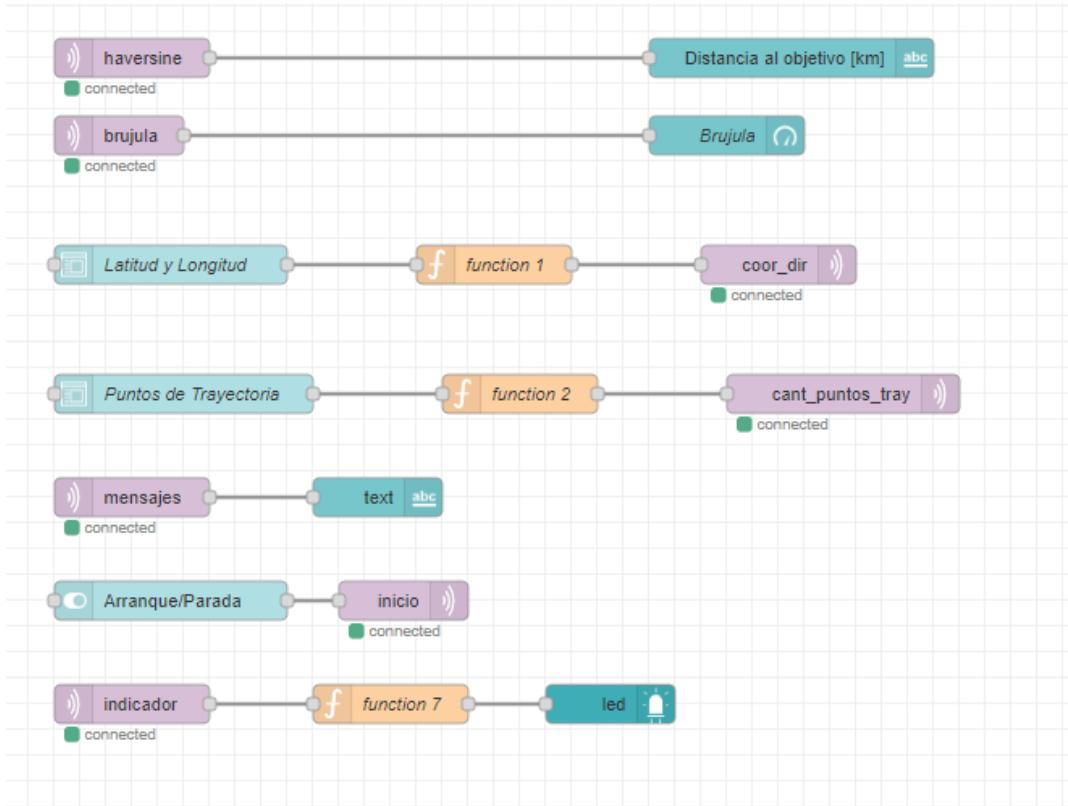


Figura 97: Diagrama de flujo para la visualización de datos recibidos por MQTT y para el envío de datos por MQTT, Edición Propia

El flujo siempre avanza de izquierda a derecha, y como se observa hay nodos de diferentes colores y símbolos, se explicará en función de los colores que es cada uno:

- Nodos de color rosa: estos nodos sirven para realizar la comunicación MQTT tanto para la entrada como la salida de datos. Los nodos que tienen el conector del lado derecho, son aquellos a los cuales le llega información desde el Gateway. Y aquellos que se conectan del lado izquierdo son para enviar información desde Node-Red al Gateway.
- Nodos de color celeste claro: estos se utilizan para realizar acciones en el dashboard, que posteriormente serán enviados por MQTT al Gateway y de ahí al vehículo.

- Nodos de color celeste oscuro: estos representan de forma gráfica en el Dashboard la información enviada desde el vehículo y recibida por el Gateway.
- Nodos de color naranja: Estos nodos llamados “function” son programados en JavaScript y sirven para pasar de un tipo de dato a otro para que sean accesibles para otros nodos.

El funcionamiento del programa mostrado es sencillo, los nodos MQTT de entrada (con el flujo a la derecha) reciben datos publicados en un tópico determinado, por ejemplo “haversine” que recibe la distancia en km con respecto al objetivo. Estos datos continúan el flujo para que pasen a un nodo que se vincula a un objeto en el dashboard en el cual vamos a mostrar el dato en cuestión. En el caso de los nodos MQTT de salida (con el flujo a la izquierda), se utilizan para publicar datos en el microcontrolador que oficia como Gateway de tal manera que estos son usados para acciones de control. El flujo descripto en la imagen superior, muestra un nodo de inserción de texto el cual es un objeto que aparece en el dashboard para introducir datos como cadenas de texto, y posteriormente se usa un nodo de funciones para establecer como se publicara el dato y posteriormente es publicado por MQTT.

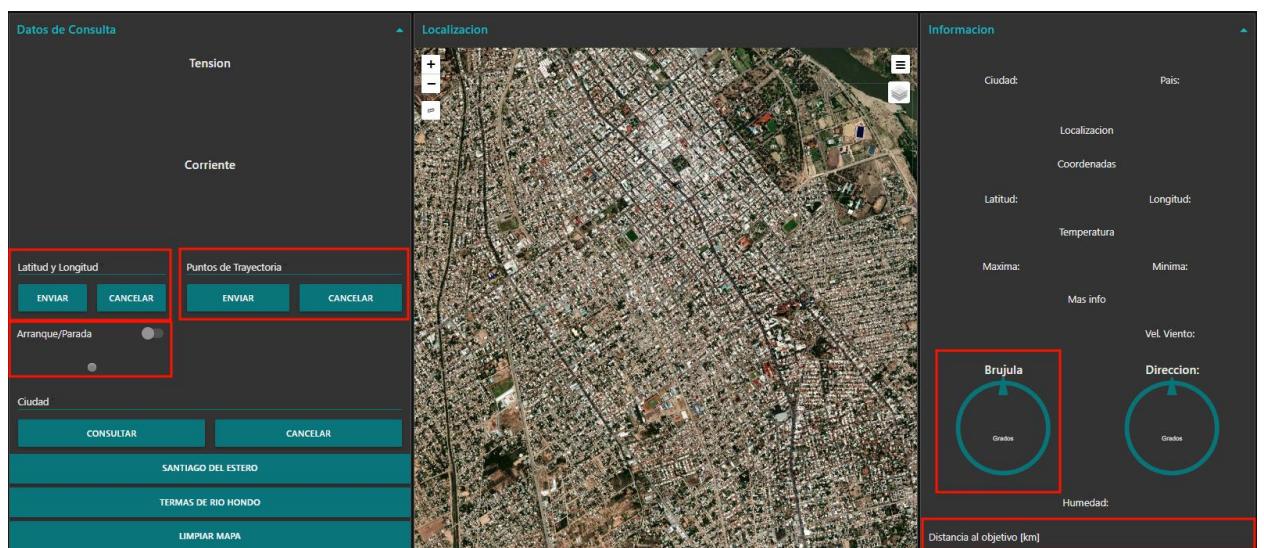


Figura 98: Representación en el Dashboard del diagrama de flujo, Edición Propia

Proceso para generar archivos CSV con datos de tensión y corriente

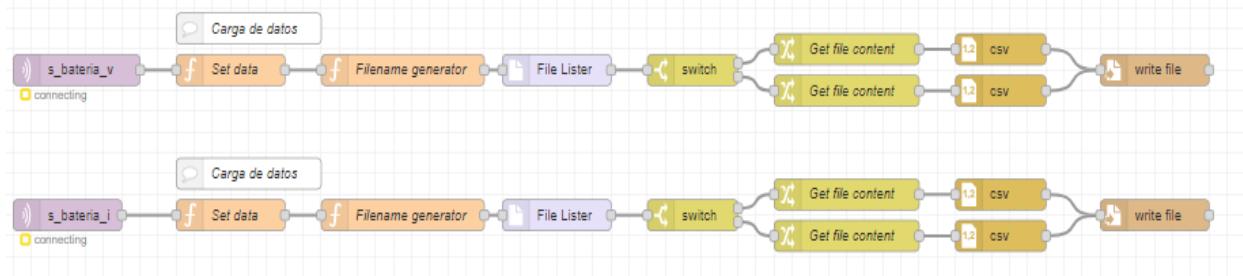


Figura 99: Flujos de almacenamiento de datos en archivo CSV, Edición Propia

Estos flujos tienen la finalidad de generar un archivo CSV y cargar datos procedentes de flujos MQTT respectivos (tensión y corriente obtenidos por el sensor del vehículo). Ambos flujos desempeñan la misma tarea, pero crean y cargan archivos que contienen información separada sobre mediciones de tensión y corriente.

Cuando un dato es publicado a través de MQTT, el nodo de funciones "Set Data" primero genera una marca de tiempo (timestamp) y organiza ambos datos en un formato JSON. A continuación, el nodo "Filename" genera el nombre del archivo y especifica la ubicación donde se almacenará el dato. En nuestro caso, este se guarda en una carpeta denominada "data" en la unidad "D:".

El siguiente nodo, llamado "FS Lister" (en color celeste), configura la ubicación del archivo para dirigir el flujo de datos hacia él.

El nodo "Switch" (color verde) bifurca el flujo en dos posibles casos y determina si el dato se envía a un archivo que aún no existe o a uno que ya existe. Si el archivo no existe, el "Switch" direcciona el flujo de datos a un proceso donde se crea un archivo CSV, y en la primera fila de este se incluyen los nombres de las columnas. Si el archivo ya existe, el flujo se dirige a un proceso donde el dato se carga en el archivo correspondiente, en la última fila de este.

