Prototipo de control virtual para simulador de conducción básico usando una IMU y un filtro de Kalman

Yesid Ricardo Rengifo Sanabria

Melissa Alvarez Anaya

Andres Felipe Gomez Agudelo

Trabajo de Grado para Optar al Título de Ingeniero Electrónico

Director

David Alberto Padilla Toloza, Ms. en Ingeniería Electrónica

Codirectores

Jaime Guillermo Barrero Perez, PhD. Ingeniería Electrónica

Universidad Industrial de Santander

Facultad de Ingenierías Fisicomecánicas

Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones

Bucaramanga

2023

Dedicatoria

A mis padres, Yesid Rengifo y Cenobia Offir Sanabria que desde pequeño me inculcaron el amor por el aprendizaje, quienes me enseñaron a querer conquistar el mundo, a mi familia por darme la fuerza necesaria para seguir luchando, este logro es todo nuestro.

A Melissa, mi gran compañera; a Andrés, un espectacular colega. A todos los compañeros de la universidad, gracias por ser parte de este viaje.

A mis compañeros de piso y amigos, gracias por estar siempre ahí. También a los personajes que hicieron cameos en la película de mi vida.

A los profesores que trascendieron el aula: Carlos, Gabriel, Manuel Jose, y otros.

A Vivian, amor de mi vida, mi eterno Bob Construye.

A todas mis ex y Angie Daniela, gracias por inspirar "La Bruja de mi Ex", les deseo éxito.

A Jhohanna Castellanos, quien me animó a estudiar esta carrera años atrás, éxitos para ti y tu familia.

-Yesid Ricardo Rengifo

Dedicado a mis queridos compañeros de trabajo, Yesid y Andrés, quienes compartieron conmigo el camino de este proyecto con apoyo incondicional y valiosa colaboración. A mis amigos que siempre estuvieron pendientes de mi progreso y siempre estuvieron ahí brindando apoyo académico y emocional. A mi amado Juan Diego, quien siempre me brindó ánimos,amor y apoyo inquebrantable en cada paso hacia el éxito y nunca dudó de mi. A mi familia, quiero expresar mi agradecimiento. A pesar de la distancia, su amor y apoyo me acompañaron en cada desafío. A mi padre, aunque no esté físicamente presente, sé que desde el cielo celebra con orgullo y amor cada paso de este camino.

Este proyecto está dedicado a cada uno de ustedes, mis amigos , familia y mi pareja, porque fueron mi inspiración para no rendirme. A pesar de los desafíos, juntos demostramos que es posible alcanzar nuestras metas cuando se camina de la mano y se comparten sueños. Gracias por creer en mí y formar parte de esta maravillosa travesía hacia el logro de mis sueños.

A todos ellos, gracias.

-Melissa Alvarez

No podría haber llegado hasta aquí sin mis padres, Nelvis Gomez y Cecilia Miranda. Siempre han estado a mi lado, apoyándome en cada paso de esta aventura universitaria. Su amor y su fuerza me han dado el valor para seguir adelante, incluso en los momentos más difíciles.

A mis compañeros de la universidad, gracias por estar ahí. Cada uno de ustedes han dejado una huella en mi vida que siempre llevaré conmigo.

Finalmente, a mi mejor amigo, Michael Cadena, le agradezco por su incondicional amistad. Gracias por estar siempre allí, por tus palabras de aliento y por ser un verdadero amigo.

-Andrés Gomez

.

Agradecimientos

Queremos expresar nuestros sinceros agradecimientos a todas las personas e instituciones que hicieron posible la realización de este proyecto.

En primer lugar, agradecemos de manera especial a nuestro director, David Padilla, por su invaluable orientación, apoyo y guía durante todo el desarrollo de este proyecto. Su experiencia y conocimientos fueron fundamentales para darle forma y dirección a nuestras ideas.

También queremos agradecer a la Universidad Industrial de Santander por brindarnos la oportunidad de estudiar y formarnos como profesionales. Gracias a esta institución, hemos adquirido los conocimientos y habilidades necesarios para llevar a cabo este proyecto con éxito.

A nuestro amigo Sebastián Campo, le agradecemos su incondicional apoyo y disposición para ayudarnos en los momentos en que más lo necesitábamos. Su presencia y aliento fueron un pilar fundamental durante todo el proceso.

Así mismo, extendemos nuestro agradecimiento a nuestros amigos del IEEE, quienes nos acompañaron en este camino y compartieron con nosotros el entusiasmo por la tecnología y la innovación.

A todas las personas que, de una u otra manera, contribuyeron con su apoyo, tiempo y ánimo en este proyecto, les damos nuestras más sinceras gracias. Sin su colaboración, este logro no habría sido posible.

Este proyecto representa un esfuerzo conjunto y un camino lleno de aprendizaje y superación. Estamos agradecidos por cada experiencia y desafío que encontramos en el trayecto, ya que cada uno de ellos nos ha permitido crecer como profesionales y como personas.

Nuestro sincero agradecimiento a todos. Sin duda, este proyecto no solo ha sido una muestra de nuestro trabajo en equipo, sino también una oportunidad para fortalecer lazos y compartir una pasión por la tecnología y la innovación.

A todas las personas que intervinieron de manera directa o indirecta en la realización de este trabajo, gracias totales.

