UNIVERSIDAD TECNOLOGICA NACIONAL
Facultad Regional Bahía Blanca

Ingeniería Electrónica

Proyecto Final

SmartCaddy: Carro de Golf Automatizado

Autores

Sr. Dello Russo Facundo Andrés

Sr. Fidelibus Javier Romano Amaro

Sr. Muñoz Facundo

Tutores

Mg. Guillermo R. Friedrich

Ing. Adrián H. Laiuppa

Bahía Blanca, 14 de Febrero de 2024

Tabla de contenido

Resumen **3**
Introducción **4**
Objetivos **4**
Desarrollo **5**
ESP32 5
Control de Pantalla y Guardado de estadísticas: 6
Comunicación entre Dispositivos: 6
Principio de funcionamiento del guiado: 6
NRF 6
Frecuencia de Operación: 7
Principio de funcionamiento: 7
Conexión y Configuración: 8
GPS 9
Conexionado y configuración 12
BRÚJULA DIGITAL 13
Principio de funcionamiento: 13
SENSOR ULTRASÓNICO 15
Principio de funcionamiento: 15
Conexionado y configuración 16
PANTALLA TFT 17
Interfaz de Conexión: 18
Conexionado: 20
ENCODER ROTATIVO 20
Principio de funcionamiento 21
Conexionado y configuración 22
DISEÑO DE LA TRACCIÓN 24
CONSTRUCCIÓN DEL CADDY 25
Interfaz 25
Control interno: Receptor 31
Control Remoto: Emisor 34
Control de giro bidireccional 37
Conclusión **40**
Bibliografía y Referencias **41**

Resumen

Este proyecto se centra en el diseño, construcción y programación de un carro de golf que no solo permite a los golfistas controlar sus estadísticas, mejoras y hándicap en cualquier campo de golf del país, sino que también ofrece la capacidad de desplazarse de manera autónoma hacia la ubicación del golfista mediante el uso de coordenadas GPS.

El objetivo principal de este proyecto es mejorar la movilidad del golfista, permitiéndole concentrarse plenamente en su juego y obtener mejores resultados. Esto se logra eliminando la necesidad de cargar con el carro que contiene los palos de golf y otros accesorios, lo que reduce significativamente la fatiga acumulada durante la partida.

Además de proporcionar comodidad en el transporte, nuestro proyecto también busca facilitar la gestión continua del progreso del golfista, brindándole una experiencia más eficiente y enfocada en el juego. Con esta innovación, aspiramos a crear una solución integral que mejore la experiencia del golfista en el campo de golf.

Palabras Claves: Usuario – Carro – Automatización – GPS – NRF – Estadísticas

Introducción

En el contexto de la electrónica y la ingeniería, surge un proyecto que fusiona la tecnología con el deporte: el desarrollo de un carro de golf automático. Este dispositivo, dotado de un sistema de guiado hacia la posición donde se encuentra el golfista y un sistema de seguimiento de juego, representa una innovación en el campo de la movilidad y jugabilidad como también así en el análisis de datos.

La premisa central es simple: un usuario presiona un botón, el carro de golf automático entra en acción, moviéndose de forma autónoma hacia las coordenadas del usuario. Esta automatización busca eliminar la tarea manual de transportar el equipo, y no sólo aporta comodidad, sino que también redefine la experiencia de juego aumentando el rendimiento y concentración del usuario.

Acompañando esta función de desplazamiento autónomo, el carro de golf también integra un sistema de seguimiento de juego, que permite al usuario registrar todos los datos del juego. Al finalizar la partida, los datos recopilados se transforman en estadísticas y se guardan en una memoria SD para que el usuario las pueda visualizar en cualquier momento. Esta característica no solo proporciona un análisis retrospectivo de la actuación del golfista, sino que también facilita la planificación y la mejora continua del juego para futuras partidas.

Este informe, explorará el proceso de diseño, desarrollo e implementación del carro de golf automático. Desde los componentes electrónicos hasta su aplicación práctica en el campo, se abordan los desafíos técnicos, las soluciones implementadas y los beneficios que este proyecto aporta al usuario.

Este estudio representa un paso más allá en la integración de la electrónica con actividades cotidianas, demostrando cómo la tecnología puede mejorar y simplificar incluso los aspectos más arraigados en la tradición de los deportes.

Objetivos

Este proyecto tiene como objetivo principal mejorar la experiencia de los golfistas al ofrecer un carro de golf automático único en su clase. El objetivo es proporcionar una solución integral que no solo mejore la movilidad en el campo de golf, sino que también eleve la experiencia de juego a través de la integración de la tecnología.

El foco está puesto en cubrir las necesidades del usuario, priorizando la retroalimentación con el mismo, para garantizar actualizaciones y adaptaciones que reflejen sus demandas. Con este enfoque, se aspira a no solo ganar clientes, sino a establecer una relación duradera como parte integral de la experiencia de los golfistas en el campo.

Los objetivos claves del proyecto se basan en:

- Mejora de la Movilidad
- Optimización de la Experiencia de Juego
- Facilitación de la Gestión del Progreso
- Innovación en el Análisis de Datos
- Integración de Tecnología en el Deporte

Desarrollo

El diseño y construcción del carro de golf automático se fundamenta en la integración cuidadosa de diversos componentes, cada uno desempeñando un papel crucial en la funcionalidad global del equipo. A continuación, se proporciona una visión general de estos elementos clave, destacando sus funciones individuales y su contribución al proyecto en su conjunto.

ESP32

Para el desarrollo y programación del proyecto se utilizaron 3 módulos de DOIT ESP32 DEVKIT V1 (DOIT, 2022) basado en el microcontrolador ESP32-WROOM-32 (Espressif Systems, 2023), los cuales presentan las siguientes características:

- Microcontrolador: Equipados con una CPU Tensilica Xtensa LX6 de dos núcleos de 32 bits, estos módulos ofrecen una potencia de procesamiento eficiente para las aplicaciones del proyecto.
- Pines de E/S Digitales (DIO): Con un total de 25 pines de E/S digitales, proporcionan una amplia flexibilidad para la conexión y el control de componentes externos.
- Pines de Entrada Analógica (ADC): Incorporan 6 pines de entrada analógica.
- Pines de Salida Analógica (DAC): Cuentan con 2 pines de salida analógica, los cuales no se utilizaron en el desarrollo del proyecto.
- Interfaces de Comunicación: Ofrecen 3 UART, 2 SPI y 3 I2C, lo que facilita la comunicación con una variedad de dispositivos externos y periféricos.
- Memoria Flash y SRAM: Poseen una memoria Flash de 4MB y una SRAM de 520KB, proporcionando un amplio espacio de almacenamiento para el firmware y los datos del proyecto.
- Velocidad del reloj: Funcionan a una velocidad de reloj de 240 megaciclos, lo que garantiza un rendimiento óptimo en aplicaciones exigentes.
- Wi-Fi IEEE 802.11: Equipados con conmutador TR integrado, balun, LNA, amplificador de potencia y red correspondiente, estos módulos ofrecen conectividad Wi-Fi IEEE 802.11 b/g/n/e/i, lo que permite la comunicación inalámbrica con otros dispositivos y redes.
- Fuente de Alimentación: La alimentación del ESP32 DEVKIT V1 se puede suministrar a través del conector USB Micro B integrado o directamente a través del pin "VIN". Selecciona automáticamente la fuente de alimentación entre una alimentación externa de 6 a 20 voltios, con un rango recomendado de 7 a 12 voltios para evitar daños en el dispositivo debido al sobrecalentamiento del regulador de voltaje. El cual no se utilizó en el desarrollo del proyecto.

Estas características hacen de la ESP32 DEVKIT V1 una opción ideal como columna vertebral electrónica para el proyecto, brindando potencia, flexibilidad y conectividad para su implementación.

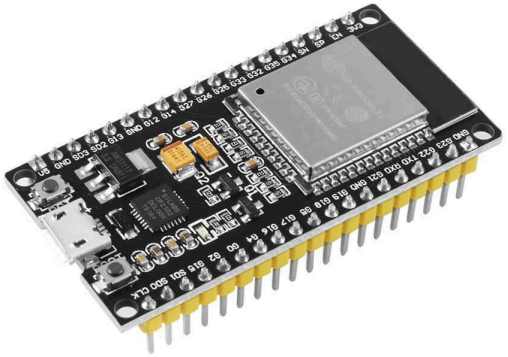

Figura 1: DOIT ESP32 DEVKIT V1. Fotografía de elaboración propia

Control de Pantalla y Guardado de estadísticas:

Una de las ESP32 se encarga de gestionar la interfaz de usuario del carro, controlando la pantalla y registrando las estadísticas de juego. La capacidad de procesamiento rápida y eficiente garantiza una experiencia de usuario fluida y la preservación precisa de los datos esenciales.

Comunicación entre Dispositivos:

Las otras dos ESP32 trabajan en conjunto para facilitar la comunicación esencial entre el golfista y el carro. El golfista porta un dispositivo equipado con una ESP32 que transmite datos de posición al segundo módulo dentro del carro. Esta comunicación bidireccional establece un enlace crucial entre el usuario y el vehículo, permitiendo un desplazamiento autónomo preciso y receptivo. La ESP32 en el carro no solo recibe datos de posición, sino que también funciona como el cerebro de la operación. Con la capacidad de procesar y analizar datos de GPS, calcular distancia, rumbo y obstáculos, esta unidad guía el desplazamiento del carro y controla con precisión los motores eléctricos, asegurando un movimiento suave y eficiente por el campo de golf.

Principio de funcionamiento del guiado:

El guiado del carro se fundamenta en la obtención del ángulo de azimuth de dos puntos cardinales. Cuando el usuario activa la función "Follow" en su dispositivo portátil, este transmite sus coordenadas actuales al carro de golf. Este último cuenta con un segundo GPS que adquiere sus propias coordenadas. Utilizando estos dos conjuntos de coordenadas, se calcula el ángulo de azimuth mediante una ecuación específica. Posteriormente, con los datos obtenidos del magnetómetro instalado en el carro, se compara el ángulo del vehículo con el del objetivo. Se realiza una corrección continua hasta que ambos ángulos coinciden, lo que permite al carro avanzar hacia las coordenadas del usuario. Una descripción más detallada de este proceso se proporcionará en secciones posteriores.

NRF

Para la comunicación entre las dos placas ubicadas dentro del dispositivo del usuario y la otra dentro del carrito se utilizó el módulo NRF24L01 (Nordic Semiconductor, 2007).

El módulo NRF24L01 es un dispositivo de comunicación inalámbrica de corto alcance que opera en la banda de frecuencia de 2.4 GHz.

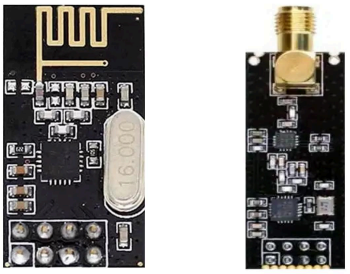

Figura 2: NRF24L01. Fotografía de elaboración propia.

A continuación, sus características principales:

Frecuencia de Operación:

El NRF24L01 opera en la banda de 2.4 GHz, utilizando múltiples canales para evitar interferencias con otros dispositivos que operan en la misma frecuencia. Esto lo hace ideal para entornos donde la interferencia electromagnética es una consideración importante. El módulo transmite y recibe datos en una frecuencia específica conocida como canal. Para que dos o más módulos se comuniquen entre sí, deben estar en el mismo canal. Este canal puede tener cualquier frecuencia en la banda ISM de 2,4 GHz, o más precisamente, cualquier frecuencia entre 2.400 y 2.525 GHz (2400 a 2525 MHz). Esto significa que el nRF24L01+ puede funcionar en 125 canales diferentes.

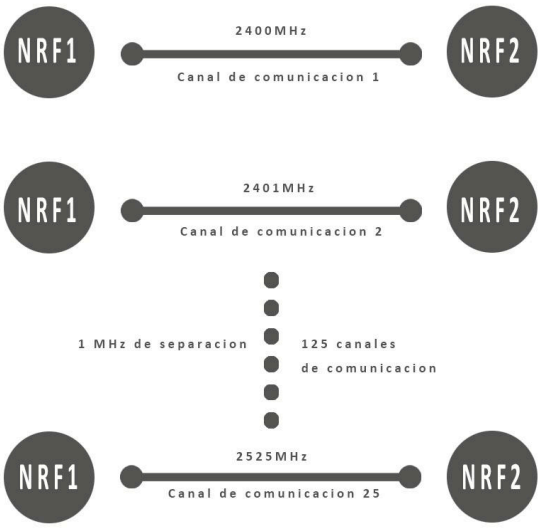

Figura 3: Canales del NRF. Gráfico de elaboración propia.

Para esta aplicación da la posibilidad de un futuro, poder trabajar con varios carros de golf en una misma cancha, y que no haya interferencias de comunicación debido a la elección de distintos canales.