El archivo CSV que arroja como resultado, presenta una tabla de la siguiente forma:

Tabla 3: Valores de corriente obtenidos, Edición Propia

Fecha Hora, Corriente (mA)
8/10/2023 17:41:43,124.5
8/10/2023 17:41:54,119.84
8/10/2023 17:42:05,128.89
8/10/2023 17:42:15,584.28
8/10/2023 17:42:26,584.28

8/10/2023 17:42:36,289.15
8/10/2023 17:42:47,294.87
8/10/2023 17:43:07,114.39
8/10/2023 17:43:17,124.5
8/10/2023 17:43:28,116.65
8/10/2023 17:43:38,124.9
8/10/2023 17:43:48,124.76
8/10/2023 17:43:59,116.39
8/10/2023 17:44:09,120.64
8/10/2023 17:44:19,114.39
8/10/2023 17:44:30,116.78
8/10/2023 17:44:40,115.99

Estos datos pueden ser separados mediante Excel y quedar presentados de la siguiente forma:

Tabla 4: Valores de corriente obtenidos, separados de la fecha y hora, Edición Propia

Fecha	Hora	Corriente (mA)
8/10/2023	17:41	124.5
8/10/2023	17:41	119.84
8/10/2023	17:42	128.89
8/10/2023	17:42	584.28
8/10/2023	17:42	584.28
8/10/2023	17:42	289.15
8/10/2023	17:42	294.87
8/10/2023	17:43	114.39
8/10/2023	17:43	124.5
8/10/2023	17:43	116.65
8/10/2023	17:43	124.9
8/10/2023	17:43	124.76
8/10/2023	17:43	116.39
8/10/2023	17:44	120.64
8/10/2023	17:44	114.39
8/10/2023	17:44	116.78
8/10/2023	17:44	115.99

Para obtener los valores de tensión se realiza el mismo procedimiento.

10 Capítulo 10: PRUEBAS DE FUNCIONAMIENTO y AUTONOMIA

En el presente capítulo, se describirán de manera exhaustiva las pruebas realizadas en ambos bloques con el fin de verificar su correcto funcionamiento. Estas pruebas abarcan la comprobación del sistema mecánico del vehículo, la evaluación del seguimiento del vehículo mediante la plataforma de visualización desarrollada (Dashboard), y una prueba completa del prototipo en un entorno acuático para garantizar su funcionamiento integral en la interacción entre ambos bloques. Además, se proporcionarán detalles sobre los consumos de energía durante cada prueba, lo que nos permitirá estimar la autonomía del sistema. Cabe destacar que se empleará un sensor de tensión y corriente en el bloque vehículo para obtener mediciones, este sensor será explicado a continuación.

10.1 Sensor de Tensión y Corriente del Vehículo

Cuando se habla de cualquier tipo de movilidad eléctrica, siempre se habla de la autonomía del vehículo en cuestión. Por lo tanto, es importante desarrollar mediciones de corriente y tensión de tal forma que estas mediciones sean significativas. Dado que el desarrollo es de un vehículo, se hace difícil hacer las mediciones con equipamiento normal como amperímetros o voltímetros, por lo tanto, se plantea la necesidad de montar un equipo a bordo que funcione de igual manera.

La forma en la que realizaremos las mediciones será a través del conversor analógico digital (ADC) del ESP32. Todos los modelos de ESP32 poseen dos ADC de aproximaciones sucesivas de máximo 12bits.

Vref es el voltaje de referencia utilizado internamente por los ADCs ESP32 para medir el voltaje de entrada. Los ADCs ESP32 pueden medir voltajes analógicos de 0 V a Vref. Entre los diferentes chips, el Vref varía, la mediana es de 1,1 V. Con el fin de convertir voltajes mayores que Vref, los voltajes de entrada pueden ser atenuados antes de ser introducidos en los ADCs. Hay 4 opciones de atenuación disponibles, cuanto mayor sea la atenuación, mayor será la tensión de entrada medible:

Attenuation	Measurable input voltage range
ADC_ATTEN_DB_0	100 mV ~ 950 mV
ADC_ATTEN_DB_2_5	100 mV ~ 1250 mV
ADC_ATTEN_DB_6	150 mV ~ 1750 mV
ADC_ATTEN_DB_11	150 mV ~ 2450 mV

Figura 100: Tabla de rangos de tensión en función de atenuación dada por el fabricante por <https://docs.espressif.com/>

En esta tabla podemos ver los distintos rangos medibles de tensión en función de la atenuación seleccionada, donde la tensión por defecto es -11dB. Una de las desventajas que tiene el ADC de este microcontrolador es que tiene un significativo comportamiento no lineal. En esto, la atenuación juega un papel importante, ya que mientras menos atenuada sea la señal antes de entrar al ADC, más lineal se comporta, a costas de un menor rango de voltaje. Por lo que para seleccionar con que atenuación se trabajará, primero se debe conocer el rango de voltaje que será medido.

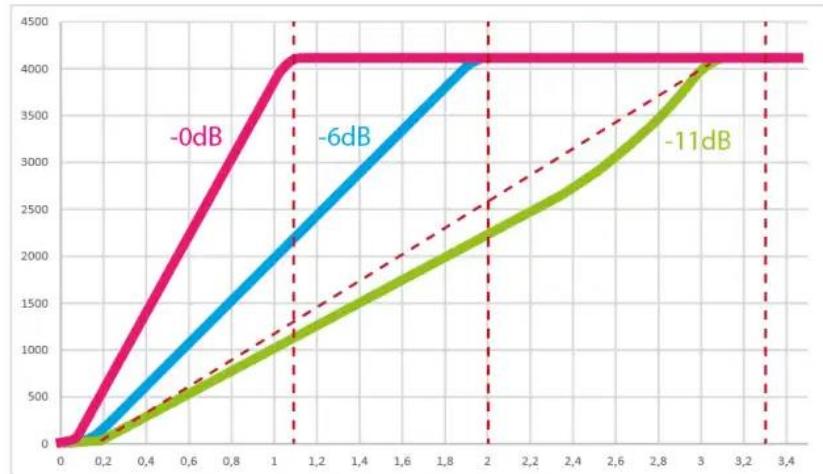


Figura 101: Curvas de cuantificación del ADC para distintos niveles de atenuación por <https://www.luisllamas.es/esp32-adc/>

Conociendo las capacidades que tenemos para desarrollar mediciones, proponemos un modelo para la medición de corriente.

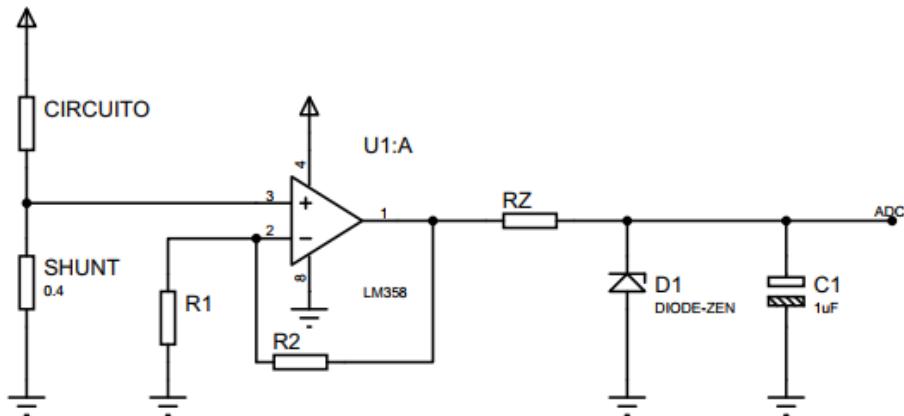


Figura 102: Circuito para medición de corriente, Edición Propia

En este circuito vemos lo que es básicamente un divisor de tensión donde la caída de tensión de la segunda resistencia será amplificada por un amplificador no inversor. Pero este, tiene la salvedad especial de que el circuito se desarrolla utilizando una resistencia shunt.

Un shunt se caracteriza por tener una resistencia de bajo valor, siendo especialmente útil en la medición de corriente, razón por la cual también recibe el nombre de resistencia de censado de corriente. Su aplicación principal se da cuando la corriente que se desea medir excede el rango de capacidad del dispositivo de medición convencional. En tales casos, el shunt se conecta en paralelo al dispositivo de medición mencionado. La corriente total fluye a través del shunt, generando una caída de tensión que puede ser cuantificada. Mediante la aplicación de la ley de Ohm y el conocimiento de la resistencia del shunt, es posible calcular la corriente ($I = V/R$). Con el objetivo de minimizar las pérdidas de potencia y, como consecuencia, la generación de calor, es fundamental que los shunts posean un valor resistivo extremadamente bajo.

Como se mencionó, estas resistencias se utilizan para realizar mediciones con cierta precisión. Estas características las vuelven difíciles de conseguir y con costos bastante elevados. Dada la dificultad de conseguir una y para los fines de este proyecto donde lo que se desea es conseguir valores estimados de corriente para establecer datos de autonomía, se decidió usar como shunt la resistencia de menor valor que era posible conseguir por los proveedores de la ciudad de Santiago del Estero. En nuestro caso fue una resistencia de metal film de 0.4Ω .

Para poder establecer los valores de ganancia y, por consiguiente, el valor de atenuación al que será configurado el ADC, es necesario conocer los consumos que tendrá el vehículo. Estos valores se obtuvieron reemplazando la batería del vehículo por una fuente de laboratorio. De esta forma se estableció:

- Cuando se encuentra estático se tiene un consumo aproximado de **100mA**.
- Cuando se encuentra en movimiento, con movimientos forzados de motor y actuaciones del servomotor se tiene un consumo aproximado de **500mA**.

De esta forma, la caída de tensión en la resistencia shunt ante el menor consumo será:

$$V_{min} := 0.4 \Omega \cdot 100 \text{ mA} = 40 \text{ mV}$$

La caída de tensión en la resistencia shunt ante el mayor de los consumos será:

$$V_{max} := 0.4 \Omega \cdot 500 \text{ mA} = 200 \text{ mV}$$

Como se ve en la figura 96, para cualquiera de las atenuaciones, estos niveles de tensión se encuentran en una zona no lineal, la cual no es apta para desarrollar mediciones. De esta forma, se utiliza el amplificador operacional en configuración no inversora para amplificar estas tensiones. A modo de dejar un rango más amplio posible para realizar mediciones, se decidió utilizar la atenuación por defecto. Con esto, tenemos un rango más grande para ajustar la ganancia del amplificador operacional.

Se debe trabajar sobre la zona lineal de la curva de cuantificación, como se ve en la figura 96, esta zona es aproximadamente entre 0.4V y 2.2V. Por lo tanto, se debe buscar una ganancia 10 en el amplificador no inversor.

$$V_{0min} := V_{min} \cdot \left(1 + \frac{1 \text{ M}\Omega}{100 \text{ k}\Omega} \right) = 440 \text{ mV}$$

$$V_{0max} := V_{max} \cdot \left(1 + \frac{1 \text{ M}\Omega}{100 \text{ k}\Omega} \right) = 2200 \text{ mV}$$

Donde $R_2=1\text{M}\Omega$ y $R_1=100\text{k}\Omega$.