Tabla de Contenido

Lista de Figuras

Lista de Tablas

Lista de Apéndices

pág.

Resumen

Título: Prototipo de control virtual para simulador de conducción básico usando una IMU y un filtro de Kalman [*]

Autores: Yesid Ricardo Rengifo Sanabria, Melissa Alvarez Anaya, Andres Felipe Gomez Agudelo [**]

Palabras Claves: ESP32, Filtro Kalman, IMU, Unreal Engine 4, UDP, JSON.

Descripción: El proyecto de grado consiste en el diseño e implementación de un prototipo de control virtual para un simulador de conducción básico, que utiliza una unidad de medición inercial (IMU) BNO05 junto a un filtro de Kalman para la obtener y procesar datos de movimiento. Los datos obtenidos serán transmitidos a través de la placa ESP32 por medio de una conexión WiFi a un ordenador, para que sean procesador por el motor gráfico Unreal Engine 5, que será utilizado para la creación de la interfaz de usuario y la simulación del entorno de conducción. El objetivo principal es desarrollar una solución accesible y de bajo costo para mejorar la experiencia de usuario en la simulación de conducción, a través de un control virtual que permita una mayor interacción y realismo en la simulación.

---

[*] Trabajo de grado

[**] Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones. Directora: David Alberto Padilla Toloza, Ms. en Ingeniería Electrónica. Codirectores: Jaime Guillermo Barrero Perez, PhD. Ingeniería Electrónica

Abstract

Title: Virtual control prototype for basic driving simulator using an IMU and a Kalman filter [*]

Authors: Yesid Ricardo Rengifo Sanabria, Melissa Alvarez Anaya, Andres Felipe Gomez Agudelo [**]

Keywords: ESP32, Kalman Filter, IMU, Unreal Engine 4, UDP, JSON.

Description: The degree project consists of the design and implementation of a virtual control prototype for a basic driving simulator using a BNO055 inertial measurement unit (IMU) and a Kalman filter to obtain and process motion data. The obtained data will be transmitted via an ESP32 board over the Internet to the Unreal Engine 5 graphics engine, which will be used for the creation of the user interface and the simulation of the driving environment. The primary aim of this ambitious endeavor is to present an accessible and cost-effective solution, ensuring that users can have an unparalleled driving simulation experience that fosters heightened interaction and an unparalleled sense of realism. By mitigating the barriers of conventional driving training, such as the scarcity of physical vehicles, high costs, and potential safety risks, this project offers a compelling alternative that promotes safer, more confident, and skilled drivers. Beyond its immediate applications, this project holds vast potential for future development and expansion. Its versatile nature opens doors for potential usage in other industries, such as aviation and heavy machinery operation, where precise training and operational expertise are of utmost importance. In essence, this virtual control prototype is a visionary undertaking that seeks to reshape the landscape of driving simulations and education. As it evolves, it promises to redefine the training approach, provide a secure environment for learners to refine their driving skills, and

---

[*] Bachelor Thesis

[**] Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones. Director: David Alberto Padilla Toloza, Ms. In Electronic Engineering. Advisors: Jaime Guillermo Barrero Perez, PhD. In Electronic Engineering

unlock a realm of possibilities for advanced driving education.

Introducción

En un mundo donde la variedad de vehículos y maquinaria pesada es diversa y esencial para diferentes industrias, surge la necesidad de capacitar a las personas en su manejo de forma segura y efectiva. Sin embargo, la adquisición y uso físico de estos vehículos para entrenamiento presenta desafíos, como riesgos de accidentes y costos elevados. Es en este contexto que nuestro proyecto propone una solución innovadora: un prototipo de conducción virtual básico.

Nuestro prototipo se basa en la integración de tecnologías avanzadas, como la IMU BNO055 y el filtro de Kalman, para adquirir y procesar datos precisos sobre la posición y movimiento del usuario. Estos datos se transmiten a través de una placa ESP32 al motor gráfico Unreal Engine 5, lo que permite crear una simulación de conducción inmersiva y realista. Mediante esta solución, las personas interesadas en aprender a manejar vehículos pueden hacerlo de manera segura y asequible, sin estar expuestas a riesgos de accidentes o daños materiales.

Además de su aplicación en el aprendizaje individual, nuestro prototipo tiene un potencial impacto a nivel industrial. Permitirá realizar entrenamientos masivos y simultáneos para la capacitación del personal en el manejo de maquinaria pesada y otros vehículos, mejorando la eficiencia y la seguridad en diversos sectores.

El problema que aborda este proyecto de grado es la falta de interactividad y realismo en los

simuladores de conducción básicos. Actualmente, los usuarios interactúan con los simuladores utilizando dispositivos de entrada estáticos, como un teclado o un controlador de videojuegos, lo que limita la sensación de inmersión y realismo en la simulación. Además, los sensores utilizados en los simuladores básicos a menudo no proporcionan una precisión suficiente en la medición de los movimientos del usuario.

Para solucionar estos problemas, se propone diseñar y desarrollar un prototipo de control virtual para un simulador de conducción básico, utilizando como sensor una IMU BNO055 y procesando sus mediciones con un filtro Kalman. La IMU BNO055 proporcionará una medición de los movimientos que realicé el usuario en el prototipo, mientras que el filtro de Kalman mejorará la precisión y estabilidad de estas mediciones. La transmisión de datos a través de la ESP32 permitirá una conexión por internet WiFi con el motor gráfico Unreal Engine 5, que se utilizará para la creación de la interfaz de usuario y la simulación del entorno de conducción, la cual podra funcionar desde un celular o un computador.