Principio de funcionamiento:

Este módulo facilita la comunicación bidireccional entre dispositivos, permitiendo tanto la transmisión como la recepción de datos. Esta capacidad es esencial para aplicaciones en las que se requiere una interacción continua entre nodos, como es en este caso donde un NRF se utiliza como transmisor y el otro como receptor. El transmisor inicia la comunicación enviando un paquete de datos al receptor. Después de transmitir el paquete, el transmisor espera aproximadamente 130 µs hasta que llegue el acuse de recibo (ACK). El receptor envía el ACK después de recibir exitosamente el paquete. Una vez que el transmisor recibe el ACK, la transacción finaliza.

El alcance efectivo del NRF24L01 puede variar según las condiciones del entorno y la configuración, pero condiciones favorables y con configuraciones optimizadas, los módulos NRF24L01 pueden alcanzar una distancia efectiva de comunicación de hasta 100 metros o más.

Este módulo admite velocidades de transferencia de datos variables, con opciones que van desde 250 kbps hasta 2 Mbps. La capacidad de ajustar la velocidad permite adaptar la comunicación para este uso donde lo único que interesa es tener el mayor alcance posible por lo que se setea en 250kbps.

El NRF24L01 está diseñado para ser eficiente en términos de consumo de energía, lo que lo hace adecuado para dispositivos alimentados por baterías. Al depender de una batería de litio la conservación de energía es crucial. La potencia de salida del módulo se puede programar para que sea 0 dBm, -6 dBm, -12 dBm o -18 dBm. A 0 dBm, el módulo consume sólo 12 mA durante la transmisión, que es menos que el consumo de un solo LED. Su consumo es de sólo 26 µA en modo de espera y 900 nA en modo apagado. Por eso es el dispositivo inalámbrico ideal para aplicaciones de bajo consumo.

Conexión y Configuración:

El módulo se configuró y controló mediante una interfaz SPI (Serial Peripheral Interface), lo que proporciona flexibilidad y facilidad de integración con la plataforma de programación y el microcontrolador. Todos los parámetros, incluido el canal de frecuencia (125 canales seleccionables), la potencia de salida (0 dBm, -6 dBm, -12 dBm o -18 dBm) y la velocidad de datos (250 kbps, 1 Mbps o 2 Mbps), se pueden configurar a través de la interfaz SPI. El mismo se conecta de la siguiente manera:

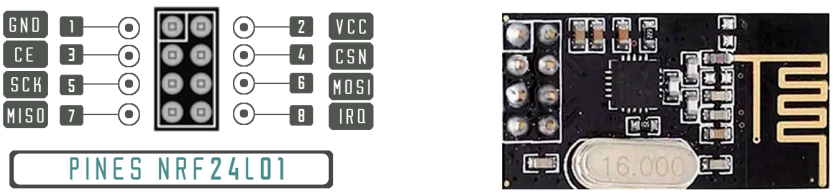

Figura 4: Pines de conexión. Fotografía de elaboración propia.

- **GND**: es el pin de tierra. Tiene una marca cuadrada para distinguirlo de los demás pines.

- **VCC**: suministra energía al módulo. Puede oscilar entre 1,9 y 3,9 voltios.
- **CE**: (habilitación de chip) es un pin activo-alto. Cuando está habilitado, el nRF24L01 transmitirá o recibirá, según el modo.
- **CSN**: (no selección de chip) es un pin activo-bajo que normalmente se mantiene ALTO. Cuando este pin baja, el nRF24L01 comienza a escuchar datos en su puerto SPI y los procesa en consecuencia.
- **SCK**: (reloj serie) Acepta pulsos de reloj del bus maestro SPI.
- **MOSI**: (Master Out Slave In) es la entrada SPI para el nRF24L01.
- **MISO**: (Maestro en esclavo fuera) es la salida SPI del nRF24L01.
- **IRQ**: Es un pin de interrupción que puede notificar al maestro cuando hay nuevos datos para procesar.

El bus SPI utiliza el concepto de maestro y esclavo. En este proyecto, el ESP32 actúa como maestro y el módulo nRF24L01+ actúa como esclavo. A diferencia del bus I2C, el bus SPI tiene un número limitado de esclavos. Por lo tanto, puede utilizar hasta dos esclavos SPI (dos módulos nRF24L01+) en un solo Arduino.

Código:

El código utilizado para el uso del mismo contiene la librería *nRF24L01.h* y dentro del mismo haciendo uso principalmente de las siguientes sentencias:

```
//// INICIO TRANSMISOR NRF ////
gpsSerial.begin(GPSBaud);
radio.begin();
radio.openReadingPipe(1, 0xF0F0F0F0E1LL);
radio.setDataRate(RF24_250KBPS);
radio.setPALevel(RF24_PA_MAX);
radio.setChannel(0x76);
radio.startListening();


if (radio.available()) {
   radio.read(&gpsGolfista, sizeof(dataStruct));
    }
```

Figura 5: Codificación en lenguaje C. Fotografía de elaboración propia.

GPS

En la implementación del sistema de posicionamiento para el SmartCaddy, se ha seleccionado la utilización de dos módulos GPS basado en el chip NEO-6M (Ublox, 2011), el primero ubicado en el carro de golf y el segundo en el dispositivo del usuario.

Figura 6: GPS NEO 6M Fotografía de elaboración propia.

Especificaciones Técnicas:

- Voltaje de Alimentación: 3.3 V ~ 5.0 V.
- Consumo de Corriente: ~ 100 mA.
- Rendimiento GPS:
    - Recepción: 50 Canales; GPS L1 frequency, C/A Code, SBAS: WAAS, EGNOS, MSAS.
    - Tiempo de Enganche Promedio: 1 s ~ 27 s.
    - Sensibilidad:
        - Tracking & Navegación: -161 dBm.
        - Reacquisition: -160 dBm.
        - Inicio en Frío: -147 dBm.
        - Inicio en Caliente: -156 dBm.
        - Tasa de Actualización: 5 Hz.
        - Precisión de Posicionamiento: 2.5 MCEP.
- Antena:
    - Tipo de Antena: Cerámica.
    - Ganancia de la Antena: 5 dB.
    - Dimensiones: 22 x 22 mm.
    - Longitud del Cable: 20 mm.
- Protocolo de Comunicación: NMEA, UBX binario.
- Velocidad de Comunicación: 9600 baudios.
- EEPROM Dedicada para Almacenamiento de Datos de Configuración.
- LEDs Indicadores de Estado

A continuación, sus principales características:

Receptores de Satélite:

Los receptores de satélite permiten la conexión con múltiples satélites para obtener señales de posición confiables. Los receptores pueden rastrear hasta 22 satélites simultáneamente a través de 50 canales, con una sensibilidad de seguimiento de -161 dB, mientras mantienen un consumo de corriente eficiente de 45 mA.

Bajo Consumo de Energía:

Este chip posee el modo de ahorro de energía (PSM). Esto permite una reducción en el consumo de energía del sistema al encender y apagar selectivamente ciertas partes del receptor, reduciendo drásticamente el consumo de energía del módulo a solo 11ma.

Indicador LED de posición fija

Hay un LED en el módulo GPS NEO-6M que indica el estado de la 'Fijación de posición'. Parpadeará a diferentes velocidades según el estado en el que se encuentre:

- Sin parpadear: está buscando satélites.
- Parpadea cada 1 segundo: se encuentra en posición fija (el módulo puede ver suficientes satélites).

Figura 7: Led indicador de satélites. Fotografía de elaboración propia.

Batería y EEPROM

El módulo también posee una pila de botón recargable que actúa como supercondensador.

La EEPROM y la batería juntas ayudan a retener la BBR (RAM respaldada por batería). BBR contiene datos de reloj, datos de posición más recientes (datos de órbita GNSS) y configuración del módulo. Pero no es para el almacenamiento permanente de datos.

La batería se carga automáticamente cuando se suministra energía al módulo y retiene los datos durante dos semanas sin energía.

Dado que la batería retiene el reloj y los datos de la última posición, el tiempo hasta el primer arreglo (TTFF) se reduce significativamente a 1 segundo. Esto permite bloqueos de posición mucho más rápidos. Sin batería el GPS siempre arranca en frío y tarda más en el bloqueo inicial del GPS.

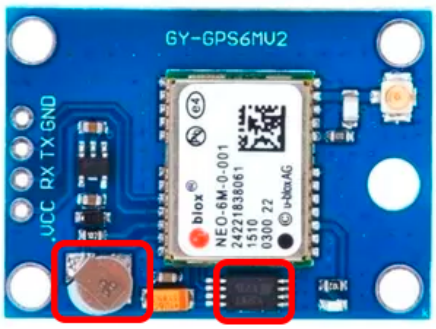

Figura 8: Batería recargable y EEPROM. Fotografía de elaboración propia.

Antena

El módulo viene con una antena de parche de sensibilidad de -161 dBm para recibir señales de radio de satélites GPS. Se conecta al módulo mediante un conector UFL.

Figura 9: Antena cerámica. Fotografía de elaboración propia.

Conexionado y configuración

El módulo GPS NEO-6M tiene un total de 4 pines que lo conectan con el mundo exterior. Las conexiones son las siguientes:

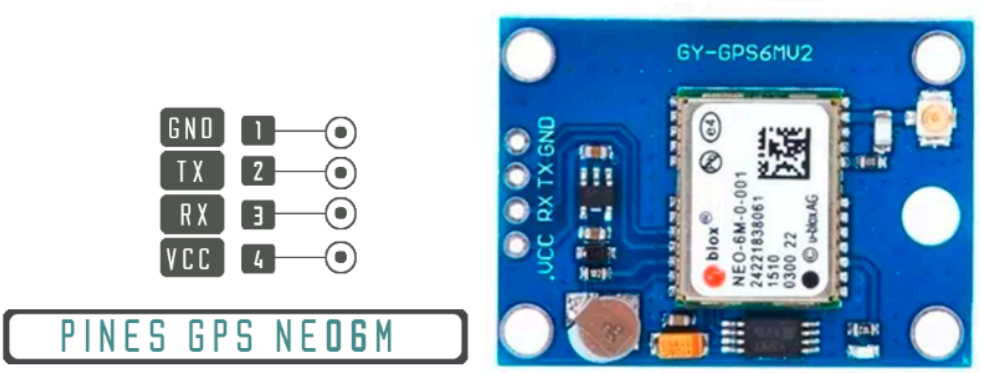

Figura 10: Pines de conexión. Fotografía de elaboración propia.

- **Tierra**: es el pin de tierra y debe conectarse al pin GND de la ESP.
- **TxD**: (Transmisor) El pin se utiliza para la comunicación en serie.
- **RxD**: (receptor) El pin se utiliza para la comunicación en serie.
- **VCC**: suministra energía al módulo. Se puede conectar directamente al pin de la ESP.

Código:

Partiendo de la librería libre TinyGPSPlus.h se utilizaron de la siguiente manera en el código:

```
//// INICIO DEFINICION GPS ////
static const int RXPin = 3, TXPin = 1;
static const uint32_t GPSBaud = 9600;

TinyGPSPlus gps;

//Conexion Serial para el GPS
SoftwareSerial gpsSerial(RXPin, TXPin);


if (gps.location.isValid())
{
  Serial.println("Coordenadas Carro: ");
  Serial.print("Latitud: ");
  Serial.println(gps.location.lat(),6);
  gpsCarro.latitude = gps.location.lat();

  Serial.print("Longitud: ");
  Serial.println(gps.location.lng(), 6);
  gpsCarro.longitude = gps.location.lng();
}
else
{
  gpsCarro.longitude = 0.0;
  gpsCarro.latitude = 0.0;
  Serial.println("Location: Not Available");
}
delay(500);
```

Figura 11: Codificación en lenguaje C. Fotografía de elaboración propia.

BRÚJULA DIGITAL

En la implementación del proyecto también se incorporó la brújula digital HMC5883 (Honeywell, 2010) como un componente fundamental para proporcionar una navegación precisa y una orientación confiable en el campo de juego. Esta brújula digital se erige como una herramienta crucial para determinar el rumbo real y actual del carro.

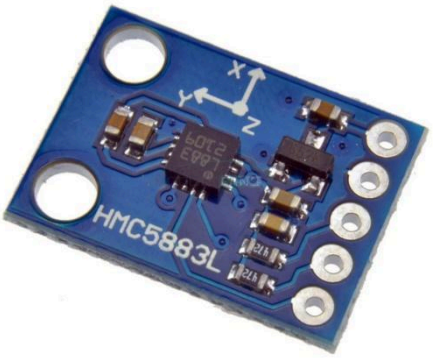

Figura 12: Magnetómetro HMC5883. Fotografía de elaboración propia.