Dado que cuando se va de los cálculos a la aplicación real pueden aparecer ciertas variaciones, se decidió reemplazar la resistencia R2 por un preset de $2.2M\Omega$ de tal forma que la ganancia pueda ajustarse a las necesidades.

El diodo zener a la salida del circuito de amplificación tiene la tarea de evitar que las tensiones superen los 3.3V, por lo tanto, se selecciona un diodo cuya tensión zener sea 3.3V.

Cuando se van a tomar las mediciones a través del ADC, el fabricante tiene dos recomendaciones muy importantes:

- Insertar un capacitor en paralelo en la entrada (esta es la función del capacitor que está en paralelo con el diodo zener).
- Realizar un multisampling. Esto significa, hacer muchas mediciones y promediarlas para dar lugar a una medición definitiva.

La finalidad de estas dos técnicas es reducir el ruido a la entrada para evitar las variaciones de manera significativa entre 2 mediciones consecutivas.

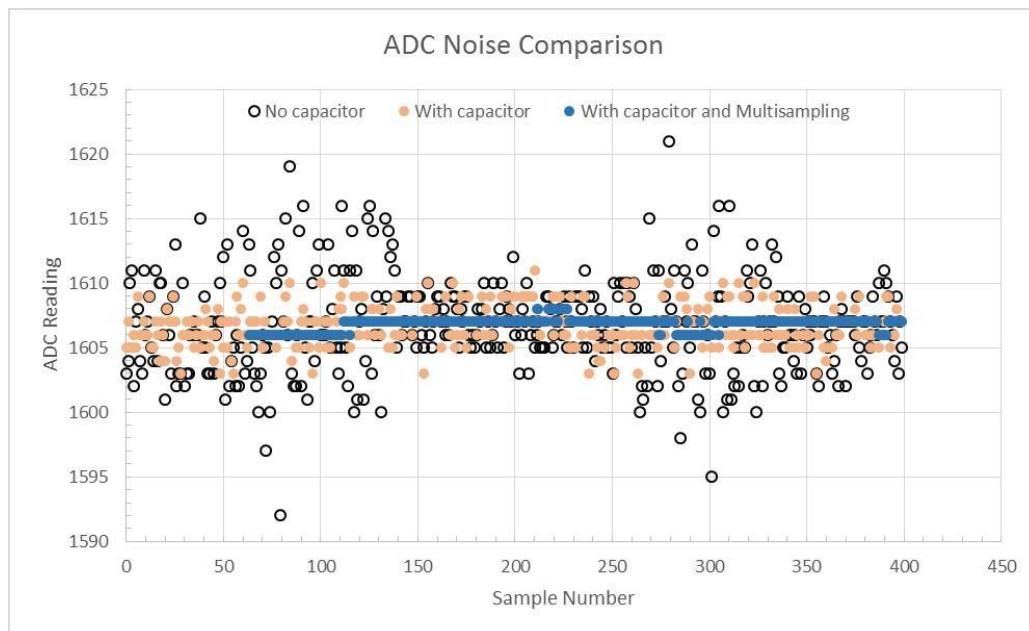


Figura 103: Grafica de comparación de ruido en ADC por <https://docs.espressif.com/>

Teniendo en cuenta todo esto, y recordando que el ADC devuelve un valor entre 0 y 4095, se debe encontrar una expresión para convertir estos valores de tensión cuantificada en un valor de corriente. Para esto, se colocaron cinco cargas para simular el circuito del

vehículo, con el fin de que el sensor mida valores de corriente intermedios entre el mínimo y el máximo. Teniendo en cuenta la figura 98, como pueden existir variaciones entre las mediciones, se desarrolló un programa sencillo en donde se muestran los valores de cuantificación cada 0.5seg. Se tomaron cincuenta muestras por cada carga y estas se promediaron.

Tabla 5: Tabla de corriente y cuantificación para sensor de corriente

Corriente(mA)	Promedio de Cuantificación
536	3798
440	2889
334	2254
227	1424
112	552

Utilizando estos datos, se realizó una gráfica de dispersión para realizar una regresión lineal, a fin de obtener la expresión matemática que relaciona los valores de cuantificación con sus correspondientes valores de corriente.

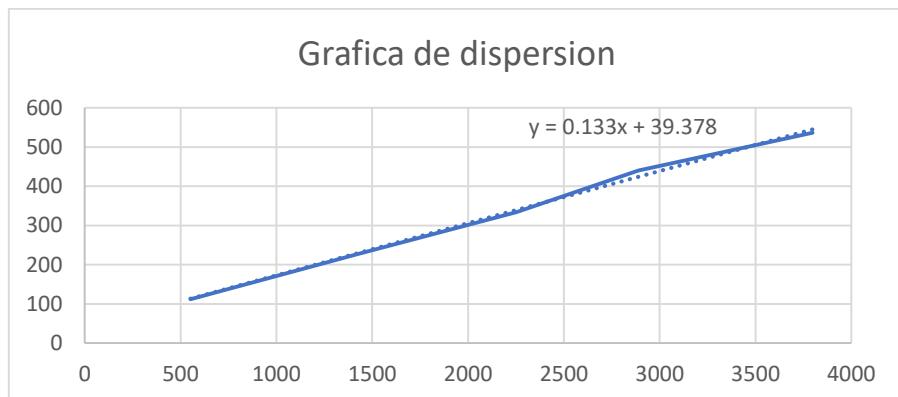


Figura 104: Grafica de dispersión de mediciones de corriente con ADC, Edición Propia

De esta forma se obtiene la expresión:

$$\text{Corriente} = 0.133 \cdot \text{RAW} + 39.378$$

Luego, a modo de comparación, se realizaron mediciones con y sin el capacitor recomendado por el fabricante:



Figura 105: Curva de corriente en función del tiempo sin Capacitor



Figura 106: Curva de corriente en función del tiempo con Capacitor

Como se puede ver, hay una diferencia significativa en el rizado de la señal. Por lo tanto, el capacitor quedó instalado en el circuito.

Para medir tensión, se propone el siguiente circuito:

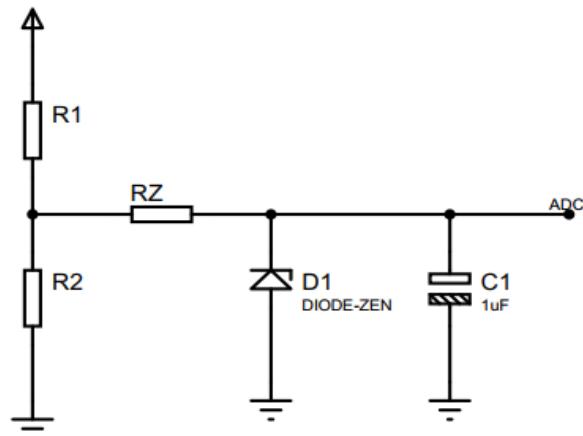


Figura 107: Circuito para medir tensión, Edición Propia

Este circuito es un simple divisor de tensión para reducir la tensión de la batería a los niveles que puede recibir el ADC. Se utilizaron para R1 una resistencia de $1M\Omega$ y un preset de $1M\Omega$ a fin de regular la tensión dentro de los parámetros necesarios.

Al igual que con el sensor de corriente, se tomaron mediciones para cinco niveles de tensión y se tomaron cincuenta muestras de cada uno para realizar un promedio.

Tabla 6: Tabla de tensión y cuantificación para sensor de tensión

Tensión(V)	Promedio de cuantificación
7	1201
9	1261
11	1316
13	1328
15	1355

Así mismo, utilizando estos datos, se realizó una gráfica de dispersión para realizar una regresión lineal, a fin de obtener la expresión matemática que relaciona los valores de cuantificación con sus correspondientes valores de tensión.

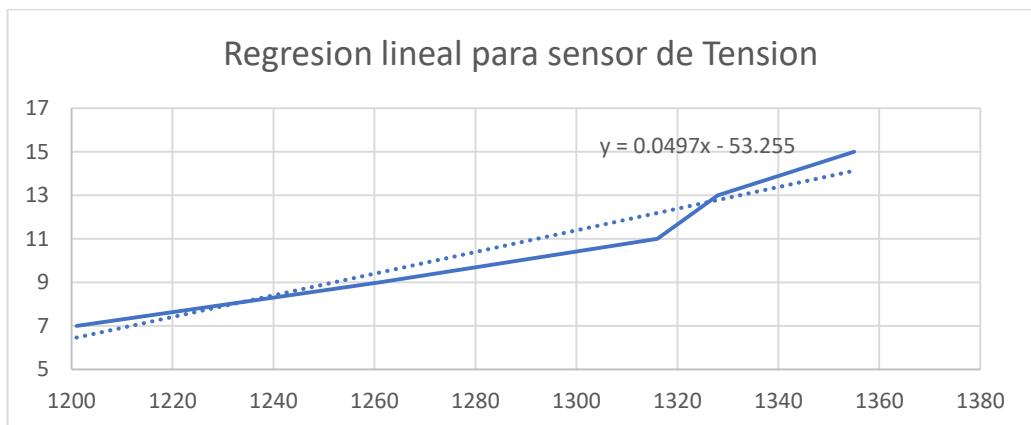


Figura 108: Grafica de dispersión de mediciones de tensión con ADC, Edición Propia

De esta forma se obtiene la expresión:

$$\text{Tension} = 0.0497 \text{ RAW} - 53.255$$

Los datos obtenidos de la medición de tensión son mucho más estables que los obtenidos de la medición corriente. Esto se debe a que la corriente depende en su totalidad de la carga, por lo tanto, es comprensible que, si se tiene una fuente de tensión estable, la medición de tensión sea estable también, pero la corriente fluctúa dependiendo de las condiciones de funcionamiento de la carga y de su ambiente (variaciones ambientales, etc.).

de esta manera se concluye que se obtuvo un sensor con la sensibilidad necesaria para realizar las pruebas que debe atravesar el proyecto. En el anexo B se podrán visualizar los esquemas de conexión y el sensor realizado.

10.2 Prueba 1: Funcionamiento del motor y aleta

Esta prueba se llevó a cabo con el propósito de verificar el correcto funcionamiento de la parte mecánica del vehículo. Para ello, se utilizó una fuente de laboratorio en lugar de la batería, lo que permitió estimar el consumo de todo el sistema durante su funcionamiento.

El procedimiento de la prueba comenzó con el envío de un punto de trayectoria junto con sus coordenadas, con el objetivo de que el vehículo se ubicara en una posición específica. En este caso, se decidió enviar un punto a una distancia aproximada de 100 metros desde la ubicación inicial del vehículo.