1. Marco teórico

1.0.1. Realidad Virtual - VR. La realidad virtual (VR) es una tecnología que permite a los usuarios sumergirse en un entorno simulado y interactuar con él mediante dispositivos como gafas o cascos de realidad virtual, controladores y sensores de movimiento. La VR utiliza la computación gráfica avanzada, el seguimiento de la posición y los movimientos del usuario y la retroalimentación háptica (sensaciones táctiles) para crear una experiencia inmersiva que puede simular la sensación de estar en un lugar diferente.

La realidad virtual se utiliza en una amplia gama de aplicaciones, desde videojuegos hasta educación, entrenamiento, terapia y más. En los videojuegos, la VR permite a los jugadores sentir que están dentro del juego, en lugar de simplemente controlar un personaje en una pantalla. En la educación, la VR puede utilizarse para simular experiencias que serían peligrosas o imposibles de llevar a cabo en la vida real, como explorar el espacio exterior o el interior del cuerpo humano(Josefa Katiuska Toala-Palma, 2020). En el entrenamiento, la VR puede utilizarse para simular situaciones de riesgo y enseñar habilidades críticas en un entorno controlado y seguro.

La realidad virtual también tiene el potencial de cambiar la forma en que interactuamos con el mundo. Por ejemplo, la VR puede utilizarse para crear experiencias inmersivas de turismo, permitiendo a los usuarios explorar lugares lejanos y culturas diferentes sin salir de casa. También puede utilizarse para crear mundos virtuales donde los usuarios pueden interactuar con otros en tiempo real, lo que puede tener implicaciones importantes para la comunicación y la colaboración a distancia.

1.0.2. Realidad Virtual en la industria y educación. La realidad virtual (VR) es una tecnología que puede ser muy útil en el campo de la simulación educativa y el entrenamiento del personal. Al permitir a los usuarios sumergirse en un entorno virtual interactivo, la VR puede simular situaciones de la vida real y proporcionar experiencias de aprendizaje y entrenamiento más inmersivas y efectivas.

En el ámbito educativo, la VR puede utilizarse para simular experiencias que de otra manera serían imposibles o peligrosas de realizar en la vida real. Por ejemplo, los estudiantes de medicina pueden utilizar la VR para practicar procedimientos médicos complejos(Lozé, 2020b) o para explorar el cuerpo humano en 3D(Lozé, 2020a). Los estudiantes de arquitectura e ingeniería también pueden utilizar la VR para diseñar y visualizar modelos en 3D de edificios y estructuras complejas(Pimentel, 2021).

En el ámbito del entrenamiento del personal, la VR puede utilizarse para simular situaciones de riesgo y enseñar habilidades críticas en un entorno controlado y seguro(Lozé, 2020a). Por ejemplo, los bomberos pueden utilizar la VR para entrenar en situaciones de incendios y rescate, y los trabajadores en entornos peligrosos como la construcción o la minería pueden utilizar la VR para practicar la seguridad en el trabajo.

1.1. Dispositivos

1.1.1. Sensor IMU BNO055 - Adafruit. Es un módulo de sensor que utiliza el BNO055 de Bosch Sensortec para proporcionar una solución completa de 9 grados de libertad (9-DoF) para

la detección de movimiento y la orientación. Al igual que el BNO055 de Bosch, el BNO055 de Adafruit combina un acelerómetro de 3 ejes, un giroscopio de 3 ejes y un magnetómetro de 3 ejes en un único paquete compacto. Además, el BNO055 de Adafruit incluye un regulador de voltaje y una interfaz de comunicación USB a TTL para facilitar la conexión con un ordenador o microcontrolador(Townsend, 2015).

Figura 1. Fotografía vista superior del Sensor BNO055 de Adafruit.

Fuente:HETPro. (s.f.). `https://hetpro-store.com/adaruit-9-dof-sensor-de-orientacion-absoluta-bno055/`

El BNO055 de Adafruit utiliza la misma tecnología avanzada de fusión de sensores que el BNO055 de Bosch para proporcionar una salida de orientación y velocidad angular de alta precisión en tiempo real. El módulo también incluye un sistema integrado de calibración automática que permite al sensor autocalibrarse y compensar automáticamente los errores del sensor causados por la temperatura y otros factores ambientales.

El BNO055 de Adafruit se comunica con los dispositivos host a través de una interfaz de comunicación serial estándar, como I2C o SPI, y se puede programar y configurar a través de un conjunto de comandos y registros de configuración. Adafruit también proporciona una biblioteca de software para ayudar en el desarrollo de aplicaciones que utilizan el BNO055. En resumen, el BNO055 de Adafruit es un módulo de sensor completo y fácil de usar que se utiliza en una amplia variedad de aplicaciones de detección de movimiento y orientación.

1.1.1.1. Magnetómetro. El magnetómetro incorporado en la IMU BNO055 tiene un papel fundamental en nuestro proyecto. Este sensor de tres grados de libertad mide la magnitud del campo magnético a lo largo de tres ejes coordenados, proporcionando información crucial sobre la orientación del dispositivo con respecto al campo magnético de la Tierra.