Principio de funcionamiento:

EL HMC5883L es un magnetómetro de 3 ejes, el cual permite leer las componentes del campo magnético presente, de esta forma conociendo la dirección del campo magnético terrestre, se puede calcular la orientación con respecto al norte magnético de la tierra, esto siempre y cuando el sensor no esté expuesto a algún campo magnético externo u algún objeto metálico que altere el campo magnético terrestre. Se tuvo en cuenta la proximidad inicial del sensor con los motores del carro, los cuales generaban un campo magnético que alteraba los resultados obtenidos por el sensor.

Conexionado y configuración

Este pequeño módulo HMC5883L incorpora tres sensores de magnetorresistencia, cancelación de desfases, y conversores de 12 bits, lo que le proporciona una precisión de ±2ºC, trabaja con una tensión de 1.8 a 3.3V, pero cuenta con un regulador interno por lo que se puede alimentar con 5v. Este módulo de Honeywell HMC5883L, se comunica a través del bus I2C, cuya dirección fija es 0x1E, por lo que es sencillo obtener los datos medidos.

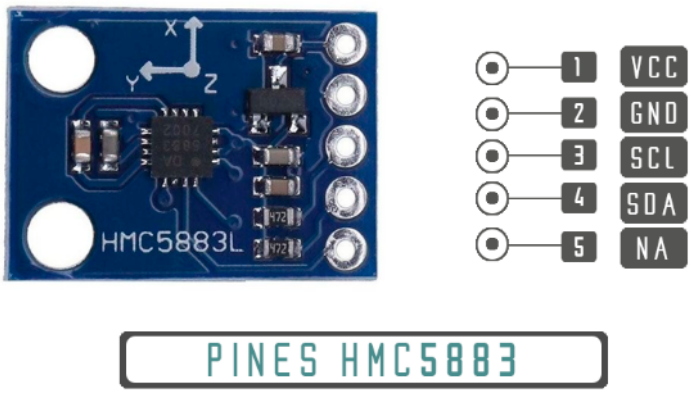

Figura 13: Pines y conexionado. Fotografía de elaboración propia.

- **VCC**: Se conecta a la fuente de alimentación de 3.3V o 5V.
- **GND**: tierra del sistema.
- **SCL:** (Serial Clock) Este pin lleva la señal de reloj del protocolo I2C. Es esencial sincronizar las transmisiones y recepciones de datos entre el módulo y el microcontrolador.
- **SDA:** (Serial Data) Este pin transmite los datos entre el módulo y el microcontrolador a través del protocolo I2C. La información, como las lecturas de la brújula, se envía y recibe a través de este canal.
- **DRDY:** (Sensor de Inclinación) Este pin se usa para recibir una señal de interrupción cuando los nuevos datos están listos para ser leídos o para configurar la dirección I2C del módulo.

Código:

El código realizado estuvo vinculado a la librería compass.h, de la cual primero se le realizó una calibración inicial al sensor donde no solo se tenía en cuenta la posición geográfica actual sino también los campos magnéticos a los cuales se ve

afectada la misma y así tener valores lo más certeros posible. Se inicializa el sensor con esos valores obtenidos de la siguiente manera.

```
//// INICIO BRUJULA ////
  compass.init();
  compass.setCalibration(-461, 690, -323, 890, 0, 261);
//// FIN BRUJULA ////
```

Luego se hace uso de la función compass.read() para obtener el valor de azimuth actual. El cual más tarde se utilizará para contrastarlo con el obtenido por los GPS.

```
void QMC5883LCompass::read(){
    Wire.beginTransmission(_ADDR);
    Wire.write(0x00);
    int err = Wire.endTransmission();
    if (!err) {
        Wire.requestFrom(_ADDR, (byte)6);
        _vRaw[0] = (int)(int16_t)(Wire.read() | Wire.read() << 8);
        _vRaw[1] = (int)(int16_t)(Wire.read() | Wire.read() << 8);
        _vRaw[2] = (int)(int16_t)(Wire.read() | Wire.read() << 8);

        if ( _calibrationUse ) {
            _applyCalibration();
        }

        if ( _smoothUse ) {
            _smoothing();
        }

        //byte overflow = Wire.read() & 0x02;
        //return overflow << 2;
    }
}


    int QMC5883LCompass::getAzimuth(){
        int a = atan2( getY(), getX() ) * 180.0 / PI;
        return a < 0 ? 360 + a : a;
    }
```

Figura 14: Codificación en lenguaje C. Fotografía de elaboración propia.

SENSOR ULTRASÓNICO

Para la detección de obstáculos y evitar que el carro colisione en su trayecto al usuario, se añade un sensor de ultrasonidos HCSR04 (Cytron technologies, 2013). Su funcionamiento se basa en el envío de un pulso de alta frecuencia, no audible por el ser humano. Este pulso rebota en los objetos cercanos y es reflejado hacia el sensor, que dispone de un micrófono adecuado para esa frecuencia.

Midiendo el tiempo entre pulsos, conociendo la velocidad del sonido, se estima la distancia del objeto contra cuya superficie impacta el impulso de ultrasonidos

Principio de funcionamiento:

El sensor se basa simplemente en medir el tiempo entre el envío y la recepción de un pulso sonoro. Sabiendo que la velocidad del sonido es 343 m/s en condiciones de temperatura 20 ºC, 50% de humedad, presión atmosférica a nivel del mar. Transformando unidades resulta

$$343\frac{m}{s}\cdot 100\frac{cm}{m}\cdot\frac{1}{1000000}\frac{s}{\mu s}=\frac{1}{29.2}\frac{cm}{\mu s}$$

Ecuación 1: Cálculos previos. Fotografía de elaboración propia.

Es decir, el sonido tarda 29,2 microsegundos en recorrer un centímetro. Por tanto, se puede obtener la distancia a partir del tiempo entre la emisión y recepción del pulso mediante la siguiente ecuación.

$$Distancia(cm)=\frac{Tiempo(\mu s)}{29.2\cdot 2}$$

Ecuación 2: Cálculo final de distancia. Fotografía de elaboración propia.

El motivo de dividir por dos el tiempo (además de la velocidad del sonido en las unidades apropiadas) es porque se mide el tiempo que tarda en ir y volver el pulso, por lo que la distancia recorrida por el pulso es el doble de la que se desea medir.

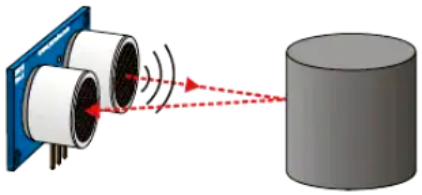

Figura 15: Funcionamiento básico. Fotografía de elaboración propia.

Precisión:

El rango de medición teórico del sensor HC-SR04 es de 2cm a 400 cm, con una resolución de 0.3cm. En la práctica, sin embargo, el rango de medición real es mucho más limitado, en torno a 20cm a 2 metros, en esta ocasión se adapta perfectamente ya que solo lo utilizara para evitar colisiones de más de 30cm de distancia.

Conexionado y configuración

El sensor está ubicado en la parte delantera del carro para poder detectar los obstáculos mucho antes de que cualquier parte del carro colisione.

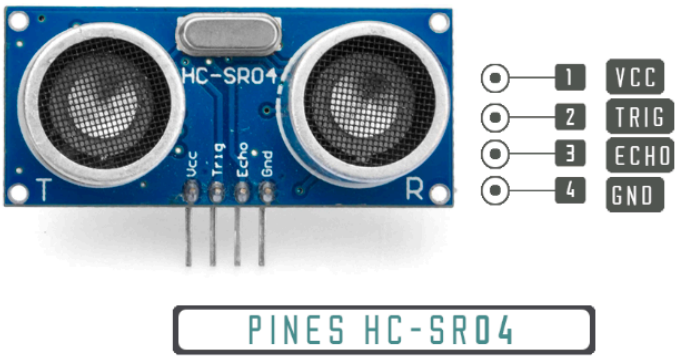

Ilustración 16: Pines y conexionado. Fotografía de elaboración propia.

- **VCC**: Se conecta a la fuente de alimentación de 5V.
- **GND**: tierra del sistema.
- **TRIG:** El pin TRIG es el encargado de iniciar la medición de distancia. Cuando este pin recibe un pulso de señal de corta duración (generalmente de 10 microsegundos), el sensor emite un pulso ultrasónico. Es el encargado de "disparar" el pulso ultrasónico para iniciar el proceso de medición de distancia.
- **ECHO:** El pin ECHO es utilizado para medir la duración del tiempo que tarda en llegar el eco de vuelta al sensor después de emitir el pulso ultrasónico. Cuando el sensor recibe el eco, el pin ECHO produce un pulso de duración proporcional al tiempo que tarda el pulso ultrasónico en viajar hacia el objeto y regresar.

Código:

Para el código del mismo, se inicializan los pines como entradas y salida y se realizan los cálculos que se establecieron previamente por lo que queda el siguiente código.

```
//// INICIO HCSR04 ////
 pinMode(trig, OUTPUT);
 pinMode(echo, INPUT);
//// FIN HCSR04 ////


//Inicio Funcion Sensor Ultrasonico.//
void fobstaculo(){
  digitalWrite(trig, HIGH);
  delay(1);
  digitalWrite(trig, LOW);

  duracion= pulseIn(echo, HIGH);
  obstaculo= (duracion/58.2);
  Serial.print(obstaculo);
  Serial.println("cm");

}
//Fin Funcion Sensor Ultrasonico.//
```

Ilustración 17: Codificación en lenguaje C. Fotografía de elaboración propia.

PANTALLA TFT

Para la interfaz del usuario y poder hacer un correcto seguimiento del juego tanto previo como presente, utilizamos una pantalla TFT de 2.8” SPI 240x320 basado en el chip ILI9341 (ILITEK, 2010).

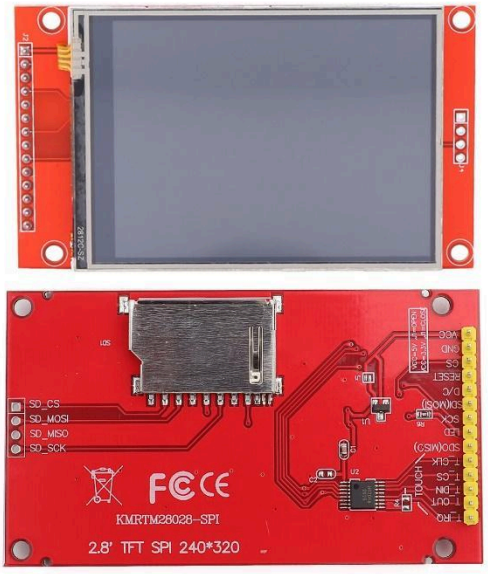

Figura 18: Pantalla TFT 2,8" SPI 240x320. Fotografía de elaboración propia.

La pantalla TFT ILI9341 utiliza la tecnología de transistores de película delgada (TFT).

Especificaciones técnicas:

- Tamaño: Serie SPI de 2.4 "
- Área de visualización: 36.72 mm (W) X 48.96 (H) mm
- Elemento del controlador: matriz activa a-Si TFT
- Disposición de píxeles: banda vertical RGB
- IC del controlador: ILI9341
- Luz de fondo: LED blanco
- Dirección de visualización: 6 en punto
- Profundidad del color: 262K / 65K
- Resolución (puntos): 240px * 320px
- Voltaje de alimentación 3.3V

Interfaz de Conexión:

Utiliza protocolos estándar como SPI (Serial Peripheral Interface) para la comunicación con la ESP32.

El bus SPI es un bus de interfaz comúnmente utilizado para enviar datos entre microcontroladores y pequeños periféricos, sensores y tarjetas SD. Utiliza líneas separadas de reloj y datos, junto con un pin selección llamado CS (chip select) para elegir el dispositivo con el que desea hablar.

En concreto, un bus SPI necesitará de 3 pines de datos y otro pin de control. Esas señales se denominan de la siguiente forma:

- **MISO:** Entrada
- **MOSI:** Salida

- **CLOCK:** Señal de reloj (también llamada SCK o CLK)
- **CS:** Chip select

En un bus SPI, solo un lado genera la señal de reloj (CLOCK). El lado que genera el reloj se llama "controlador", y el otro lado se llama "periférico". Solo puede haber un controlador (en este caso es el microcontrolador), pero puede haber múltiples periféricos conectados al mismo bus.

Cuando los datos se envían desde el controlador a un periférico, se envían en una línea de datos llamada MOSI, para "Salida controlador/ Entrada periférico". Si el periférico necesita enviar una respuesta al controlador, el controlador continuará generando un ciclo de reloj y el periférico colocará los datos en una tercera línea de datos llamada MISO, para "Entrada controlador/ salida periférico".