Para una comprensión más clara de esta prueba, se adjuntará un video en el CD que acompaña este documento, y a continuación, se proporcionarán imágenes que ilustran el proceso.

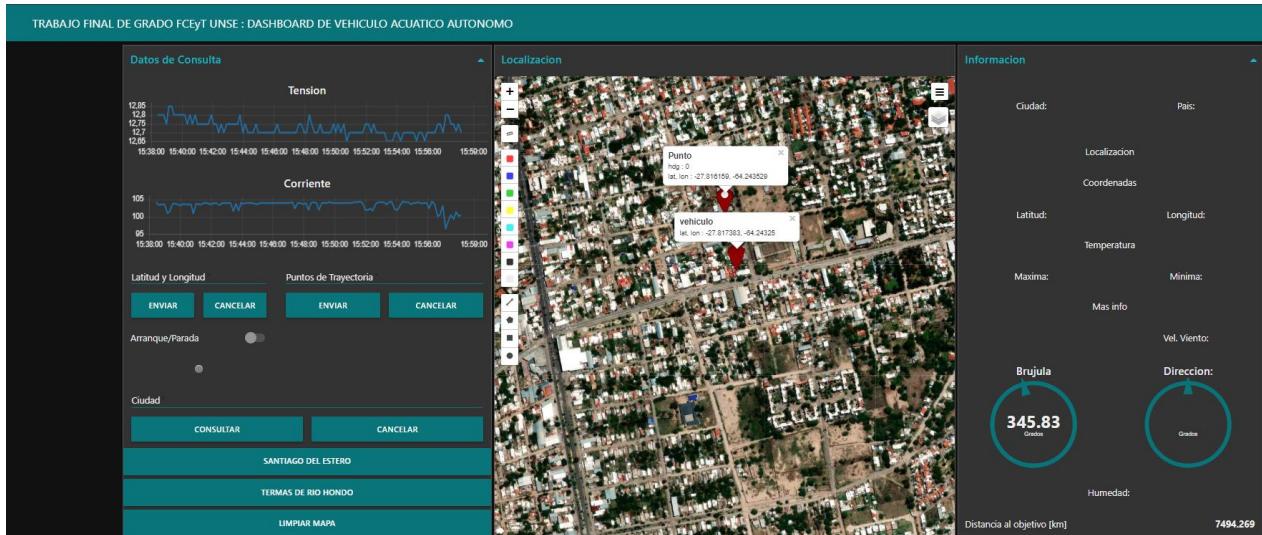


Figura 109: Ubicación del Vehículo y punto de trayectoria, Edición Propia



Figura 110: Consumo del Vehículo en reposo, Edición Propia

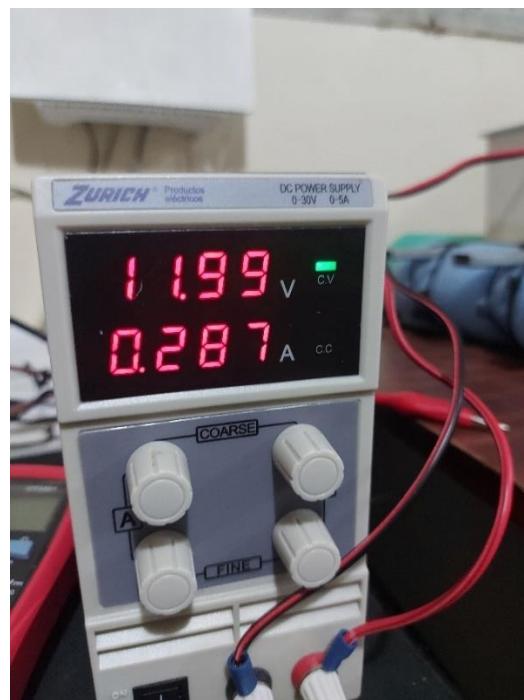


Figura 111: Consumo del vehículo funcionando, Edición Propia

En la figura 110, se muestra el consumo del vehículo cuando todos los módulos están en funcionamiento, pero la parte mecánica permanece inactiva. Esto implica que ni el motor ni el servo motor están realizando movimientos en este momento.

Por otro lado, en la figura 111, se representa el consumo del vehículo cuando la parte mecánica se encuentra activa. Cada vez que el servo motor realiza un movimiento, se observa un aumento en el consumo de aproximadamente 350 mA.

Estos resultados confirman que el vehículo tiene un consumo eléctrico adicional cuando la parte mecánica está en funcionamiento, lo cual era esperado. Estos datos son fundamentales para comprender el rendimiento energético del sistema en diferentes condiciones de operación.

Este conjunto de pruebas ha cumplido con las expectativas, lo que nos permite avanzar hacia la siguiente prueba, que se explicará a continuación.

10.3 Prueba 2: Funcionamiento del Tracker GPS

En esta prueba, se llevó a cabo la verificación del correcto funcionamiento del sistema de GPS. El objetivo era determinar si el sistema era capaz de realizar un seguimiento en tiempo real del vehículo y mostrar este seguimiento en el mapa del dashboard desarrollado.

Se realizó un video mostrando este funcionamiento y a continuación se adjuntarán imágenes para una mejor compresión.