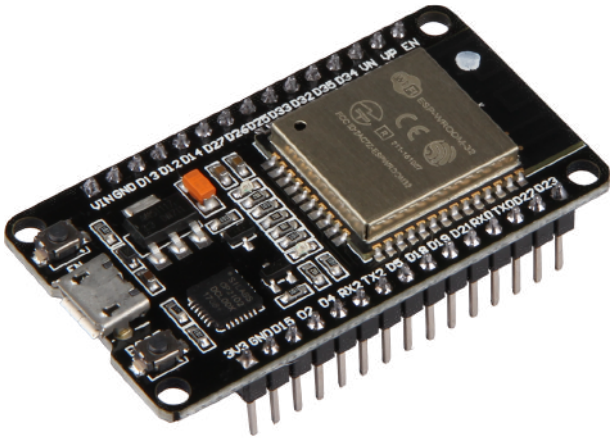
En nuestro proyecto, el dispositivo estará operando en condiciones donde puede mantenerse estable y cerca del suelo. Esto limita las rotaciones en los ejes de *roll* y *pitch* (rotación alrededor de los ejes horizontal y transversal, respectivamente), lo que hace que la estimación del ángulo de *yaw* (la rotación alrededor de un eje vertical) a partir de las mediciones del magnetómetro sea particularmente precisa. El cálculo se realiza utilizando la expresión:

$$\text{Yaw} = \text{atan2}\left(-\frac{m_y}{m_x}\right) \tag{1}$$

Donde m_y y m_x son las componentes del campo magnético medido en los ejes y y x, respectivamente.

1.1.2. ESP32. La ESP32 es un microcontrolador de bajo costo y bajo consumo de energía, diseñado y fabricado por la empresa china Espressif Systems.2 Se caracteriza por su alto rendimiento y su versatilidad, ya que cuenta con una amplia gama de periféricos y capacidades integradas, como conectividad WiFi y Bluetooth, procesador dual-core de 32 bits, memoria flash, interfaz SPI, I2C, UART, ADC, DAC, entre otros.

Figura 2. Fotografía vista isométrica del dispositivo Esp32.



Fuente: ESP32-DevKITC Board i Espressif.

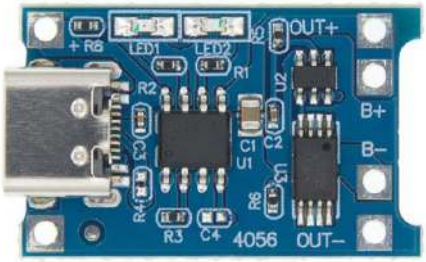
(s.f.).https://www.espressif.com/en/products/devkits/esp32-devkitc

El ESP32 es ampliamente utilizado en aplicaciones de IoT (Internet de las cosas), ya que permite conectar dispositivos y sensores a internet de manera sencilla y eficiente, y también se utiliza en proyectos de electrónica y robótica gracias a sus características avanzadas y su bajo

costo. Además, cuenta con una gran comunidad de desarrolladores que contribuyen con bibliotecas y recursos para facilitar el desarrollo de proyectos con este microcontrolador.

1.1.3. CI TP4056. Modulo de carga y descarga comercial el cual ofrece una entrada USB Tipo B. Este modulo ofrece una corriente de carga de 1A que se corta cuando haya terminado cuando el voltaje de la batería cae por debajo de 2,4 V, el chip de protección desconecta la carga para proteger la celda de funcionar a una tensión demasiado baja y también protege contra la conexión de sobre tensión y polaridad inversa. Los diferentes estados de carga y Standby los muestra al usuario por medio de LEDs indicadores. Se utiliza este CI para brindar una forma de interacción familiar con el usuario, por su entrada USB para facilitar su carga y los LEDs que indican el estado de la batería visibles para el usuario. Une vez cargada la batería,este modulo entregar a la tensión de la batería de manera segura pues, cuenta con algunos capacitores integrados de 10uF en paralelo a la batería y a la tensión de entrada que regula la corriente y evita sobrecargas o cambios abruptos en la misma,como se observa en la figura.

Figura 3. Modulo TP4056



Fuente: Electronilab(s.f)https://electronilab.co/tienda/modulo-cargador-bateria-de-lipo-1a-micro-usb-5v

1.1.4. LM2596. Este módulo esta basado en el Regulador DC-DC Step Down LM2596 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 3A. 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 LM2596 es una fuente de alimentación conmutada, así que su eficiencia es significativamente mayor en comparación con los populares reguladores lineales de tres terminales, especialmente con tensiones de entrada superiores.