Una forma resumida un esquema de cómo funciona el bus SPI en el siguiente gráfico:

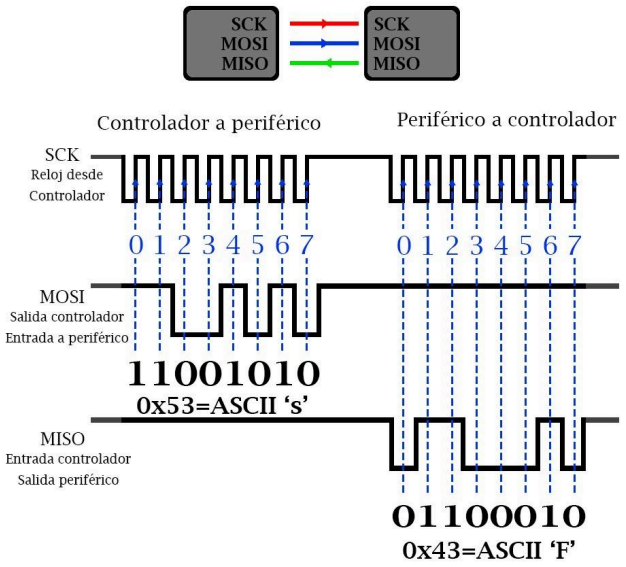

Figura 19: Protocolo de comunicación SPI. Gráfico de elaboración propia.

Solo puede haber un solo controlador o maestro, pero puede haber varios periféricos (esclavos). Entonces si los periféricos están conectados al mismo bus, el microcontrolador los identificará con el pin extra llamado **CS** (Chip Select).

Se puede decir que es la única "desventaja" como tal del bus SPI, ya que cada periférico necesitará un pin único para ser controlado dentro del bus. Aparte de los pines MISO, MOSI y CLOCK, necesitamos un pin extra por cada periférico SPI que se utiliza, como es el caso de la memoria SD.

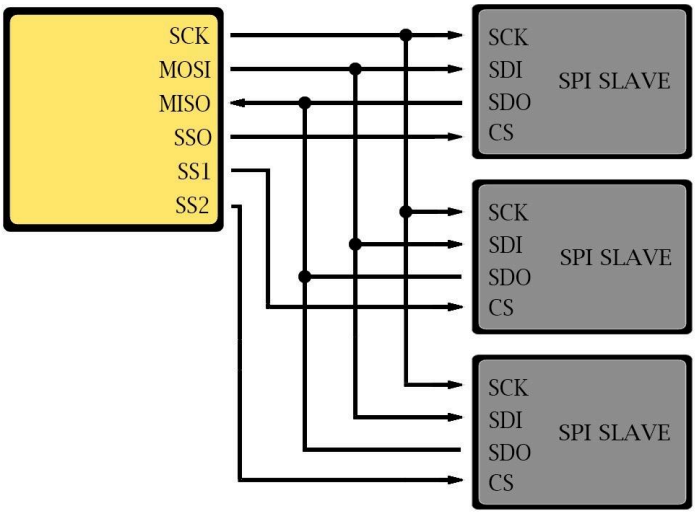

Figura 20: Comunicación Master/Slave. Gráfico de elaboración propia.

En conclusión, a pesar de las limitaciones que puede presentar la comunicación SPI, como la necesidad de un pin adicional por cada dispositivo, la ausencia de control de errores y acuse de recibo, y su alcance limitado a cortas distancias, este protocolo se adapta a las necesidades específicas del proyecto. La velocidad notablemente rápida, la capacidad bidireccional (full-dúplex) y la compatibilidad generalizada con la mayoría de los microcontroladores son aspectos que se resaltan positivamente y hacen que la elección de SPI sea adecuada para la implementación de esta solución, brindando eficiencia y eficacia en la comunicación entre los componentes del sistema.

Conexionado:

La pantalla requiere una alimentación de 3.3v y se utilizaran los siguientes pines con la comunicación que se explicó con anterioridad, además los pines que no se utilizan son para la pantalla touch, de la cual no se hizo uso debido a que no pareció práctico para el usuario.

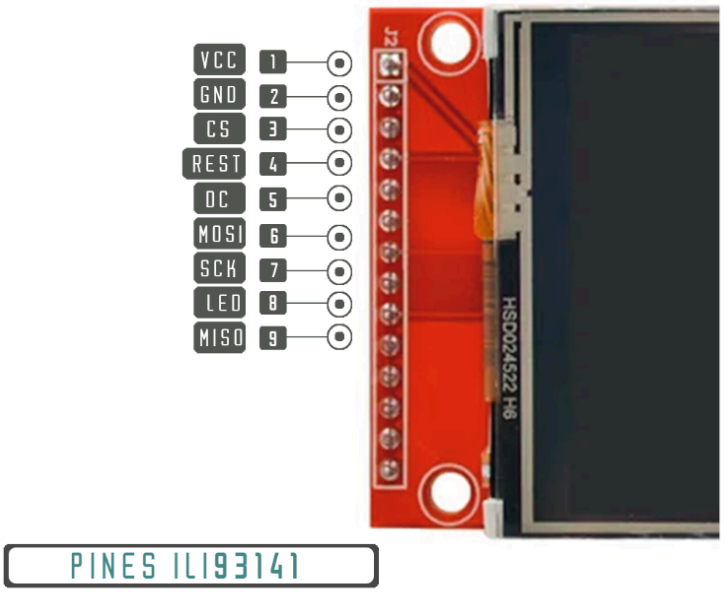

Figura 21: Pines y conexionado. Fotografía de elaboración propia.

ENCODER ROTATIVO

Se decidió conectar al usuario con la pantalla mediante un encoder rotativo, debido a que con tan solo tres movimientos se puede navegar libremente en todo el menú y sus opciones. Se tiene la posibilidad de moverse a la izquierda, derecha y presionar el encoder para elegir el menú seleccionado. A continuación, se explicará con más detalle el encoder utilizado.

Un codificador rotatorio es un tipo de sensor de posición que convierte la posición angular (rotación) de una perilla en una señal de salida que se puede usar para determinar en qué dirección se gira la perilla.

Los codificadores rotatorios se clasifican en dos tipos: absolutos e incrementales. El codificador absoluto informa la posición exacta de la perilla en grados, mientras que el codificador incremental informa el número de incrementos que se ha movido el eje. El codificador rotatorio que se utilizó en este proyecto es del tipo incremental Encoder KY-040 (JOY IT, 2017)

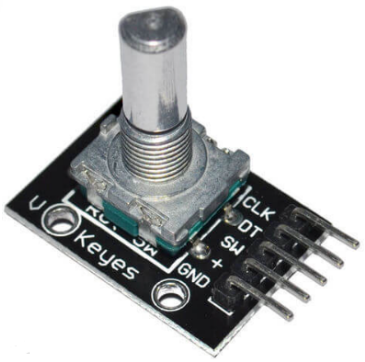

Figura 22: Encoder rotativo KY-040. Fotografía de elaboración propia.

Principio de funcionamiento

Dentro del codificador hay un disco ranurado que está conectado al pin C, la tierra común. También tiene dos pines de contacto A y B, como se muestra a continuación.

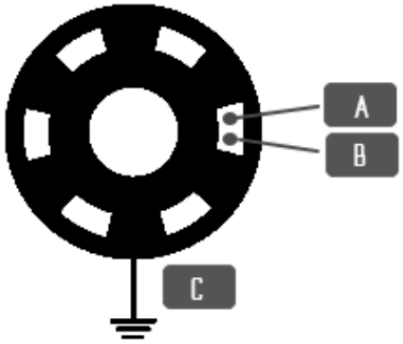

Figura 23: Pines de contacto. Fotografía de elaboración propia.

Cuando se gira la perilla, A y B hacen contacto con el pin de tierra común C en un orden específico dependiendo de en qué dirección gire la perilla.

Cuando hacen contacto con un terreno común, se generan dos señales. Estas señales están desfasadas 90° porque un pin hace contacto con tierra común antes que el otro. Se le conoce como codificación en cuadratura.

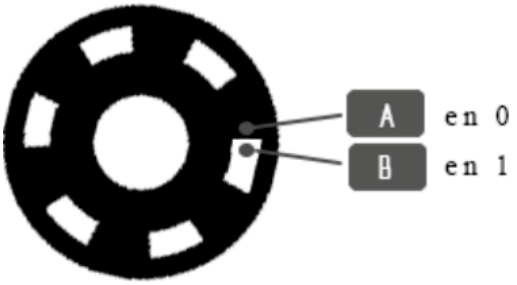

Figura 24: Detección de pulsos. Fotografía de elaboración propia.

Cuando se gira la perilla en el sentido de las agujas del reloj, el pin A se conecta a tierra antes que el pin B. Cuando se gira la perilla en sentido antihorario, el pin B se conecta a tierra antes que el pin A.

Al monitorear cuándo cada pin se conecta o desconecta de tierra, podemos determinar la dirección en la que se gira la perilla. Esto se puede lograr simplemente observando el estado de B cuando cambia el estado de A.

Cuando A cambia de estado:

- sí B! = A, entonces la perilla se giro en el sentido de las agujas del reloj.

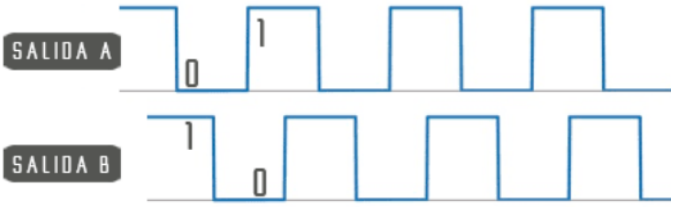

Ilustración 25: Pulsos de A y B. Gráfico de elaboración propia.

- sí B = A, la perilla se giro en sentido antihorario.

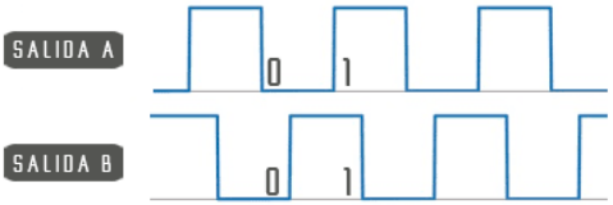

Ilustración 26: Pulsos de A y B. Gráfico de elaboración propia.

Conexionado y configuración

La distribución de pines del módulo codificador rotatorio es la siguiente:

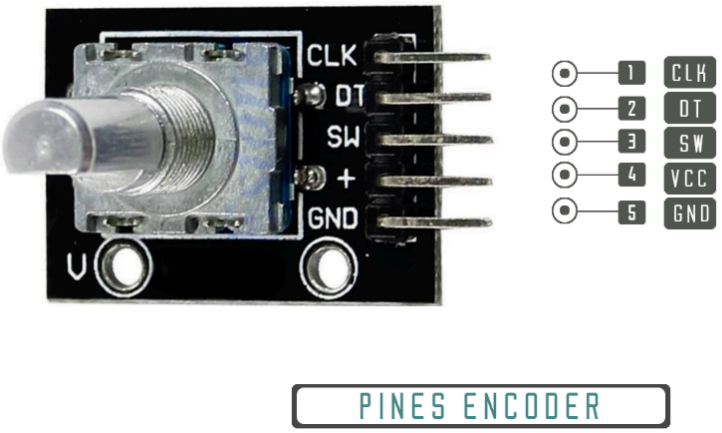

Figura 27: Pines y conexionado. Fotografía de elaboración propia.

- **GND**: es la conexión a tierra.
- **VCC**: es el voltaje de suministro positivo, que normalmente está entre 3,3 y 5 voltios.
- **SW**: es la salida del interruptor de botón (activo bajo). Cuando se presiona la perilla, el voltaje pasa a BAJO.
- **DT**: (Salida B) es similar a la salida de CLK, pero está por detrás de CLK en un cambio de fase de 90°. Esta salida se utiliza para determinar la dirección de rotación.
- **CLK**: (Salida A) es el pulso de salida principal utilizado para determinar la cantidad de rotación. Cada vez que se gira la perilla en cualquier dirección con

solo un retén (clic), la salida 'CLK' pasa por un ciclo de paso ALTO y luego BAJA.

Código:

Para el código del mismo se detectaron los movimientos para saber el lado al cual giraba el encoder de la siguiente manera:

```
void ezButton::loop(void) {

    int currentState = digitalRead(btnPin);
    unsigned long currentTime = millis();

    if (currentState != lastFlickerableState) {

        lastDebounceTime = currentTime;

        lastFlickerableState = currentState;
    }

    if ((currentTime - lastDebounceTime) >= debounceTime) {

        previousSteadyState = lastSteadyState;
        lastSteadyState = currentState;
    }

    if(previousSteadyState != lastSteadyState){
        if(countMode == COUNT_BOTH)
            count++;
        else if(countMode == COUNT_FALLING){
            if(previousSteadyState == HIGH && lastSteadyState == LOW)
                count++;
        }
        else if(countMode == COUNT_RISING){
            if(previousSteadyState == LOW && lastSteadyState == HIGH)
                count++;
        }
    }
}
```

Figura 28: Codificación en lenguaje C. Fotografía de elaboración propia.

El botón que presiona hacia abajo se lo trata como un switch normal donde se detecta el cambio de estado del mismo y de ser así se realizan las acciones solicitadas por el usuario.