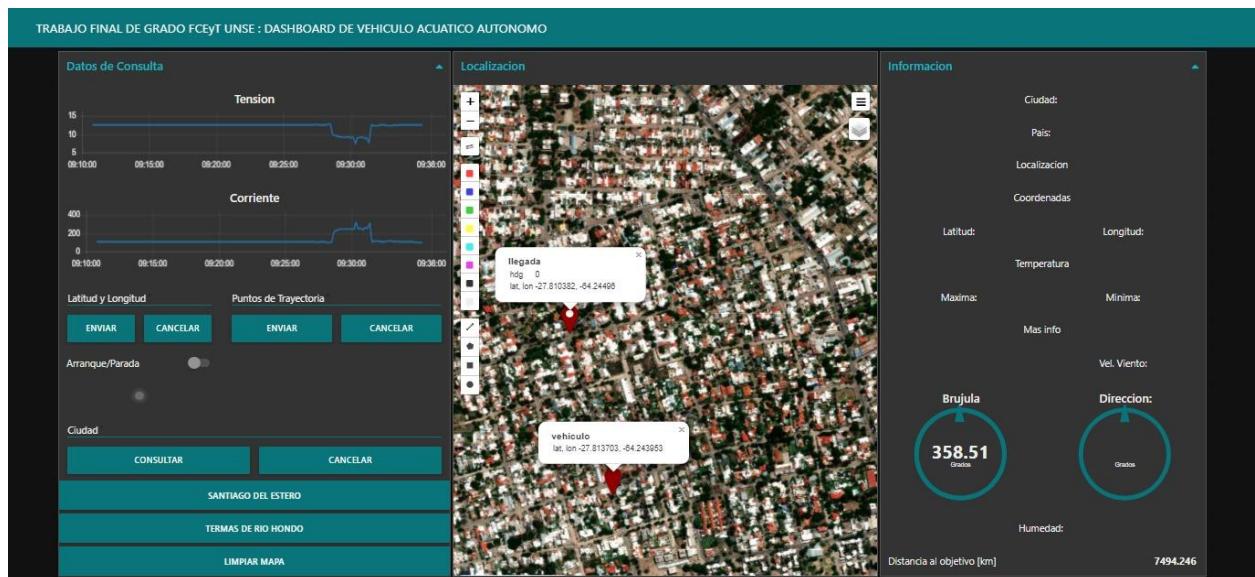


Figura 112: Ubicación del vehículo y punto de trayectoria final, Edición Propia

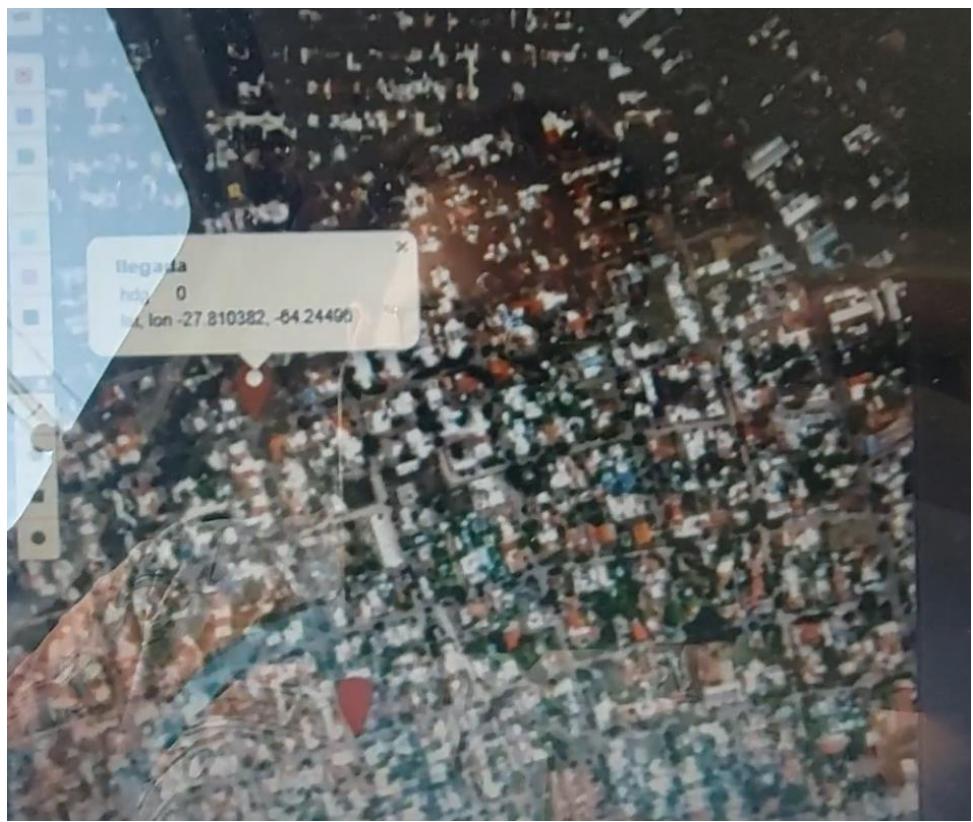


Figura 113: Fragmento del video con la posición de inicio, Edición Propia

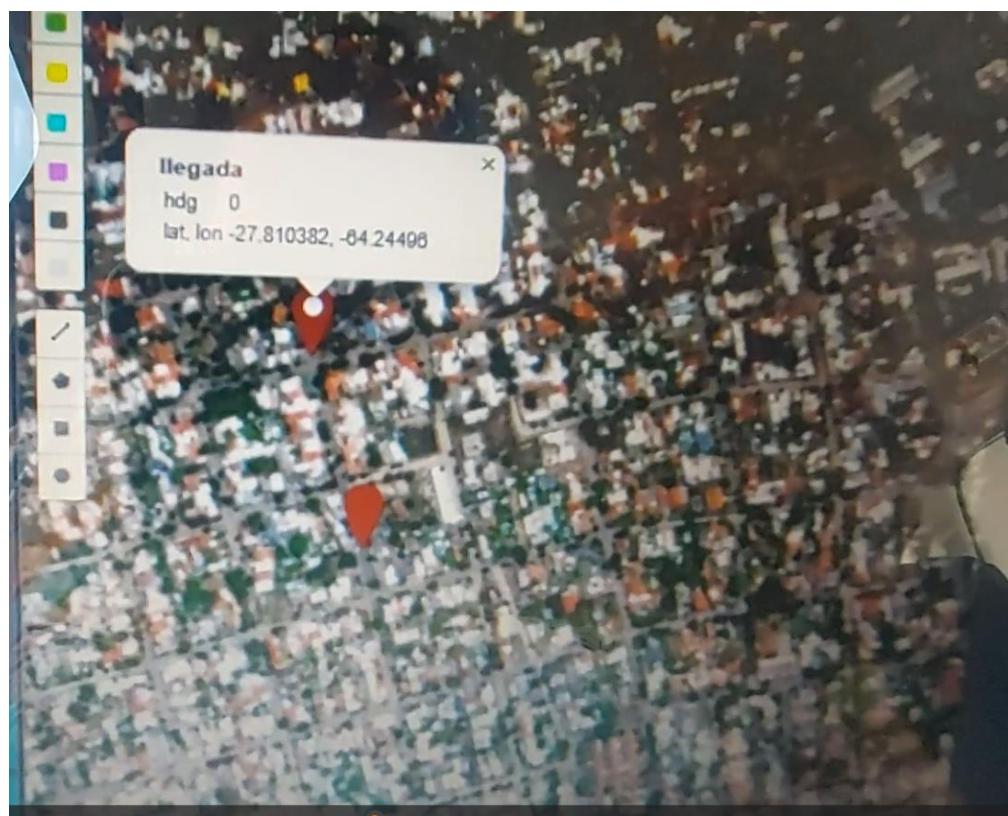


Figura 114: Fragmento del video al actualizar los datos en el dashboard, Edición Propia

Como se puede observar en las imágenes, el seguimiento en tiempo real es preciso, lo que indica que esta prueba resultó satisfactoria. Además, es importante destacar que el consumo de energía durante esta prueba se mantuvo igual que en la prueba anterior, ya que el vehículo no estaba sumergido en agua y no había carga en el sistema mecánico, funcionando solo en vacío.

Es relevante mencionar que esta prueba se llevó a cabo en un automóvil. En primer lugar, se estableció la ubicación inicial del vehículo. Una vez obtenida esta ubicación, se definió un punto de destino a una distancia aproximada de 500 metros en línea recta. Luego, se puso en marcha el vehículo y se avanzó hacia el punto de destino. Durante este proceso, se pudo observar cómo la posición del vehículo se actualizaba en tiempo real en el mapa del dashboard. Cuando el vehículo se encontraba a una distancia de 10 metros del punto de destino, el sistema mecánico se detuvo automáticamente, tal como está programado en el software del vehículo. Se adjuntará un video en el CD que acompaña este documento para proporcionar una visualización más detallada de la prueba realizada.

Con esta prueba exitosa, procedemos a realizar la siguiente fase de pruebas, la cual se describe a continuación.

10.4 Prueba 3: Funcionamiento de ambos bloques en conjunto en un entorno acuático

En esta prueba, se buscará verificar el correcto funcionamiento del prototipo en su conjunto en un entorno acuático. El lugar elegido para llevar a cabo esta prueba se encuentra en la orilla del Río Dulce, específicamente en la intersección de Av. Solís y Costanera Nueva, justo al lado del puente Nuevo Solís "Juan Francisco Borges". Se adjuntan imágenes del lugar de la prueba:



Figura 115:Imagen satelital del lugar de prueba por Google Maps



Figura 116:Lugar de Prueba vista de cerca, Edición Propia



Figura 117: Lugar de Prueba vista superior, Edición Propia



Figura 118: Instalación de ambos bloques del prototipo, Edición Propia

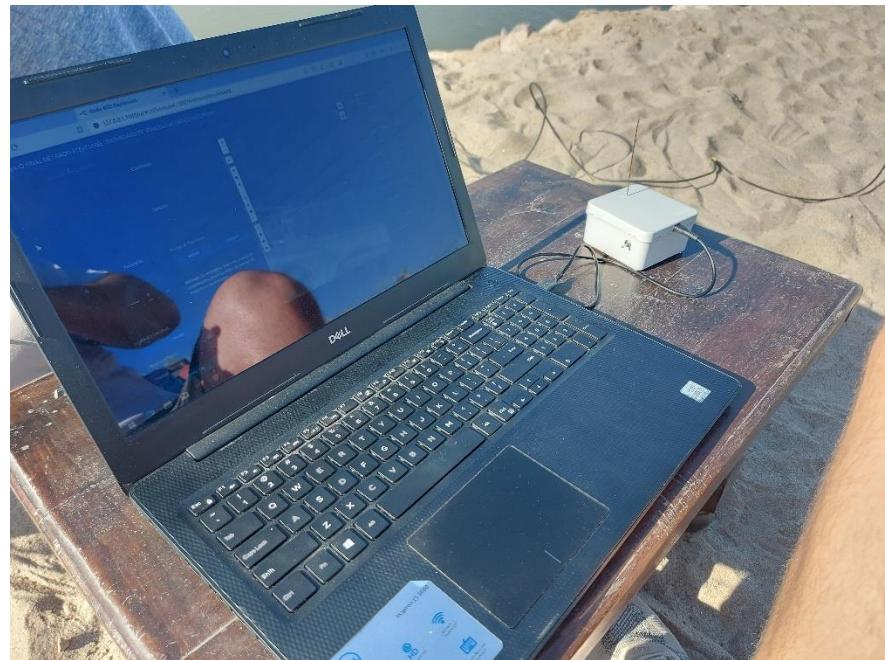


Figura 119: Bloque de tierra Firme, Edición Propia



Figura 120: Realizando la prueba 3, Edición Propia

Este escenario proporcionaba condiciones ideales para evaluar el rendimiento del vehículo en un entorno acuático y garantizar su funcionamiento integral en la interacción entre ambos componentes. Sin embargo, surgió un inconveniente durante la prueba debido a la corriente presente entre los espigones. Esta corriente hacía que el vehículo fuera arrastrado hacia la orilla, lo que resultaba en su desorientación y debido a la corriente no

podía orientarse nuevamente. Además, se notó que la batería utilizada no estaba en óptimas condiciones de funcionamiento. Debido a estas circunstancias, se decidió descartar la realización de la prueba en este entorno.

Para superar estos desafíos y adaptar la prueba a un entorno acuático adecuado, se optó por llevar a cabo la prueba en una piscina, donde no había corrientes fuertes y el viento no era lo suficientemente potente como para afectar el movimiento del vehículo. En esta nueva configuración, la prueba se llevó a cabo con éxito y se cumplieron todas las expectativas de funcionamiento del vehículo.

El video de la realización de la prueba estará en el CD que acompaña este informe. A continuación, se detallarán los procedimientos y resultados de esta importante prueba.

Se procedió a introducir el vehículo en el agua y se verificó que la comunicación con la estación de tierra firme funcionara correctamente. Luego, se estableció un punto de trayectoria desde la posición del vehículo hasta una distancia de 100 metros para evaluar su funcionamiento. Se habilitó el sistema mecánico y el vehículo comenzó a moverse de un punto a otro.

Una vez que el vehículo llegó al punto de destino, se realizó una verificación minuciosa para asegurarse de que no hubiera ingreso de agua en los compartimentos estancos. Afortunadamente, no se detectó la entrada de agua, lo que indicó el éxito tanto en el funcionamiento integral del prototipo como en su capacidad de resistencia al agua y flotación.

Esta prueba proporcionó resultados altamente satisfactorios y validó el desempeño del prototipo en un entorno acuático.

10.5 Autonomía

De las pruebas anteriores, se obtuvieron los siguientes consumos de energía. En la prueba 1, sin el sistema mecánico encendido, se registró un consumo de 100mA. En la prueba 3, con el motor funcionando a su máxima potencia y el servo motor realizando giros, se obtuvo un consumo de aproximadamente 550mA. A partir de estos valores, calcularemos la autonomía de la batería utilizada en el sistema.

Vamos a suponer que estamos realizando una ruta de 200 metros en total. En los primeros 100 metros, se deben realizar diferentes maniobras, se tendrá un consumo de 550mA y se completa en un tiempo de 20 minutos. En el segundo tramo de 100 metros, la ruta es en línea recta, lo que significa que el consumo de energía disminuye a 300 mA, y se completa en 40 minutos. En total, tenemos una hora para recorrer estos 200 metros. Además, se debe tener en cuenta una demora de 10 minutos para marcar los puntos de coordenadas y establecer adecuadamente la ruta a seguir, en este caso el consumo sería el mínimo rondando en unos 100 mA.

Ahora, vamos a realizar algunos cálculos para determinar el consumo y la autonomía en función de estos supuestos. Para ello, necesitamos considerar el consumo de energía en ambos tramos y en la configuración inicial del vehículo, luego calcular cuánto tiempo podríamos mantener este consumo antes de quedarnos sin energía.