Figura 4. Modulo LM2596

Fuente:Modulo LM2596 Regulador de voltaje DC-DC BuCK 1.25V-35V. (s.f.)https://www.ardobot.co/modulo-lm2596-regulador-de-voltaje-dc-dc-buck-1-25v-35v.html

1.1.5. Bateria 18650. El 18650 es una batería recargable de iones de litio. Un 18650 es una batería recargable construida con iones de litio. Su nombre propio es çelda 18650". La celda 18650 tiene un voltaje de 3,7 V Estos existen en dos formas; protegido y desprotegido. Recomendamos encarecidamente las baterías 18650 protegidas. El tiempo total de carga de la batería 18650 es de aproximadamente cuatro horas. El tiempo de carga puede variar con el amperaje y voltaje del adaptador, así como con el tipo de batería. En general, las celdas 18650 tienen una potencia nominal de 3,6, 3,7 y, a veces, 3,65 voltios. Todas esas calificaciones son en su mayoría iguales. Es el voltaje promedio general durante una descarga completa. Para la mayoría de los 18650, el rango de voltaje total está entre 2.5 voltios y 4.2 voltios.(Large, 2020)

El significado de 18650 corresponde a sus medidas. Los dos primeros números (18) indican su diámetro, 18 mm, mientras que el número 650 nos dice que su longitud es 65 mm. Sin embargo,

65 mm es la longitud neta, pero la longitud total de batería. Las ventajas principales de estas baterías son el nivel bajo de autodescarga y ausencia de efecto de memoria. Las pilas 18650 son relativamente livianas, no exigen mucho mantenimiento y tienen larga vida útil – de 500 a 1000 ciclos.(de Ingeniería, s.f.)

Figura 5. Bateria 18650

Fuente:Productos. (s. f.). Ferretrónica.

https://ferretronica.com/collections/all/baterias-18650

1.1.6. Pulsador. Los pulsadores son interruptores o botones que permiten o interrumpen el paso de la corriente eléctrica de manera momentánea 16 . Se utilizan ampliamente para controlar diversos dispositivos electrónicos y mecánicos, como encender luces, abrir puertas, ajustar dispositivos electrónicos, entre otras aplicaciones industriales y comerciales. La función principal de los pulsadores es desviar o interrumpir el flujo de corriente eléctrica en un circuito en el momento en

que se pulsan.(del Valle Hernández, 2022)

Figura 6. Pulsador

Fuente:https://www.areatecnologia.com. (s. f.). Pulsador.
https://www.areatecnologia.com/electricidad/pulsador.html

1.2. Código y protocolos

1.2.1. I2C. I2C (Inter-Integrated Circuit) es un protocolo de comunicación de bus serie utilizado para conectar dispositivos electrónicos de baja velocidad dentro de un sistema.

El protocolo I2C utiliza dos líneas de comunicación, una para la transmisión de datos (SDA) y otra para la sincronización de los datos (SCL). Los dispositivos conectados al bus I2C tienen direcciones únicas de 7 bits, lo que permite la comunicación con múltiples dispositivos utilizando las mismas líneas de bus.

El protocolo I2C se utiliza comúnmente en una amplia gama de aplicaciones, desde dispositivos de baja potencia, como sensores y controladores de motor, hasta dispositivos de alta velocidad, como memorias EEPROM y pantallas LCD. La simplicidad del protocolo I2C, su bajo consumo de energía y la capacidad de comunicación con múltiples dispositivos lo convierten en una opción

popular para muchas aplicaciones de sistemas embebidos.

1.2.1.1. I2C en Arduino. I2C (Inter-Integrated Circuit) es un protocolo de comunicación serie de dos hilos que se utiliza comúnmente para conectar dispositivos electrónicos en una placa de circuito impreso, incluyendo los dispositivos Arduino. El protocolo I2C utiliza dos líneas para la comunicación: una línea de datos (SDA) y una línea de reloj (SCL).

En Arduino, se puede utilizar la biblioteca Wire para implementar la comunicación I2C. Para comunicarse con un dispositivo I2C, primero es necesario conocer su dirección I2C. Cada dispositivo I2C tiene una dirección única de 7 bits que se utiliza para direccionarlo en la comunicación.

Para enviar datos a un dispositivo I2C en Arduino, se utiliza la función **Wire.beginTransmission(address)** para comenzar una transmisión hacia la dirección del dispositivo especificado por **address**. Luego, se envían los datos utilizando la función **Wire.write(data)**. Después de enviar todos los datos, se utiliza la función **Wire.endTransmission()** para finalizar la transmisión.

Para recibir datos de un dispositivo I2C, primero se utiliza la función **Wire.requestFrom(address, quantity)** para solicitar una cantidad específica de datos del dispositivo con la dirección especificada por **address**. Luego, se utilizan las funciones **Wire.available()** y **Wire.read()** para leer los datos recibidos.

Aquí hay un ejemplo básico de cómo enviar y recibir datos mediante I2C en Arduino:

```
1 #include <Wire.h>
```

```
2
3  int address = 0x2A;  // Direcci n I2C del dispositivo
4
5  void setup() {
6    Wire.begin();       // Inicializar la comunicaci n I2C
7    Serial.begin(9600); // Inicializar la comunicaci n serial para imprimir
       resultados
8  }
9
10 void loop() {
11   Wire.beginTransmission(address);  // Comenzar la transmisi n hacia el
       dispositivo
12   Wire.write(0x01);                 // Enviar el primer byte de datos
13   Wire.write(0x02);                 // Enviar el segundo byte de datos
14   Wire.endTransmission();           // Finalizar la transmisi n
15
16   Wire.requestFrom(address, 2);     // Solicitar dos bytes de datos del
       dispositivo
17   while (Wire.available()) {        // Leer los datos recibidos
18     Serial.print(Wire.read(), HEX); // Imprimir los datos recibidos en
       hexadecimal
19     Serial.print(" ");
20   }
21   Serial.println();
22
```

```
23   delay(1000);                          // Esperar un segundo antes de enviar otra
       vez
24 }
```

En este ejemplo, se envían dos bytes de datos (0x01 y 0x02) al dispositivo con dirección I2C 0x2A y se solicitan dos bytes de datos de vuelta. Luego, se imprimen los datos recibidos en la comunicación serial en formato hexadecimal. La función **delay(1000)** se utiliza para esperar un segundo antes de enviar otra vez.