La detección del giro en el código está implementado con el uso de interrupciones para que siempre se tome como prioridad. En cambio, el pulsado del botón, está como loop dentro del main ya que no se necesita tanta velocidad de lectura como lo es el giro.

```
void IRAM_ATTR ISR_encoder() {
  if ((millis() - last_time) < 60)
    return;

  if (digitalRead(DT_PIN) == HIGH) {

     if(menu==1)page1 --;
     if(menu==2)page2 --;
     if(menu==3) page3--;
     if(menu==3 && page1==0)handicap --;
     if(menu==4)page4 --;
     if(menu==5)page5 --;
     if(menu==6)tiros--;
     if(menu==7)numPartida2--;
  } else {
     if(menu==1)page1 ++;
     if(menu==2)page2 ++;
     if(menu==3 && page1==0)handicap ++;
     if(menu==3) page3++;
     if(menu==4)page4 ++;
     if(menu==5)page5 ++;
     if(menu==6)tiros++;
     if(menu==7)numPartida2++;
  }
  last_time = millis();
}
```

Figura 29: Codificación en lenguaje C. Fotografía de elaboración propia.

DISEÑO DE LA TRACCIÓN

El diseño de tracción seleccionado para este proyecto se centra en la implementación de un sistema de tracción independiente para dos ruedas traseras, cada una impulsada por un motor respectivo. Este enfoque brinda un control preciso y maniobrabilidad mejorada al vehículo.

Configuración para Movimiento Recto:

Para avanzar en línea recta, ambos motores operan en la misma dirección, proporcionando una fuerza propulsora equilibrada que impulsa al carro hacia adelante de manera eficiente.

Cuando se requiere realizar giros, el diseño permite dos modos distintos:

- **Giro hacia la Izquierda:** El motor izquierdo opera en reversa, mientras que el motor derecho avanza en la dirección habitual. Esta configuración induce un giro hacia la izquierda, permitiendo al carro cambiar de dirección de manera suave y controlada.
- **Giro hacia la Derecha:** En el caso de girar hacia la derecha, el proceso es opuesto al giro hacia la izquierda. El motor izquierdo avanza en su dirección normal, mientras que el motor derecho se pone en reversa. Esta inversión de fuerzas provoca que el vehículo gire hacia la derecha de manera eficaz.

Este diseño ofrece una solución práctica para lograr movimientos precisos y maniobras ágiles. La combinación de tracción independiente y la capacidad de controlar la dirección mediante la variación de dirección de los motores proporciona flexibilidad en la operación del carro.

CONSTRUCCIÓN DEL CADDY

Para la construcción y armado del carro, se utilizó uno ya existente y se modificó para que se adapte a las nuevas necesidades.

Interfaz

En cuanto al diseño de la interfaz, se utilizó la pantalla mencionada anteriormente junto con un encoder. Este conjunto se montó sobre una placa de circuito diseñada para facilitar su posterior ensamblaje. En esta placa, se incorporó un indicador del nivel de batería del carro, para lo cual se implementó un divisor de tensión que permite medir la entrada de hasta 3V proveniente de la ESP32.

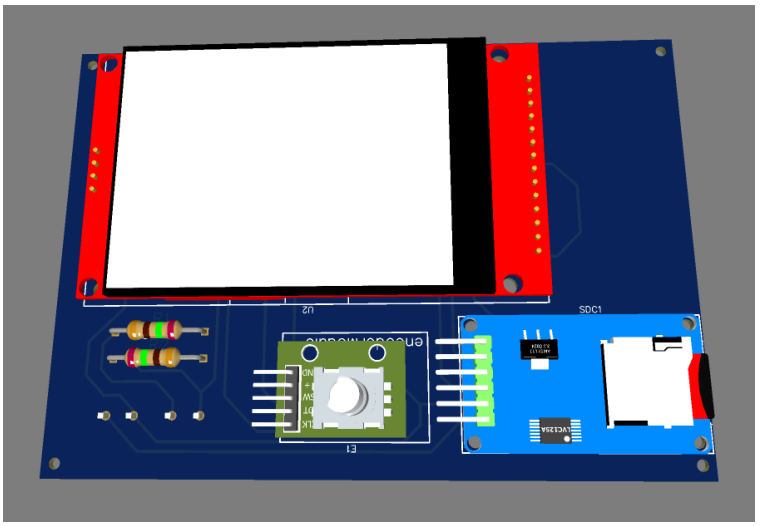

Figura 30: Diseño 3D del control de interfaz. Fotografía de elaboración propia.

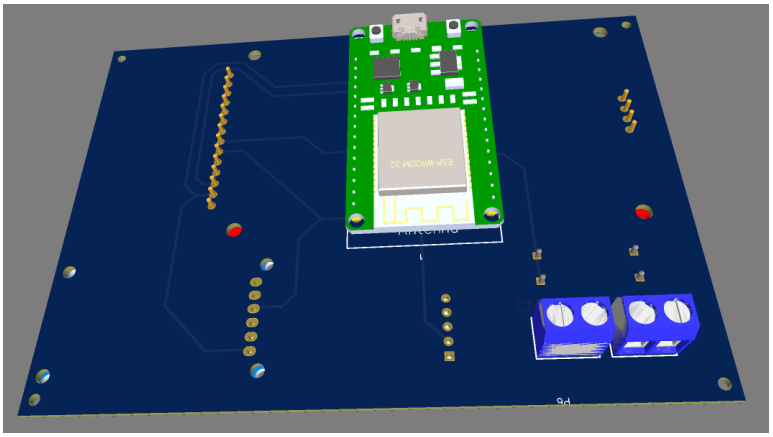

Figura 31: Parte trasera. Fotografía de elaboración propia.

Toda la soportería y carcaza fue diseñada por el grupo exclusivamente para este proyecto. A continuación, imágenes de la interfaz finalizada y montada.

Figura 32: Montaje final pantalla encoder y botones. Fotografía de elaboración propia.

El menú cuenta con distintas opciones visuales, como lo son el cambio de tema de la pantalla entre CLARO y OBSCURO. También permite hacer un seguimiento hoyo por hoyo, en cualquier cancha del país, la cual se selecciona previo a jugar, para finalmente guardar todas las estadísticas en una memoria SD. A continuación un recorrido por estas opciones:

Figura 33: Menú Principal. Fotografía de elaboración propia.

Figura 34: Menú Personalización. Fotografía de elaboración propia.

Figura 35: Menú Edición Hándicap. Fotografía de elaboración propia.

Figura 36: Selección de Par. Fotografía de elaboración propia.

Figura 37: Estadísticas Finales. Fotografía de elaboración propia.

Figura 38: Selección de cancha. Fotografía de elaboración propia.

Figura 40: Historial Guardado en SD. Fotografía de elaboración propia.

Figura 39: Cantidad de Hoyo. Fotografía de elaboración propia.

En esta memoria SD se encuentra tanto el Handicap, que puede modificarse en cualquier momento desde la pantalla, y también todas las partidas guardadas. Estas últimas también pueden verse una por una desde el menú de la pantalla.

El código para el control de la misma incluye muchas partes, pero las siguientes son las más importantes:

```
166
167⊟   if (button.isReleased()) {
168⊞     if(menu==0){
173⊞     if(menu==1){
179⊞     if(menu==2){
217⊞     if(menu==3){
238⊞     if(menu==4){
250⊞     if(menu==5){
255⊞     if(menu==6){
310⊞     if(menu==7){
319   }
320   }
```

Figura 41: Resumen de todos los menús disponibles. Fotografía de elaboración propia.

```
330⊟  if(menu==1){
331⊞              if (page1==0){
343⊟              if (page1==1){
344               limpiarPantalla();
345               tft.drawXBitmap(287, 7, LBateria , 28, 15, letras);
346               leerBateria();
347               tft.setTextColor(letras);
348               tft.drawXBitmap(14, 93, LhandicapChico , 51, 53, letras);
349               tft.drawXBitmap(120, 64, Ljugar , 80, 82, letras);
350               tft.drawXBitmap(255, 93, LhistorialChico , 51, 53, letras);
351               tft.setFreeFont(CF_OL32);
352               tft.setCursor (62, 200);
353               tft.print("   JUGAR");
354               }
355⊟              if (page1==2){
356               limpiarPantalla();
357               limpiarArriba();
358               tft.drawXBitmap(287, 7, LBateria , 28, 15, letras);
359               leerBateria();
360               tft.setTextColor(letras);
361               tft.drawXBitmap(14, 93, LjugarChico , 51, 53, letras);
362               tft.drawXBitmap(120, 64, Lhistorial , 80, 82, letras);
363               tft.drawXBitmap(255, 93, LajustesChico , 51, 53, letras);
364               tft.setFreeFont(CF_OL32);
365               tft.setCursor (62, 200);
366               tft.print("HISTORIAL");
367               }
368⊟              if (page1==3){
369               limpiarPantalla();
370               limpiarArriba();
371               tft.drawXBitmap(287, 7, LBateria , 28, 15, letras);
372               leerBateria();
373               tft.setTextColor(letras);
374               tft.drawXBitmap(14, 93, LhistorialChico , 51, 53, letras);
375               tft.drawXBitmap(120, 64, Lajustes , 80, 82, letras);
376               tft.drawXBitmap(255, 93, LhandicapChico , 51, 53, letras);
377               tft.setFreeFont(CF_OL32);
378               tft.setCursor (62, 200);
379               tft.print("AJUSTES");
380               }
381   }
```

Figura 42: Detalle del menú 1, submenú 1. Fotografía de elaboración propia.

Para el cambio de menú, se sortearon varias dificultades de actualización de pantalla y problemas a la hora de la sobreescritura. Para finalmente quedar de la siguiente forma:

```
565  if(menu==3){
566        if(pagel==0 && page2==0){
567               tft.setFreeFont(CF_OL24);
568               tft.setTextSize(1);
569               tft.setCursor (30, 60);
570               tft.setTextColor(letras);
571               tft.print("NUEVO HANDICAP");
572               tft.fillRect (125, 90,110,90,background);
573               tft.setFreeFont(CF_OL32);
574               tft.setTextSize(2);
575               tft.setCursor (125, 140);
576               tft.setTextColor(letras);
577               tft.print(handicap);
578               tft.setTextSize(1);
579               tft.drawXBitmap(5, 184, LhandicapChico, 51, 53, letras);
580
581            }
582            if(pagel==1 && page2==0){
583
584
585              tft.setTextColor(letras);
586             tft.setFreeFont(CF_OL24);
587             tft.setCursor (5, 20);
588             tft.print("ELEGIR CANCHA");
589              tft.setFreeFont(FM12);
590               tft.setTextColor(letras);
591               tft.setCursor (20, 50);
592               tft.print("  PALIHUE");
593               tft.setCursor (20, 90);
594               tft.print("  PUERTO BELGRANO");
595               tft.setCursor (20, 130);
596               tft.print("  PAGO CHICO");
597               tft.setCursor (20, 170);
598               tft.print("  MONTE HERMOSO");
599               tft.setCursor (190, 220);
600               tft.print("  VOLVER");
601               tft.drawXBitmap(5, 184, LjugarChico, 51, 53, letras);
```

Figura 43: Detalle de la edición del Handicap. Fotografía de elaboración propia.

Parece algo sencillo e intrascendente, pero para elegir cómo seleccionar entre las opciones disponibles en pantalla, se pasó por varias ideas hasta llegar a la última versión, más vistosa y elegante. La misma fue diseñada de una forma circular que se rellena cuando se está seleccionando dicha opción.

```
case 0:


    //1ER  PINTADO
    tft.fillCircle(25, 43, 7, letras);
    //2DO SIN PINTAR
    tft.fillCircle(25, 83, 7, letras);
    tft.fillCircle(25, 83, 5, background);
    //3ERO SIN PINTAR
    tft.fillCircle(25, 123, 7, letras);
    tft.fillCircle(25, 123, 5, background);
    //4TO SIN PINTAR
    tft.fillCircle(25, 163, 7, letras);
    tft.fillCircle(25, 163, 5, background);
    //5TO SIN PINTAR
    tft.fillCircle(195, 213, 7, letras);
    tft.fillCircle(195, 213, 5, background);
    canchaElegida="Palihue";
     break;
```

Figura 44: Detalle de diseño del seleccionador de opciones. Fotografía de elaboración propia.