En el primer tramo de 100 metros, con un tiempo de 20 minutos, podemos calcular el consumo de energía de la siguiente manera:

$$\begin{aligned} \text{Consumo} &= \text{Corriente (mA)} \times \text{Tiempo (h)} \\ \text{Consumo} &= \frac{550 \text{ mA} \times 20 \text{ minutos} \times 1 \text{ hora}}{60 \text{ minutos}} = 183.33 \text{ mAh} \end{aligned}$$

En el segundo tramo de 100 metros, con un tiempo de 40 minutos, el consumo de energía a 300 mA es:

$$\text{Consumo} = \frac{300 \text{ mA} \times 40 \text{ minutos} \times 1 \text{ hora}}{60 \text{ minutos}} = 200 \text{ mAh}$$

Obtenemos el consumo para los 10 minutos:

$$\text{Consumo} = \frac{100 \text{ mA} \times 10 \text{ minutos} \times 1 \text{ hora}}{60 \text{ minutos}} = 16 \text{ mAh}$$

Ahora, sumemos estos consumos para obtener el consumo total en la ruta:

$$\text{Consumo total} = 183.33 \text{ mAh} + 200 \text{ mAh} + 16 \text{ mAh} = 399.33 \text{ mAh}$$

Dado que tenemos un consumo total de 399.33 mAh y al tener una batería de 7000mAh tenemos:

$$\text{Autonomía} = \frac{\text{Capacidad de la fuente de energía (mAh)}}{\text{Consumo total (mAh)}}$$

$$\text{Autonomía} = \frac{\text{Capacidad de la fuente de energía (mAh)}}{399.33 \text{ (mAh)}}$$

$$\text{Autonomía} = \frac{7000 \text{ (mAh)}}{399.33 \text{ (mAh)}} \approx 17.52 \text{ horas}$$

Por lo tanto, con una fuente de energía de 7000 mAh, podríamos mantener el dispositivo en funcionamiento durante aproximadamente 17.52 horas para completar esta ruta de 200 metros, según las suposiciones dadas.

Además, hay que tener en cuenta que la batería es cargada en un periodo de 14hs con el panel que se tiene en este caso.

11 Capítulo 11: CONCLUSIONES

Se logró construir un prototipo que cumple con todos los requisitos establecidos en los objetivos de este proyecto. Se ensamblaron los dos bloques del sistema con las características deseadas, superando las expectativas previstas.

Se llevaron a cabo la selección y dimensionamiento adecuados de los componentes principales y complementarios del prototipo, y se rediseñó la estructura del vehículo para adaptarla a las necesidades específicas de este proyecto.

Se implementó una programación efectiva para el control de las velocidades y el movimiento del vehículo, garantizando su funcionamiento óptimo.

Se estableció un servidor para la recolección de información y se desarrolló una plataforma de visualización de datos y control del vehículo, lo que brinda una completa supervisión y gestión de las operaciones.

Se llevaron a cabo diversas pruebas exhaustivas para verificar el correcto funcionamiento del prototipo en su conjunto y de sus partes por separado, garantizando la calidad y fiabilidad del sistema.

Durante la ejecución de todos estos aspectos, se hizo uso de tecnologías de vanguardia, tanto en hardware como en software. Esto proporcionó la oportunidad de adquirir conocimientos y experiencia en la utilización de tecnologías de punta, como el diseño de software para microcontroladores mediante el sistema en tiempo real FreeRTOS, la tecnología LoRa para la comunicación inalámbrica de largo alcance y bajo consumo, así como el entorno de automatización por flujo Node-RED, que ha demostrado su utilidad en aplicaciones incluso en entornos industriales.

Si bien se satisficieron los objetivos, es posible seguir trabajando sobre el prototipo realizado para proporcionarle algunas mejoras o nuevas características como las siguientes:

Mejoras para el servidor:

- Implementar una Raspberry: Al implementar una computadora de placa simple tal como está, tenemos la capacidad de realizar las tareas que el ESP32 realizaba como Gateway y además la capacidad de ser el servidor que

contenga la programación en node-red y softwares adicionales dentro de un sistema compacto, transportable y de relativo bajo consumo.

- Implementación de base de datos: En lugar de generar archivos ejecutables en tablas de Excel con los datos de tensión y corriente (u adicionales se le agregan sensores para medir distintas variables), podría ser recomendable utilizar bases de datos ya que estas proveen recursos y respaldos que llevan a un nivel más alto de organización al sistema de gestión que se desarrolla en el servidor.

Mejoras para el Vehículo:

- Mejoramiento de las técnicas de control: A pesar de que la técnica utilizada (control proporcional) cumplió satisfactoriamente los objetivos, para escalar a un nuevo nivel de calidad y de confianza en el sistema de navegación es posible realizar modificaciones para usar técnicas de control automático más completas, tal como la utilización de un controlador PID (Proporcional Integral Derivativo), o bien, utilizar controladores más modernos como los controladores difusos que requieren aplicación de sistemas estadísticos y de lógica difusa o controles neuronales los cuales utilizan algoritmos de machine learning (redes neuronales) para realizar las acciones de control. Este último podría ser el más óptimo en la actualidad ya que posee la capacidad del aprendizaje para mejorar sus acciones.
- Utilización de más motores y de mayor potencia: Al utilizar más motores dispuestos de manera conveniente, los movimientos podrían llegar a ser más sencillos (por ejemplo, si se ponen dos motores, uno en cada lateral, en lugar de hacer marcha atrás para ir a un objetivo en un punto opuesto a su dirección,

este podrá rotar, acortando su desplazamiento y facilitando el movimiento en lugares más estrechos). Si los motores son más potentes, sería más fácil de implementar en ríos para andar en contra de la corriente y corregir los rumbos.

- Mejorar la construcción del vehículo: Si bien no era un objetivo a cumplir, el desarrollo de una estructura náutica más óptima para el vehículo mejora la capacidad de desplazamiento sobre el agua contra el oleaje y mejor comportamiento frente al viento.

BIBLIOGRAFIA

[1] Roger Martín Valls. Estudio, diseño, automatización y construcción de una embarcación RC impresa en 3D. Barcelona, Cataluña. 2017. Disponible en:

<https://upcommons.upc.edu/handle/2117/106521?show=full>

[2] Rubén Heras Zurita. Trabajo final de carrera para la obtención de la Diplomatura en Navegación Marítima “El sistema de propulsión del buque: las hélices marinas y el fenómeno de la cavitación”. Barcelona .2013. Disponible en:

<https://upcommons.upc.edu/handle/2099.1/20293>

[3] Tom Crosson, General Dynamics Mission Systems Celebrates 25 Years of Bluefin Robotics Innovation, 2019, Disponible en:

<https://gdmissionsystems.com/articles/2023/04/19/featured-news-celebrating-25-years-of-bluefin-robotics>

[4] Vehículos: ASV, AUV, ROV y otros, Barcelona, 2020 Disponible en:

<https://www.oceanografialitoral.com/pregunta-frecuente/vehiculos-asv-auv-rov-y-otros/>

[5] Diego Valdés Ruiz, Estado del Arte en robótica Submarina, Master en diseño y Fabricación de asistidos por computador. 2009-2010. Disponible en:

<https://riunet.upv.es/bitstream/handle/10251/59038/TFM%20-%20Vald%9Es%20Ruiz,%20D..pdf?sequence=1>

[6] Gabriel Carrasco Pagan, Evaluación y análisis de un AUV mediante CFD y estudio de alternativas operativas, Cartagena, 2016 Disponible en:

[efaidnbmnnibpcajpcglclefindmkaj/https://core.ac.uk/download/pdf/60434141.pdf](https://core.ac.uk/download/pdf/60434141.pdf)

[7] Instalan en Argentina la primera boya autónoma de monitoreo de lagunas, 2011. Disponible en:

https://www.ecoticias.com/naturaleza/46027_instalan-en-argentina-la-primeraboya-autonomia-de-monitoreo-de-lagunas

[8] Cecilia Farré, vehículos autónomos de exploración y vigilancia diseñados en la Argentina, 2023. Disponible en:

<https://www.opisantacruz.com.ar/2023/06/10/van-por-aire-tierra-y-agua-asi-son-los-vehiculos-autonomos-de-exploracion-y-vigilancia-disenados-en-la-argentina/>

[9] Alberto Pérez, Acuerdo entre Razer y ClearBot para crear un robot autónomo que limpie los desechos marinos. 2021. Disponible en:

https://www.hibridosyelectricos.com/coches/acuerdo-razer-clearbot-crear-robot-autonomo-que-limpie-desechos-marinos_45907_102.html

[10] ESP32 Technical Reference Manual, Disponible en:

https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf

[11] ESP32 Series Datasheet, Disponible en:

https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

[12] Canal Cartt thum, Proyecto de hélice para Barco RC, 2021. Disponible en:

https://www.youtube.com/watch?v=XTI-1fAyBX8&t=1187s&ab_channel=Carttthum

[13] Raúl Álvarez, Carro Robot Guiado por GPS: Arduino UNO en Acción, 2019. Disponible en:

<https://www.tecbolivia.com/index.php/articulos-y-tutoriales-microcontroladores/100-carro-robot-guiado-por-gps>

[14] Información oficial del Gobierno de los Estados Unidos relativa al Sistema de Posicionamiento Global y temas afines. Sistema de Posicionamiento Global. Disponible en:

<https://www.gps.gov/spanish.php>

12 ANEXO A Plano del Vehículo y Vistas 3D

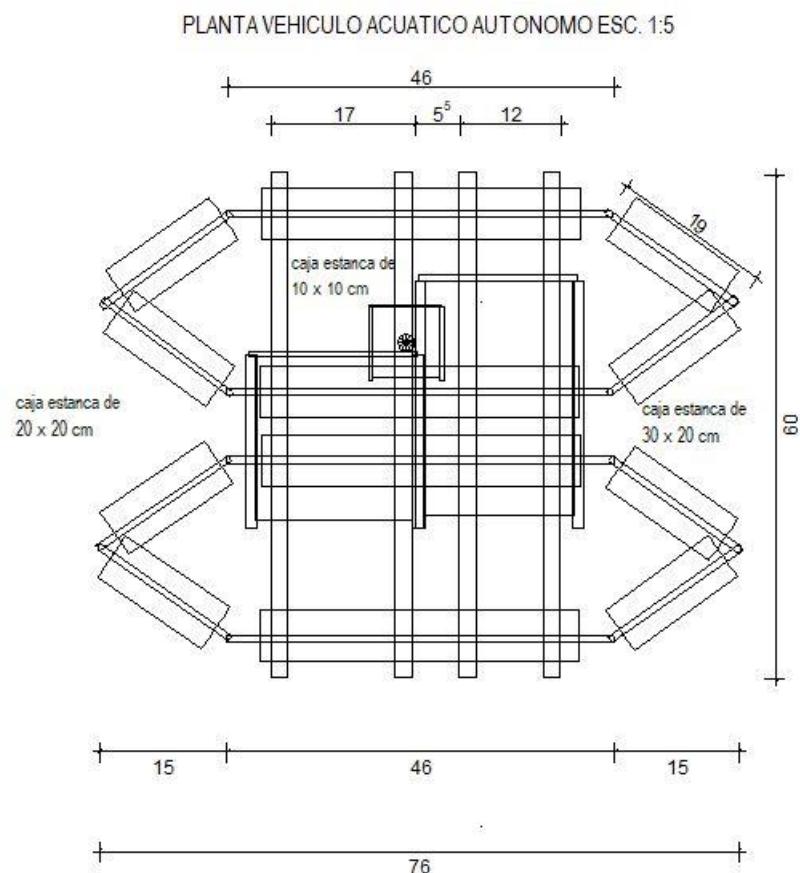


Figura 121: Planta del vehículo en escala 1:5, Edición Propia

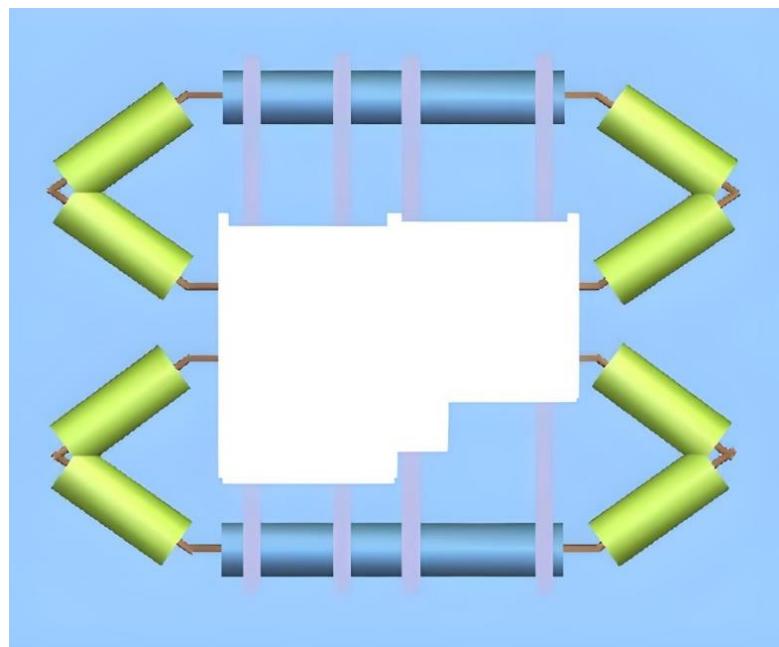


Figura 122: Vista Superior del vehículo, Edición Propria

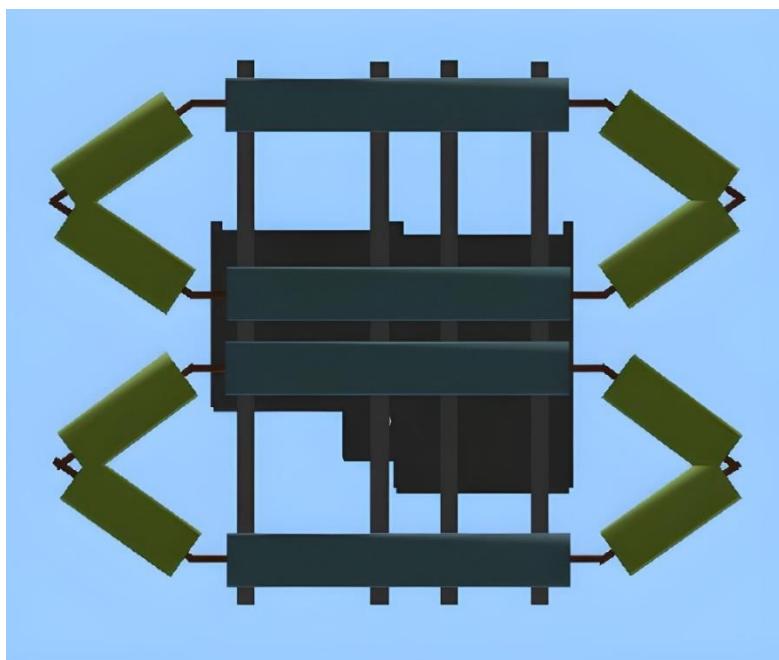


Figura 123: Vista Inferior del vehículo, Edición Propria

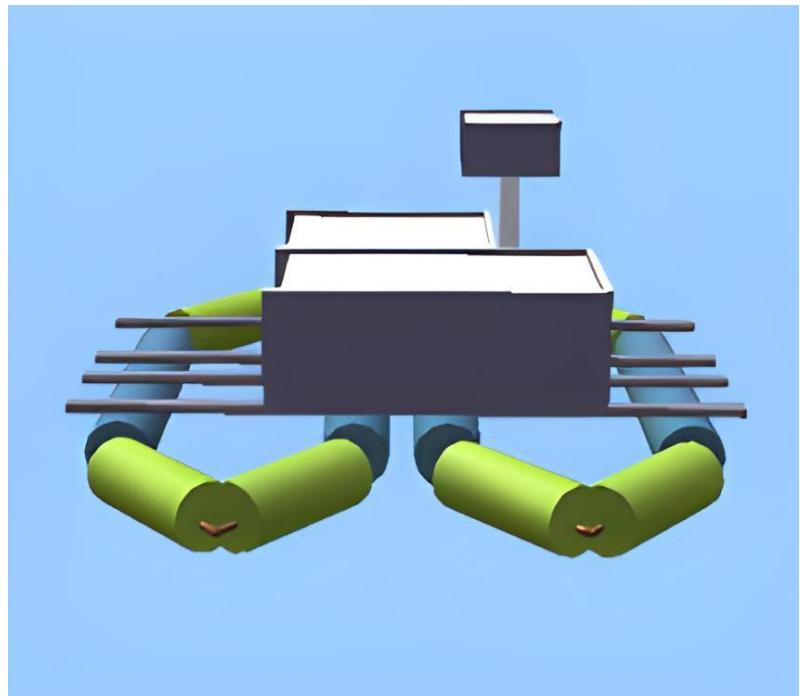


Figura 124: Vista Frontal del vehículo, Edición Propia

13 ANEXO B Esquemas de Conexión

Bloque Tierra Firme:

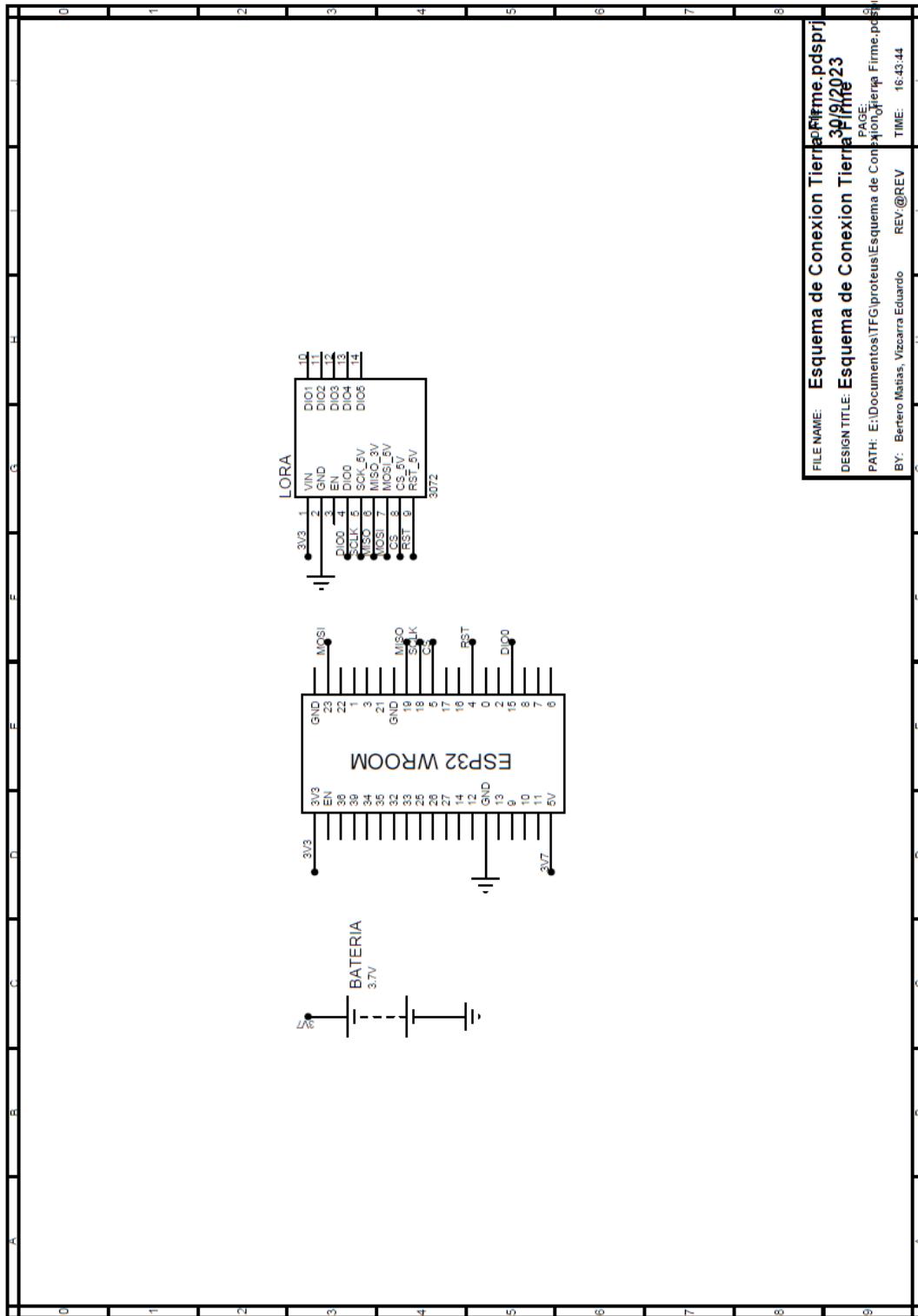


Figura 125: Esquema de conexión bloques tierra firme, Edición Propia

Bloque Vehículo:

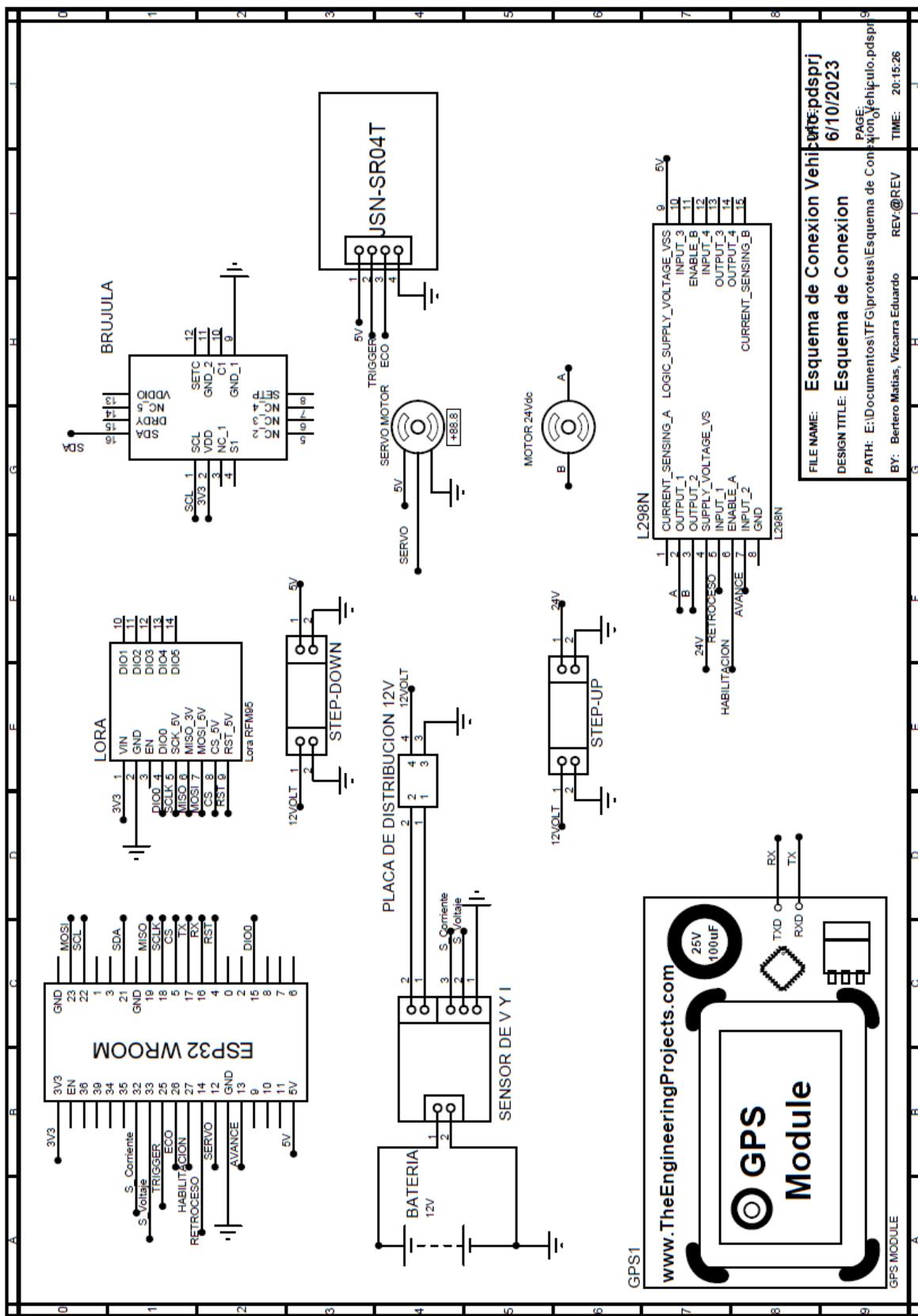


Figura 126: Esquema de conexión bloque vehículo, Edición Propia

Sensor de Tensión y corriente del Vehículo:

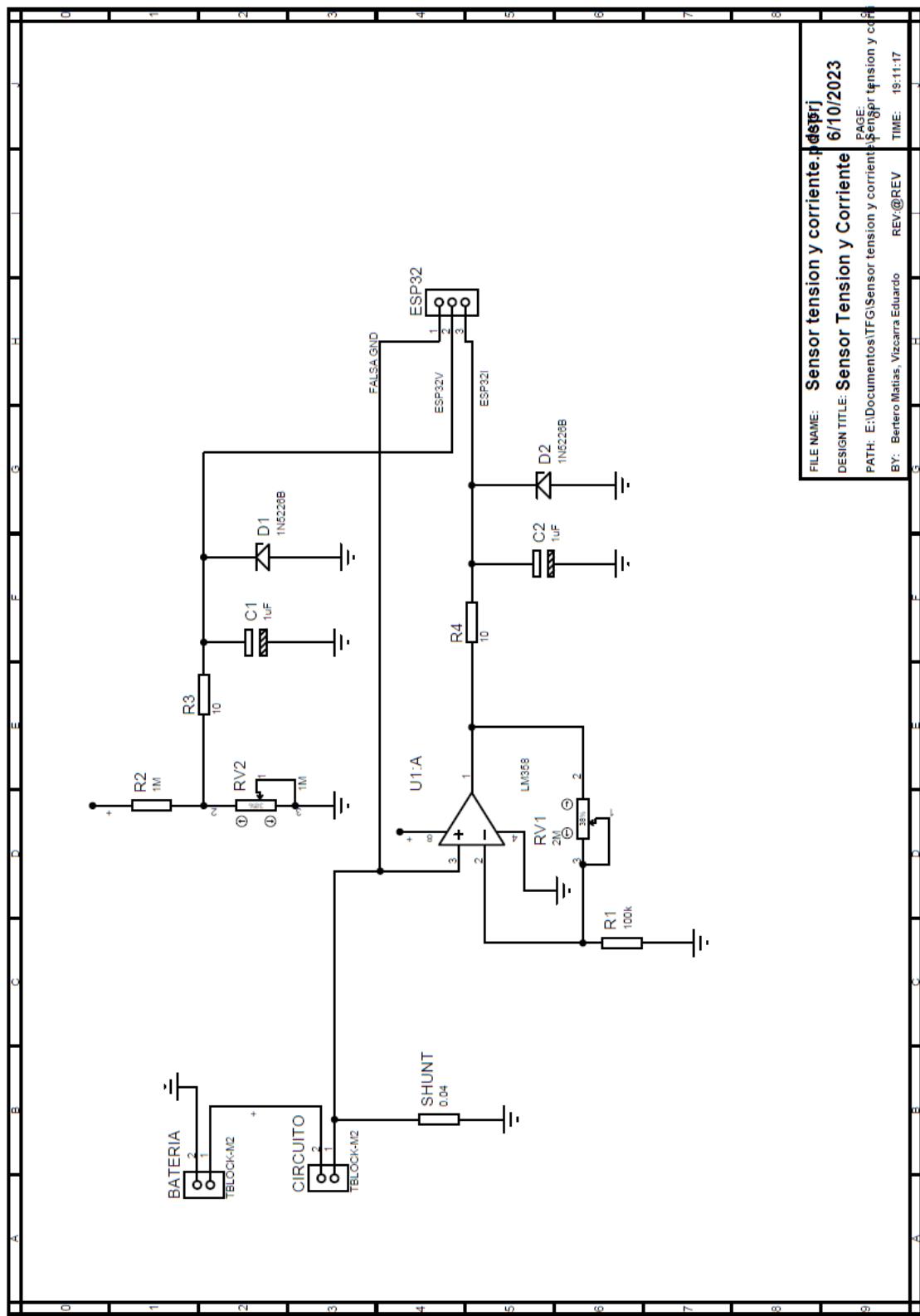


Figura 127: Esquema de conexión sensor de tensión y corriente del vehículo, Edición Propia

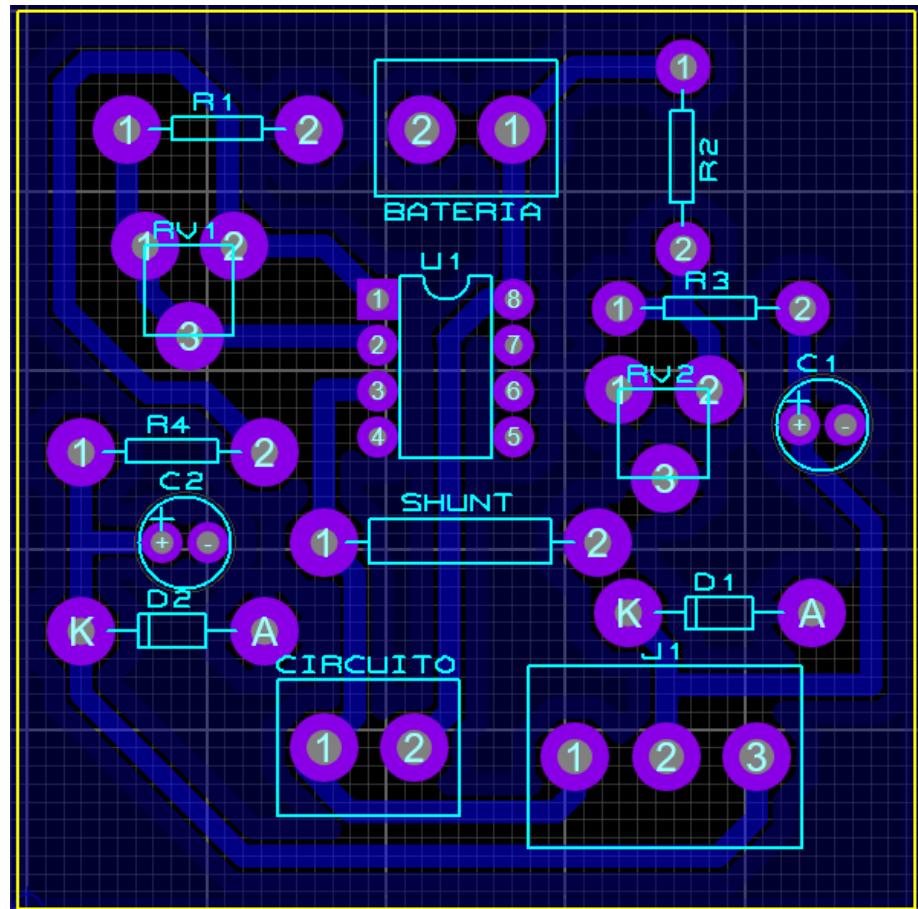


Figura 128: Diseño PCB del sensor de tensión y corriente, Edición Propia



Figura 129: Diseño Final del sensor, Edición Propia