1.2.2. JSON. JSON (JavaScript Object Notation) es un formato ligero y flexible de intercambio de datos que se utiliza para transmitir datos entre aplicaciones y servicios web. Es fácilmente legible por humanos y también es fácil de procesar por las máquinas. Se basa en un subconjunto del lenguaje de programación JavaScript y es compatible con muchos lenguajes de programación, incluyendo Python, PHP, Java, C#, entre otros.

El formato JSON consiste en una estructura de datos que se compone de pares de çlave:valor", separados por comas y encerrados entre corchetes o llaves. Las claves son siempre cadenas de texto entre comillas dobles ", y los valores pueden ser cadenas de texto, números, booleanos, listas, objetos o nulos.

1.2.2.1. Empaque de datos JSON. JSON se construye utilizando dos estructuras de datos universales: una colección de pares de nombre / valor y una lista ordenada de valores.

En muchos lenguajes de programación, estas estructuras se implementan como objetos, registros, estructuras, diccionarios, tablas hash, listas con claves o matrices asociativas. La colección de pares de nombre / valor permite nombrar y organizar los datos, mientras que la lista ordenada de valores permite mantener una secuencia específica de datos.

Debido a que estas estructuras de datos son comunes y se implementan en prácticamente todos los lenguajes de programación modernos, tiene sentido que JSON se base en estas estructuras. Esto significa que JSON es fácilmente intercambiable entre diferentes sistemas, lo que lo hace muy útil para la transmisión de datos en la web.

1.2.2.2. JSON en Arduino. En Arduino, es posible trabajar con JSON utilizando bibliotecas externas, como ArduinoJson.

La biblioteca ArduinoJson permite la creación y análisis de objetos JSON en Arduino. Para utilizar esta biblioteca, se debe descargar e instalar desde el administrador de bibliotecas de Arduino.

Para crear un objeto JSON en Arduino utilizando la biblioteca ArduinoJson, primero es necesario incluir la biblioteca en el programa utilizando la directiva include <ArduinoJson.h>. Luego, se debe crear un objeto de tipo DynamicJsonDocument que representará el objeto JSON.

Por ejemplo, el siguiente código crea un objeto JSON con dos claves "*sensor*" y "*valor*", y los valores "*temperatura*" y "25" respectivamente:

```
1 #include <ArduinoJson.h>
```

En este ejemplo, se crea un objeto **DynamicJsonDocument** con capacidad para 128 bytes.

Se agregan dos claves al objeto (”*sensor*” y ”*valor*”) con los valores correspondientes. Luego, se utiliza la función **serializeJson()** para convertir el objeto a una cadena JSON y se envía por la comunicación serial mediante la función **Serial.println()**.

Para analizar una cadena JSON en Arduino utilizando la biblioteca **ArduinoJson**, se utiliza la función **deserializeJson()** para convertir la cadena en un objeto **DynamicJsonDocument**.

Por ejemplo, el siguiente código analiza una cadena JSON y extrae los valores de las claves ”*sensor*” y ”*valor*”:

```
#include <ArduinoJson.h>
String json; //Json objeto

void outputProcesamiento()
{
  json = GetJSONString ("hand", pitch, roll, yaw, A, B, C);
  Serial.println(json);
  proceso = 2;
}

String GetJSONString(String sensorName, float pitch, float roll, float yaw,
    int A, int B, int C)
{
    DynamicJsonDocument doc(1024);
    doc["name"] = sensorName;

    DynamicJsonDocument orientation(768);
```

```
17      orientation["pitch"] = pitch;
18      orientation["roll"] = roll;
19      orientation["yaw"] = yaw;
20      orientation["A"] = A;
21      orientation["B"] = B;
22      orientation["C"] = C;
23      doc["orientation"] = orientation;
24      char docBuf[1024];
25      serializeJson(doc, docBuf);
26      return String(docBuf);
27  }
```

Este código en Arduino utiliza la biblioteca ArduinoJson.h para crear y serializar un objeto JSON. En la parte superior del código se declara una variable de cadena llamada json que se utilizará para almacenar el objeto JSON.

La función **GetJSONString** toma varios parámetros (sensorName, pitch, roll, yaw, A, B, C) y utiliza la biblioteca ArduinoJson para crear un objeto JSON con estos valores. La función comienza inicializando un documento JSON dinámico con capacidad para 1024 bytes. A continuación, se crean pares de nombre y valor para el objeto JSON. El par de nombre y valor "nameçontiene el valor del sensorName que se pasa como parámetro a la función. Luego se crea un objeto JSON para la orientación, que contiene los valores de pitch, roll, yaw, A, B y C. Luego, la función serializa el documento JSON en una cadena utilizando **serializeJson** y devuelve esta cadena. En la función **outputProcesamiento**, se llama a la función **GetJSONString** y se le pasan

los valores de pitch, roll, yaw, A, B y C junto con un valor de sensorName "hand". La cadena JSON resultante se almacena en la variable global "json".