Cuando se programa el guardado y sobreescritura de los datos que se utilizaran en la SD, se realiza de la siguiente manera:

```
888 void sobrescribirHandicap(int nuevoValor) {
889   // Abrir el archivo en modo de lectura
890   myFile = SD.open("/handicapSD.txt");
891   String contenidoCompleto = "";
892
893   if (myFile) {
894     while (myFile.available()) {
895       contenidoCompleto += (char)myFile.read();
896     }
897     myFile.close();
898   } else {
899     Serial.println("Error al abrir el archivo.");
900     return;
901   }
902
903   // Buscar la posición del primer salto de línea
904   int posicionSaltoDeLinea = contenidoCompleto.indexOf('\n');
905
906   // Construir el nuevo contenido con la primera línea actualizada
907   String nuevoContenidoCompleto = String(nuevoValor) + contenidoCompleto.substring(posicionSaltoDeLinea);
908
909   // Abrir el archivo en modo de escritura (sobrescribir)
910   myFile = SD.open("/handicapSD.txt", FILE_WRITE);
911   if (myFile) {
912     // Escribir el nuevo contenido en el archivo
913     myFile.print(nuevoContenidoCompleto);
914     myFile.close();
915     Serial.println("Sobrescritura completa.");
916   } else {
917     Serial.println("Error al abrir el archivo para escritura.");
918   }
919 }
```

Figura 45: Función para la edición y sobreescritura del Handicap. Fotografía de elaboración propia.

```
void guardarPartida(int n, int total,  int handicap, String cancha, int par1) {
  File archivo1;
  archivo1 = SD.open("/partidas.txt", FILE_WRITE);
Serial.println("Entro");
  if (archivo1) {
    archivo1.seek(archivo1.size());
  archivo1.print("Cancha "+ String(cancha)+ ": Gross: " + String(total) + "  Neto:  " + String(total+handicap)+",  "+ "Par: " +String(par1)+ "  "+String(n+1)+ " hoyos.");
    archivo1.println();
    archivo1.close();
    Serial.println("Escritura Completa");
  } else {
    Serial.println("Error al abrir el archivo");
  }
}
```

Figura 46: Función para el guardado correcto de la partida. Fotografía de elaboración propia.

Teniendo el divisor resistivo en la entrada analógica de la ESP32, se puede de esta manera medir directamente el estado de la batería para poder mostrarlo en pantalla para que el usuario sepa en qué estado se encuentra su equipo.

```
void leerBateria(){

  bateria = analogRead(BAT_PIN);
  vout = (bateria * 3.3) / 4096.0;
  vin = vout /(R2/(R1+R2));
  Serial.print("Vout ");
  Serial.println(vout);
  Serial.print("Voltaje Bateria: ");
  Serial.println(vin);
  vin=25;
  if (menu!=0){
  if(vin>26.4) tft.fillRect(293, 10, 20, 9, TFT_GREEN);
  if (vin > 26 && vin < 26.4){ tft.fillRect(293, 10, 15, 9, TFT_YELLOW); tft.fillRect(309, 10, 5, 9, background);}
  if (vin > 25.5 && vin < 26){ tft.fillRect(293, 10, 10, 9, TFT_ORANGE); tft.fillRect(303, 10, 10, 9, background);}
  if(25.5>vin){ tft.fillRect(293, 10, 5, 9, TFT_RED); tft.fillRect(298, 10, 15, 9, background);}
  }
```

Figura 47: Función para el cálculo e impresión del estado de la batería. Fotografía de elaboración propia.

Cada una de las imágenes que componen el menú fueron cuidadosamente diseñadas individualmente en Photoshop. Posteriormente, se convirtieron en mapas de bits para garantizar una actualización de pantalla fluida y rápida en el dispositivo. Debido a que, durante las pruebas iniciales, se observaba que al utilizar imágenes en formato PNG, la impresión se realizaba de manera progresiva de izquierda a derecha, lo cual resultaba perceptible para el ojo humano y no quedaba bien estéticamente. Por esta razón, se decidió emplear mapas de bits, a pesar de implicar un mayor esfuerzo en su creación. Los resultados obtenidos fueron altamente satisfactorios, asegurando así una representación visual coherente y atractiva en la pantalla. Este enfoque meticuloso en el diseño y la adaptación de las imágenes contribuye significativamente a la calidad general de la interfaz y su experiencia de usuario.

```
#define logoWidth  51  // logo width
#define logoHeight 53  // logo height

// Image is stored in this array
PROGMEM const unsigned char LjugarChico[] = {
  0x00, 0x00, 0x00, 0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1F, 0xC0, 0x01,
  0x00, 0x00, 0x00, 0x80, 0x1F, 0x80, 0x07, 0x00, 0x00, 0x00, 0xC0, 0x3F,
  0x00, 0x1E, 0x00, 0x00, 0x00, 0xC0, 0x3F, 0x00, 0x38, 0x00, 0x00, 0x00,
  0xC0, 0x3F, 0x00, 0xE0, 0x00, 0x00, 0x00, 0x80, 0x3F, 0x00, 0xC0, 0x01,
  0x00, 0x00, 0x80, 0x1F, 0x18, 0x80, 0x03, 0x00, 0x00, 0x00, 0x04, 0x3F,
  0x00, 0x06, 0x00, 0x00, 0x00, 0xC0, 0x3F, 0x00, 0x0E, 0x00, 0x00, 0x60,
  0xF0, 0x3F, 0x00, 0x0C, 0x00, 0x00, 0x30, 0xFC, 0x1F, 0x00, 0x18, 0x00,
  0x00, 0x30, 0xFE, 0x07, 0x00, 0x18, 0x00, 0x00, 0x18, 0xFF, 0x03, 0x00,
  0x30, 0x00, 0x00, 0x08, 0xFF, 0x01, 0x00, 0x30, 0x00, 0x00, 0x0C, 0xFF,
  0x03, 0x20, 0x60, 0x00, 0x00, 0x04, 0xFF, 0x03, 0x60, 0x60, 0x00, 0x00,
  0x06, 0xFF, 0x03, 0x40, 0x60, 0x00, 0x00, 0x03, 0xFF, 0x07, 0xC0, 0x40,
  0x00, 0x00, 0x01, 0xFE, 0x0F, 0xC0, 0xC0, 0x00, 0x80, 0x01, 0xFE, 0x0F,
  0x80, 0xC1, 0x00, 0x80, 0x00, 0xFC, 0x1F, 0x80, 0xC1, 0x00, 0xC0, 0x00,
  0xFC, 0x1F, 0x80, 0xC1, 0x00, 0x60, 0x00, 0xF8, 0x1F, 0x00, 0xC1, 0x00,
  0x60, 0x00, 0xF0, 0x3F, 0x80, 0xC3, 0x00, 0x30, 0x00, 0xF0, 0x3F, 0x00,
  0xC1, 0x00, 0x38, 0x00, 0xE0, 0x3F, 0x80, 0xC3, 0x00, 0x78, 0x00, 0xE0,
  0x7F, 0x00, 0xC1, 0x00, 0xF8, 0x00, 0xE0, 0x7F, 0x00, 0xC3, 0x00, 0xF8,
  0x00, 0xE0, 0x7F, 0x80, 0x41, 0x00, 0xF0, 0x00, 0xE0, 0x7F, 0x80, 0x61,
  0x00, 0x60, 0x00, 0xE0, 0x7F, 0x80, 0x61, 0x00, 0x00, 0x00, 0xE0, 0x7F,
  0x80, 0x31, 0x00, 0x00, 0x00, 0xC0, 0x7F, 0xC0, 0x30, 0x00, 0x00, 0x00,
  0xE0, 0x7F, 0xC0, 0x30, 0x00, 0x00, 0x00, 0xE0, 0x7F, 0x60, 0x18, 0x00,
  0x00, 0x00, 0xE0, 0x7F, 0x60, 0x1C, 0x00, 0x00, 0x00, 0xE0, 0x7F, 0x00,
  0x0C, 0x00, 0x00, 0x00, 0xF0, 0x7F, 0x00, 0x06, 0x00, 0x00, 0x00, 0xF0,
  0x3D, 0x00, 0x02, 0x00, 0x00, 0x00, 0xF0, 0x3F, 0x00, 0x00, 0x00, 0x00,
  0x00, 0xF8, 0x3C, 0x00, 0x00, 0x00, 0x00, 0x00, 0xF8, 0x3C, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x7C, 0x3E, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3E, 0x3E,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x3F, 0x3E, 0x00, 0x00, 0x00, 0x00, 0x80,
  0x1F, 0x3E, 0x00, 0x00, 0x00, 0x00, 0x80, 0x1F, 0x3E, 0x00, 0x00, 0x00,
  0x00, 0x80, 0x07, 0x3E, 0x00, 0x00, 0x00, 0x00, 0x80, 0x07, 0x1E, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x1E, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x0C, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, };
```

Figura 48: Imagen codificada en mapa de BITS. Fotografía de elaboración propia.

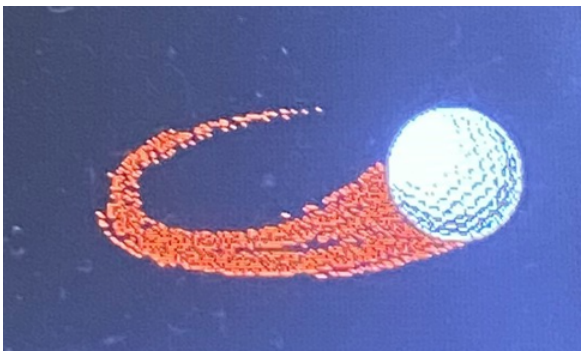

Figura 49: Logo plasmado con esta técnica. Fotografía de elaboración propia.

Control interno: Receptor

El módulo receptor se posiciona como el núcleo central del proyecto al desempeñar un papel integral. No se limita únicamente a recibir los datos de las coordenadas del usuario, sino que también asume la responsabilidad de analizarlos en detalle. Su función abarca el cálculo del rumbo, la determinación de la distancia, la detección de obstáculos y la gestión del arranque bidireccional, así como el frenado independiente de cada motor. En síntesis, este componente no solo recibe información, sino que desempeña un papel crítico al interpretar y ejecutar acciones precisas que son fundamentales para el funcionamiento global del proyecto.

Cuenta con entrada de 5V para alimentar el microcontrolador y todos los módulos, también posee las 2 salidas para el control bidireccional de cada motor, por lo tanto, tiene 4

salidas. Además, recibe los datos del sensor de obstáculos HC-SR04 los codifica y frena el carro de ser necesario.

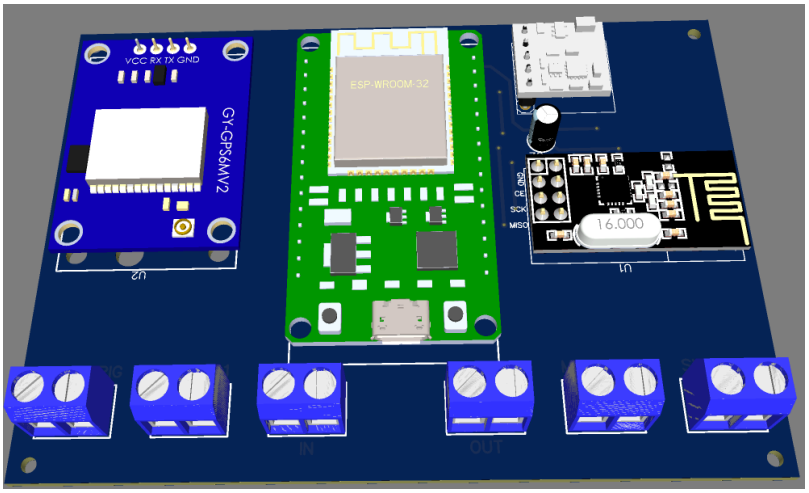

Figura 50: Plaqueta del receptor. Fotografía de elaboración propia.

Con este diseño se tuvo el inconveniente que el footprint de la brújula en el programa se encontraba a la inversa, y se tuvo que colocarlo de manera inversa. A fines prácticos queda funcional.

Figura 51: Placa real receptor. Fotografía de elaboración propia.

El código para el control del mismo posee muchas configuraciones y líneas, pero a continuación se mostraran las más importantes:

```
// Inicio Funciones Control de motores //
void carroFrenado1() {
  digitalWrite(pinMotorA1, LOW);
  digitalWrite(pinMotorA2, LOW);

  digitalWrite(pinMotorB1, LOW);
  digitalWrite(pinMotorB2, LOW);
}
void motorEnMarcha1() {
  digitalWrite(pinMotorA1, HIGH);
  digitalWrite(pinMotorA2, LOW);

  digitalWrite(pinMotorB1, HIGH);
  digitalWrite(pinMotorB2, LOW);
}
void izquierda1() {
  digitalWrite(pinMotorA1, HIGH);
  digitalWrite(pinMotorA2, LOW);

  digitalWrite(pinMotorB1, LOW);
  digitalWrite(pinMotorB2,HIGH);
}
void derecha1() {
  digitalWrite(pinMotorA1, LOW);
  digitalWrite(pinMotorA2, HIGH);

  digitalWrite(pinMotorB1, HIGH);
  digitalWrite(pinMotorB2, LOW);
}
// Fin Funciones Control de motores//
```

Figura 52: Código para control de motores. Fotografía de elaboración propia.