Finalmente, se imprime la cadena JSON en el puerto serie utilizando **Serial.println**. En resumen, este código utiliza la biblioteca ArduinoJson para crear un objeto JSON a partir de valores de entrada y luego imprime la cadena JSON resultante en el puerto serie.

En este ejemplo, se define una cadena JSON utilizando comillas escapadas (") y se analiza utilizando la función **deserializeJson()**. Luego, se extraen los valores de las claves "*sensor*" y "*valor*" y se imprimen por la comunicación serial mediante la función **Serial.print()**.

1.2.3. UDP. UDP (User Datagram Protocol) es un protocolo de red que se utiliza para la transmisión de datagramas sin conexión en redes de computadoras. UDP se considera un protocolo no confiable porque no proporciona garantía de entrega, orden o duplicación de paquetes, lo que significa que los paquetes pueden llegar fuera de orden o pueden no llegar en absoluto. UDP se utiliza comúnmente para aplicaciones de transmisión de datos en tiempo real, como video y audio en vivo, videojuegos en línea, aplicaciones de chat y otras aplicaciones en las que la velocidad y la latencia son más importantes que la integridad de los datos.

A diferencia del TCP, que establece una conexión antes de transferir datos, UDP no establece una conexión antes de enviar datos. En su lugar, los datagramas se envían directamente a la dirección IP de destino especificada en el encabezado del datagrama. Debido a la falta de establecimiento de conexión y la falta de confirmación de recepción de datos, UDP es un protocolo más rápido y eficiente en términos de recursos de red que TCP.

1.2.3.1. UDP en Arduino. Este es un protocolo de comunicación de red que permite la transferencia de datos en paquetes sin conexión, lo que significa que no requiere una conexión establecida entre el emisor y el receptor. En Arduino, se puede utilizar el protocolo UDP para comunicarse con otros dispositivos en una red local.

A continuación, se describen los pasos para utilizar UDP en Arduino:

- **Incluir la librería Ethernet.h:** Para utilizar UDP en Arduino, se debe incluir la librería Ethernet.h. Esta librería proporciona una serie de clases y funciones que permiten la comunicación en red utilizando el protocolo TCP/IP.

- **Crear un objeto EthernetUDP:** Una vez incluida la librería Ethernet.h, se debe crear un objeto EthernetUDP para utilizar el protocolo UDP. Por ejemplo:

```
1    #include <Ethernet.h>
2    #include <EthernetUdp.h>
3
4    EthernetUDP Udp;
5
```

- **Configurar la conexión UDP:** Para configurar la conexión UDP, se deben establecer el puerto local y la dirección IP del receptor. Por ejemplo:

```
1    unsigned int localPort = 8888; // puerto local de escucha
```

```
2     IPAddress remoteIP(192, 168, 0, 177); // direcci n IP del receptor
3
4     Udp.begin(localPort); // iniciar conexi n
5
```

- **Enviar y recibir datos:** Una vez establecida la conexión UDP, se pueden enviar y recibir datos utilizando las funciones de la clase EthernetUDP. Por ejemplo:

```
1     // enviar datos
2     char mensaje[] = "Hola desde Arduino";
3     Udp.beginPacket(remoteIP, 8888);
4     Udp.write(mensaje);
5     Udp.endPacket();
6
7     // recibir datos
8     char buffer[UDP_TX_PACKET_MAX_SIZE];
9     int tamano = Udp.parsePacket();
10    if (tamano) {
11      Udp.read(buffer, tamano);
12      buffer[tamano] = '\0';
13      Serial.println(buffer);
14    }
15
```

En resumen, se puede utilizar la librería **Ethernet.h** en Arduino para configurar una conexión

UDP y enviar y recibir datos en una red local. Esto abre una amplia variedad de posibilidades para la creación de proyectos de IoT (Internet de las cosas), como el control de dispositivos a través de una red local o la comunicación entre varios dispositivos conectados en una red.

1.2.4. Motor Gráfico Unreal Engine 5. Unreal Engine es un motor de juego desarrollado y mantenido por la compañía Epic Games. Es un software de creación de juegos utilizado por desarrolladores de todo el mundo para crear videojuegos de diferentes géneros y plataformas, incluyendo juegos de consola, móviles, PC y realidad virtual.

El motor de juego ofrece una amplia gama de herramientas y características para crear juegos, incluyendo un sistema de scripting basado en Blueprint y C++, un editor de escena 3D, herramientas de animación y efectos visuales, un sistema de física, soporte para redes y mucho más. Además, Unreal Engine se caracteriza por su capacidad de generar gráficos avanzados, con características de alta calidad, como iluminación dinámica, sombras en tiempo real, reflejos y texturas de alta resolución.

Unreal Engine es muy popular entre los desarrolladores de juegos debido a su versatilidad y escalabilidad, y ha sido utilizado en el desarrollo de algunos de los juegos más populares y exitosos de la industria del videojuego, como Fortnite, Gears of War, BioShock, Borderlands, entre otros.

1.3. Estimador de perturbaciones

Para aprovechar la acción feed-forward que se puede implementar fácilmente en el algoritmo de control predictivo, se propuso el diseño de un estimador de perturbaciones basado en

clasificación de señales con el objetivo de hacer que sea independiente de un modelo.