Otro problema que surge al realizar el código, fue determinar para qué lado era más conveniente girar para apuntar directamente al golfista. Debido a que el carro tiene una brújula y detecta el ángulo de desfasaje con respecto al norte, y con las coordenadas GPS se obtiene otro. En primera instancia parece muy sencillo de resolver, cuando el ángulo objetivo es menor al actual, que gire a la izquierda y cuando es mayor, que gire a la derecha. Sin embargo, es mucho más complejo, debido a que los ángulos con respecto al norte van desde los 0 grados hasta los 360 grados, en forma circular. Es decir que cuando este en, por ejemplo 5 grados y el objetivo en 340 grados, el carro debería girar a la izquierda ya que es el camino más corto. Todo esto fue resuelto gracias a la utilización de ángulos radianes y el cálculo con respecto al número PI para poder hallar a donde debería girar el carro. Está resumido en la siguiente función la cual recibe el ángulo actual de la brújula y el objetivo previamente calculado con las dos coordenadas GPS, la misma devuelve dos enteros, 1 para izquierda y 2 para derecha.

```
//Inicio Funcion Camino mas corto//
int direccion_giro(int azimuth_actual, int azimuth_objetivo) {
    // Convertir ángulos a radianes
    float azimuth_actual_rad = radians(azimuth_actual);
    float azimuth_objetivo_rad = radians(azimuth_objetivo);

    // Calcular la diferencia de ángulos
    float diferencia = azimuth_objetivo_rad - azimuth_actual_rad;

    // Asegurarse de que la diferencia esté en el rango [-pi, pi]
    if (diferencia > PI) {
        diferencia -= 2 * PI;
    } else if (diferencia < -PI) {
        diferencia += 2 * PI;
    }

    // Determinar la dirección del giro
    if (diferencia < 0) {
        return 1; // IZQUIERDA
    } else {
        return 2; // DERECHA
    }
```

Figura 53: Cálculo de camino más corto para giro. Fotografía de elaboración propia.

En el código siguiente se puede ver el control del carro, con cálculo de distancia, rumbo, control de parada y arranque. Utilizando también el resultado de la fórmula anterior, teniendo también una pequeña tolerancia para salvar las pequeñas variaciones que puede haber entre la brújula y el ángulo obtenido.

```
if(gpsGolfista.estado!=0){
if (gpsCarro.longitude != 0.0 && gpsGolfista.longitude != 0.0) {
  azimuthobjetivo = TinyGPSPlus::courseTo(gpsCarro.latitude, gpsCarro.longitude, gpsGolfista.latitude, gpsGolfista.longitude);
  brujula();
}

if (azimuthobjetivo != 0 && gpsGolfista.estado == 2) {
  brujula();
  Serial.print("Brujula:");
  Serial.println(a);
  azimuthINT = azimuthobjetivo;
  Serial.print("Azimuth a Golfista");
  Serial.println(azimuthINT);

  int direccion = direccion_giro(a, azimuthINT);
  Serial.print("Resultado Direccion");
  Serial.println(direccion);

  if (direccion == 1) {
    izquierdal();
    while (abs(a - azimuthINT) > tolerancia) {
      brujula();
      Serial.print("Azimuth a Golfista a izquierda:");
      Serial.println(azimuthINT);
      Serial.print("Brujula:");
      Serial.println(a);
    }
  }

  if (direccion == 2) {
    derechal();
    while (abs(a - azimuthINT) > tolerancia) {
      brujula();
      Serial.print("Azimuth a Golfista a derecha:");
      Serial.println(azimuthINT);
      Serial.print("Brujula:");
      Serial.println(a);
    }
  }
```

Figura 54: Control de giro y cálculo de azimuth. Fotografía de elaboración propia.

Control Remoto: Emisor

El emisor es el dispositivo que lleva el usuario, cuenta con dos botones, uno de “Follow” y otro de “Stop”, cuando se presiona cualquiera de ellos, se envía la información por NRF al carro para hacer las acciones correspondientes a cada botón.

Posee una batería de litio con un sistema de carga de 5v, para conectarlo a cualquier cargador universal de teléfono celular con ficha USB tipo C para su fácil y rápido cargado.

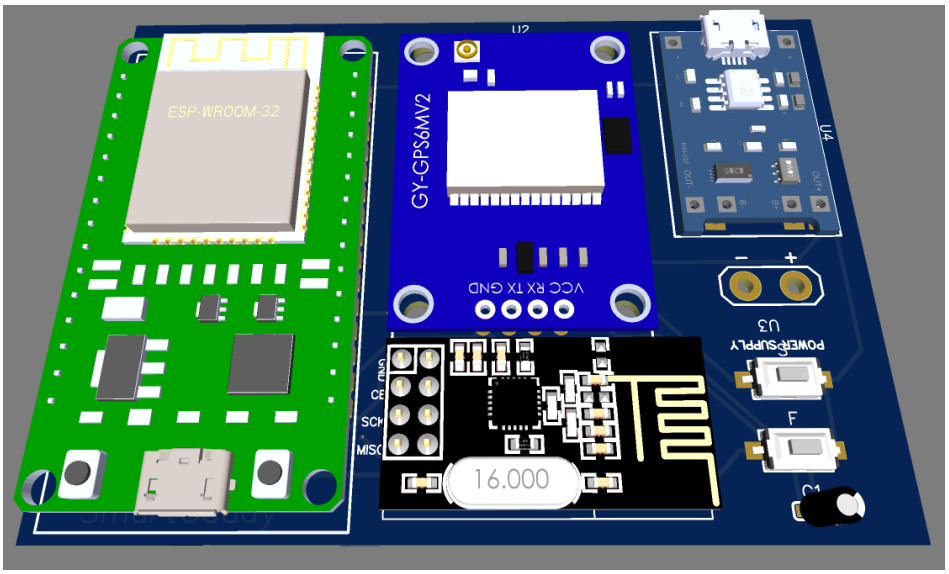

Figura 55: Plaqueta Emisor. Fotografía de elaboración propia.

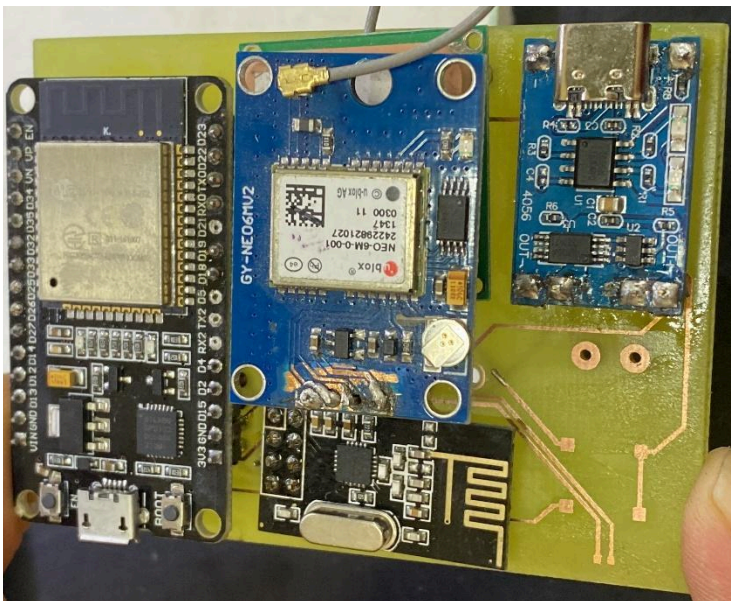

Figura 56: Plaqueta Real emisor. Fotografía de elaboración propia.

Se diseñó una carcasa especial para la comodidad del golfista y protección de todas las partes electrónicas del equipo, el mismo cuenta con la antena de transmisión de datos, con un botón de encendido y apagado. También cuenta con una batería y un sistema de carga con ficha USB tipo-C para una carga universal en cualquier cable que pueda tener el usuario. Y el resultado fue el siguiente:

Figura 57: Diseño Final para Emisor. Fotografía de elaboración propia.

El código para el control del mismo, es similar al del receptor, pero consta con unas pequeñas diferencias. Lo principal es que ambos poseen la misma configuración NRF para una correcta comunicación. Trabajamos con 250KBPS y con la potencia máxima del dispositivo para maximizar el alcance del rango de trabajo. Otro dato importante es trabajar lejos de la banda de 2,4GHz para evitar interferencias de dispositivos ya que comúnmente operan cerca de esas frecuencias.

```
Serial.begin(115200);
gpsSerial.begin(GPSBaud);
radio.begin();
radio.openWritingPipe(0xF0F0F0F0E1LL); // Dirección de la tubería de escritura
radio.setDataRate(RF24_250KBPS);
radio.setPALevel(RF24_PA_MAX);
radio.setChannel(0x76);
```

Figura 58: Configuración inicial GPS y NRF. Fotografía de elaboración propia.

La gran diferencia se centra en que el emisor solo envía los datos cuando uno de los dos botones del dispositivo se presiona. Estos están configurados como interrupciones con prioridad máxima dentro del código. En el instante en que se detecta su ejecución, se obtienen los datos GPS, se modifica la variable “estado” y se envían todos estos valores al receptor. Cabe destacar que la variable estado es la que determina si el carro debe avanzar o frenar. Estas interrupciones para optimizar su funcionamiento deben ser concisas.

```
28 void IRAM_ATTR follow1() {
29   estado = 2;
30 }
31
32 void IRAM_ATTR stop1() {
33   estado = 1;
34 }
35
```

Figura 59: Interrupciones. Fotografía de elaboración propia.

Se configura que cuando se cambia el estado de la variable principal, se envía la misma y con ella los datos leídos por el GPS, finalmente se reinicia el valor de la variable para esperar nueva actualización de los botones.

```
68 if (estado == 1 || estado == 2)  {
69
70         datosAEnviar.longitude = gpsData.longitude;
71         datosAEnviar.latitude = gpsData.latitude;
72         datosAEnviar.estado = estado;  // Cambiar el valor de acuerdo al estado actual
73         radio.write(&datosAEnviar, sizeof(datosAEnviar));
74         Serial.println("Datos enviados");
75         estado = 0;  // Restablecer el estado después de enviar los datos
76 }
77 }
```

Figura 60: Envio de datos al receptor. Fotografía de elaboración propia.

Control de giro bidireccional

Para control de los motores, primero se opta por utilizar un módulo de puente H, el cual viene con el chip L298 (ST, 2023), fueron realizadas todas las pruebas pertinentes con el primer prototipo del carro, el cual poseía unos motores muy pequeños por lo que el funcionamiento era optimo con el módulo L298. Al momento de trabajar con los motores del carro en tamaño real, surgió el problema de que su consumo era de 4A y el módulo soporta máximo 2A por salida.

Este problema se resolvió realizando un circuito basado en un puente H, donde por medio de dos salidas (por motor) de la ESP32 se controla la base de un transistor NPN que a su vez activa un relé de alta potencia. Controlar los motores de potencia mediante un circuito puente H con relés, utilizando un sistema de control de muy baja potencia, ofrece una serie de ventajas significativas. En primer lugar, este enfoque proporciona una eficiencia energética notable, ya que el sistema de baja potencia consume cantidades mínimas de energía, crucial para este proyecto donde la conservación de energía es prioritaria.

La capacidad de manejar altas corrientes es una característica esencial de los circuitos puente H, y la utilización de relés para este propósito ofrece robustez y fiabilidad en la operación del motor. Además, los relés actúan como interruptores electromecánicos, añadiendo una capa de protección al sistema de control de baja potencia, ofreciendo aislamiento eléctrico y resguardando contra posibles picos de corriente o sobretensiones. Los relés que se utilizaron fueron los siguientes:

Figura 61: Relé de 10A. Fotografía de elaboración propia.

La implementación resulta simple y económica, ya que los relés son componentes accesibles y fáciles de encontrar. Además, posee la versatilidad para ser controlado por una salida de la ESP32 la cual dispone de 3.3v y no más de un rango de los pocos mA.

En el circuito de activación del relé, se emplea un transistor NPN como interruptor controlado por un microcontrolador. La bobina del relé se alimenta con 5V, y cuando se aplica un voltaje de 3.3V a la base del transistor desde el microcontrolador, este se satura, permitiendo el paso de corriente desde el colector hasta el emisor.

Cuando el transistor se activa, se cierra el circuito entre el colector y el emisor, posibilitando que fluya la corriente de los 5V a través de la bobina del relé. La corriente en la bobina crea un campo magnético que activa el interruptor interno del relé, conectando así los contactos del relé.

La conexión del emisor del transistor a tierra y la activación de la base mediante el microcontrolador permiten el control efectivo de la corriente de la bobina del relé, activando o desactivando el relé según las necesidades del circuito. Este diseño ofrece la capacidad de controlar ambos motores de cargas grandes con una señal de baja potencia. Sumado a esto ambos relés se encuentran conectados entre sí de tal manera, que cuando se activa una salida de la ESP, el motor gira en un sentido, y cuando se activa la otra, el motor gira en sentido inverso. Para cada motor se realizó la misma configuración. A continuación, el circuito:

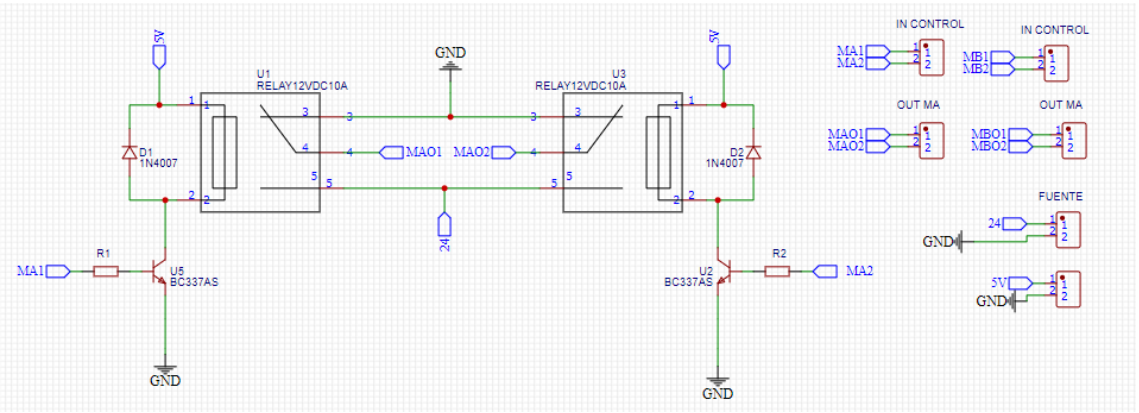

Figura 62: Conexionado Puente H con relés. Fotografía de elaboración propia.

Posteriormente se realizó el armado de la plaqueta y diseño en 3D para facilitar su ensamblaje final.

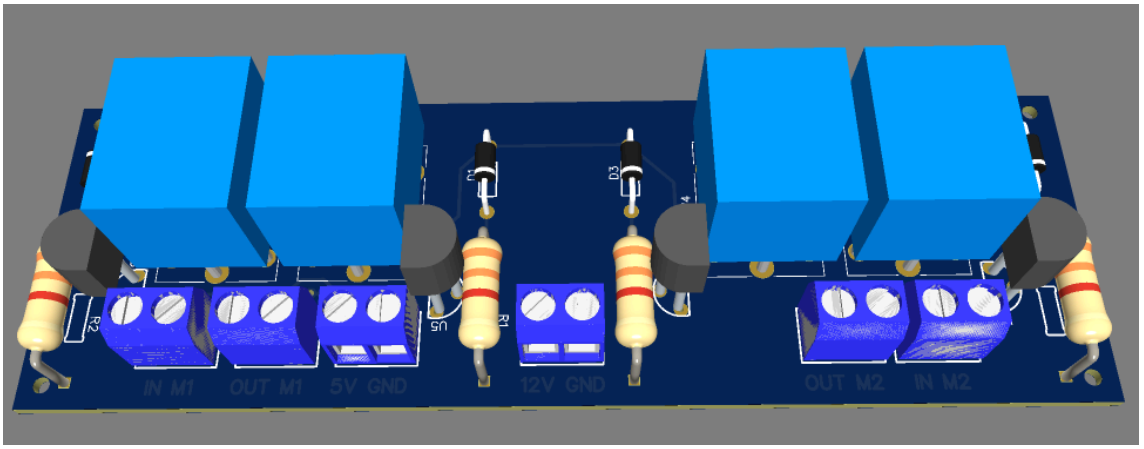

Figura 63: Placa de control de motores bidireccional. Fotografía de elaboración propia.

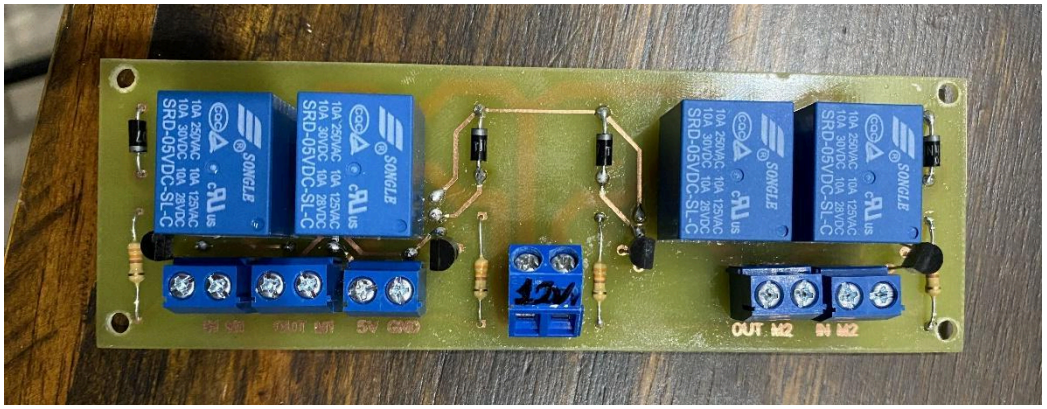

Figura 64: Placa real placa de control de motores. Fotografía de elaboración propia.

Luego de hacer todas las pruebas pertinentes, se tuvo el inconveniente de que un motor giraba a mayor velocidad que otro por lo que el carro tendía a girar hacia un lado. La solución por la que se optó, por sugerencia de los tutores de cátedra, fue la de diseñar un control PWM de los motores con un mosfet IRFZ44N (IR,2010).

Con el mismo se puede regular la velocidad del motor que giraba a mayor velocidad, el cual fue seteado a un valor de 89%, mientras que el otro se dejó en un 100% de su potencia, siendo así la combinación para una sincronización óptima.

Al contar con la posibilidad de regular las velocidades y no tener un sistema de ON/OFF como se tenía previamente con los relés, y ante la necesidad inevitable de realizar este sistema, se aprovechó esta situación para regular también el control de giro, haciendo que se realice un giro mucho más despacio y por ende, mucho más preciso. El mismo fue configurado a un giro del 30% de la rueda correspondiente al lado donde debe girar el caddy.

El mismo se controla de la misma forma que el anterior diseño, con una señal de control desde la ESP32 se activa la base de los transistores, que a su vez activan a los Mosfet de corriente, para permitir el arranque del motor. Modificando el ciclo de trabajo de la señal de control, se puede controlar la cantidad de energía que se envía a los motores.

A continuación el diseño de la plaqueta, fue diseñada en 3D, y luego testeada en plaqueta experimental, posteriormente se realizará en la plaqueta final.

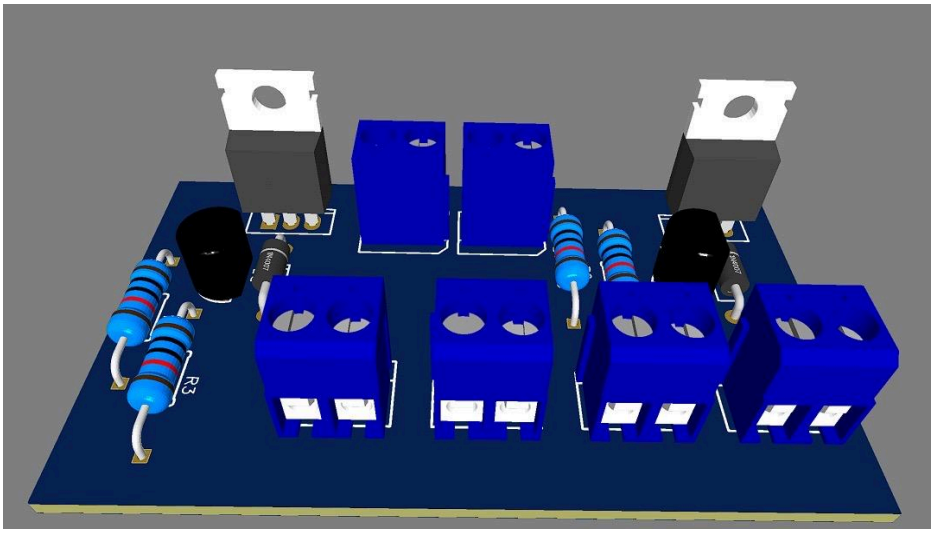

Figura 65: Diseño plaqueta en 3D control de motores por PWM. Fotografía de elaboración propia.

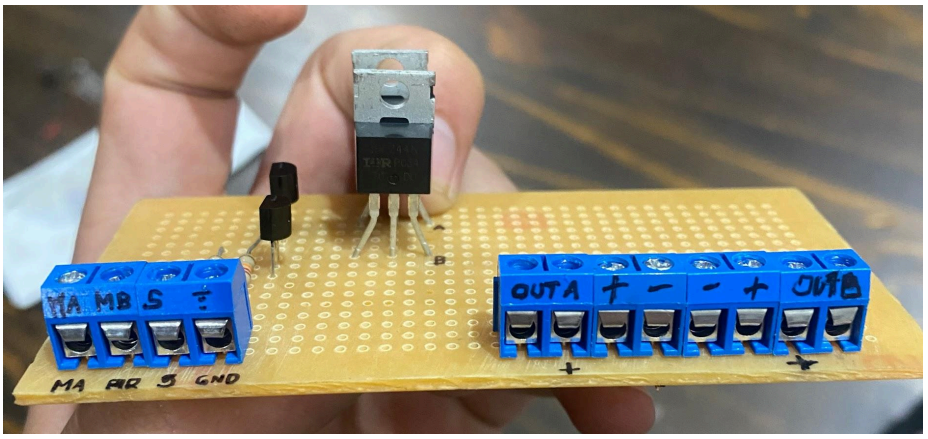

Figura 66: Plaqueta experimental de control de motores por PWM. Fotografía de elaboración propia.

Conclusión

Finalmente, como conclusión, se puede decir que este proyecto ha llevado al equipo a plasmar todos los conocimientos adquiridos a lo largo de la carrera. Cada parte del proyecto fue un desafío muy grande, ya que al tratarse de un producto sin precedentes, se tuvo que enfrentar a la falta de información existente. Sorteando varias dificultades que se fueron presentando en el camino para llegar a un buen resultado. Siendo conscientes de que este proyecto tiene un futuro y un potencial muy grande; por eso mismo, se tiene presente que hay muchas mejoras y cambios por hacer, teniendo como objetivo principal la adaptación a las necesidades del golfista.

Este proyecto no solo representa la finalización de los estudios académicos, sino también el inicio de una nueva etapa en cada una de las carreras profesionales. Sabiendo de que el producto que fue desarrollado tiene la capacidad de impactar positivamente en el mercado, por lo que se seguirá trabajando en su desarrollo.

Sin duda, este proyecto final de carrera ha sido un largo camino lleno de aprendizaje y desafíos. A lo largo de este viaje, nos hemos enfrentado obstáculos que nos obligaron a crecer tanto personal como profesionalmente. Sin embargo, gracias al esfuerzo, dedicación y apoyo de

todos los involucrados, hemos logrado convertir una idea en realidad. Queremos expresar nuestro sincero agradecimiento a todos aquellos que han contribuido de alguna manera a que este proyecto se haga realidad. Desde los profesores que nos guiaron con sus conocimientos y experiencias, hasta los compañeros de equipo, familiares y amigos que siempre estuvieron apoyando y acompañando en este camino.

Bibliografía y Referencias

DOIT (2022). ESP32 DevKit V1 datasheet. Recuperado de https://github.com/SmartArduino/SZDOITWiKi/wiki/ESP8266---ESP32.

Espressif Systems. (2023). ESP32-WROOM-32 datasheet. Recuperado de https://espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf.

Nordic Semiconductor (2007), NRF24L01 V2 datasheet. Recuperado de https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/523/nRF24L01_PS_July2007.pdf.

Ublox (2011) NEO-6M datasheet. Recuperado de https://www.u-blox.com/sites/default/files/products/documents/NEO-6_DataSheet_%28GPS.G6-HW-09005%29.pdf.

Honeywell (2010). HMC5883 datasheet. Recuperado de https://www.digikey.com/en/htmldatasheets/production/786600/0/0/1/asd2613-r.

Cytron technologies (2013). HCSR04 datasheet. Recuperado de https://docs.google.com/document/d/1Y-yZnNhMYy7rwhAgyL_pfa39RsB-x2qR4vP8saG73rE/edit.

ILITEK (2010). ILI9341 Recuperado de: https://cdn-shop.adafruit.com/datasheets/ILI9341.pdf.

JOY IT (2017) Encoder KY-040. Recuperado de: https://pdf1.alldatasheet.es/datasheet-pdf/download/1648739/JOY-IT/KY-040.html.

ST (2023) chip L298. Recuperado de: https://www.st.com/resource/en/datasheet/l298.pdf.

IR (2010) mosfet IRFZ44N. Recuperado de: https://www.infineon.com/dgdl/Infineon-IRFZ44N-DataSheet-v01_01-EN.pdf?fileId=5546d462533600a40153563b3a9f220d.