1.3.1. Quaterniones. Un cuaternión es un tipo de número hipercomplejo que extiende la idea de un número complejo a cuatro dimensiones, se representa por cuatro números reales: un escalar (parte real) y un vector (parte imaginaria) con tres componentes. Es decir, un cuaternión puede escribirse como $q = w + xi + yj + zk$, donde w, x, y y z son números reales y i, j y k son los elementos imaginarios de los cuaterniones, que satisfacen las siguientes relaciones:

$$i^2 = j^2 = k^2 = ijk = -1 \tag{2}$$

Sin embargo, para muchas aplicaciones, es más conveniente representar las rotaciones utilizando ángulos de Euler.

Los ángulos de Euler son tres ángulos que describen la rotación de un objeto en el espacio tridimensional. Estos ángulos se llaman "ángulo de cabeceo"(rotación alrededor del eje X), "ángulo de inclinación"(rotación alrededor del eje Y) y "ángulo de guiñada"(rotación alrededor del eje Z).

Es posible convertir un cuaternión en ángulos de Euler utilizando fórmulas matemáticas específicas.

1.3.1.1. Normalización de un Quaternion. La normalización de un cuaternión es un proceso matemático que consiste en convertir un cuaternión en otro cuaternión que tiene la misma dirección (es decir, la misma rotación) pero una longitud (módulo) de uno. En otras palabras,

un cuaternión normalizado se encuentra en la superficie de una esfera de radio 1 y representa una rotación pura en el espacio tridimensional.

La normalización de un cuaternión es importante porque garantiza que el cuaternión no tenga componentes de escala o translación, que podrían afectar a la representación de la rotación en el espacio tridimensional. Además, los cuaterniones normalizados se utilizan comúnmente en cálculos matemáticos que implican la interpolación o la combinación de rotaciones.

La normalización de un cuaternión se puede realizar dividiendo cada componente del cuaternión por su longitud (módulo):

$$q_{norm} = \frac{q}{|q|} \tag{3}$$

donde 'q' es el cuaternión original y '$|q|$' es su longitud. La longitud de un cuaternión se calcula como:

$$|q| = \sqrt{q_w^2 + q_x^2 + q_y^2 + q_z^2} \tag{4}$$

Después de normalizar un cuaternión, su longitud será siempre igual a 1:

$$|q| = \sqrt{(\frac{q_w}{|q|})^2 + (\frac{q_x}{|q|})^2 + (\frac{q_y}{|q|})^2 + (\frac{q_z}{|q|})^2} = 1 \tag{5}$$

Es importante tener en cuenta que la normalización de un cuaternión solo cambia su longitud y no su dirección (es decir, la rotación que representa). En otras palabras, si dos cuaterniones representan la misma rotación en el espacio tridimensional, entonces sus versiones normalizadas también representarán la misma rotación.

1.3.1.2. Ángulo de cabeceo (rotación alrededor del eje X). Primero, se calcula el valor de seno y coseno del ángulo de cabeceo:

$$sen(roll) = 2(q_w \cdot q_x + q_y \cdot q_z) \tag{6}$$

$$cos(roll) = 1 - 2(q_x^2 + q_y^2) \tag{7}$$

Luego, se utiliza la función arcotangente de dos argumentos (ATAN2) para calcular el ángulo de cabeceo:

$$roll = atan2(sin(roll), cos(roll)) \tag{8}$$

1.3.1.3. Ángulo de inclinación (rotación alrededor del eje Y). Primero, se calcula el valor de seno y coseno del ángulo de cabeceo:

$$sen(pitch) = 2(q_w \cdot q_y + q_z \cdot q_x) \tag{9}$$

$$cos(pitch) = 1 - 2(q_y^2 + q_z^2) \tag{10}$$

Luego, se utiliza la función arcotangente de dos argumentos (ATAN2) para calcular el ángulo de inclinación:

$$pitch = atan2(sin(pitch), cos(pitch)) \tag{11}$$

1.3.1.4. Ángulo de guiñada (rotación alrededor del eje Z). Primero, se calcula el valor de seno y coseno del ángulo de guiñada:

$$sen(yaw) = 2(q_w \cdot q_z + q_x \cdot q_y) \tag{12}$$

$$cos(yaw) = 1 - 2(q_y^2 + q_z^2) \tag{13}$$

Luego, se utiliza la función arcotangente de dos argumentos (ATAN2) para calcular el ángulo de inclinación:

$$yaw = atan2(sin(yaw), cos(yaw)) \tag{14}$$

El resultado de estas fórmulas son los ángulos de Euler, también conocidos como "ángulo de cabeceo"(roll), "ángulo de inclinación"(pitch) y "ángulo de guiñada"(yaw), que describen la rotación representada por el cuaternión. Es importante tener en cuenta que estos ángulos se expresan generalmente en radianes y que el orden en que se aplican las rotaciones (por ejemplo, primero el cabeceo, luego la inclinación y luego la guiñada) puede variar dependiendo de la convención que se esté utilizando.

1.3.1.5. Grados de los ángulos. El proceso de obtención de cada ángulo usando el método de Euler, nos permite obtener los grados en radianes, ya el procesos final para obtener cada

ángulo en grados es:

$$yaw = euler.x() * \frac{180}{\pi}; \tag{15}$$

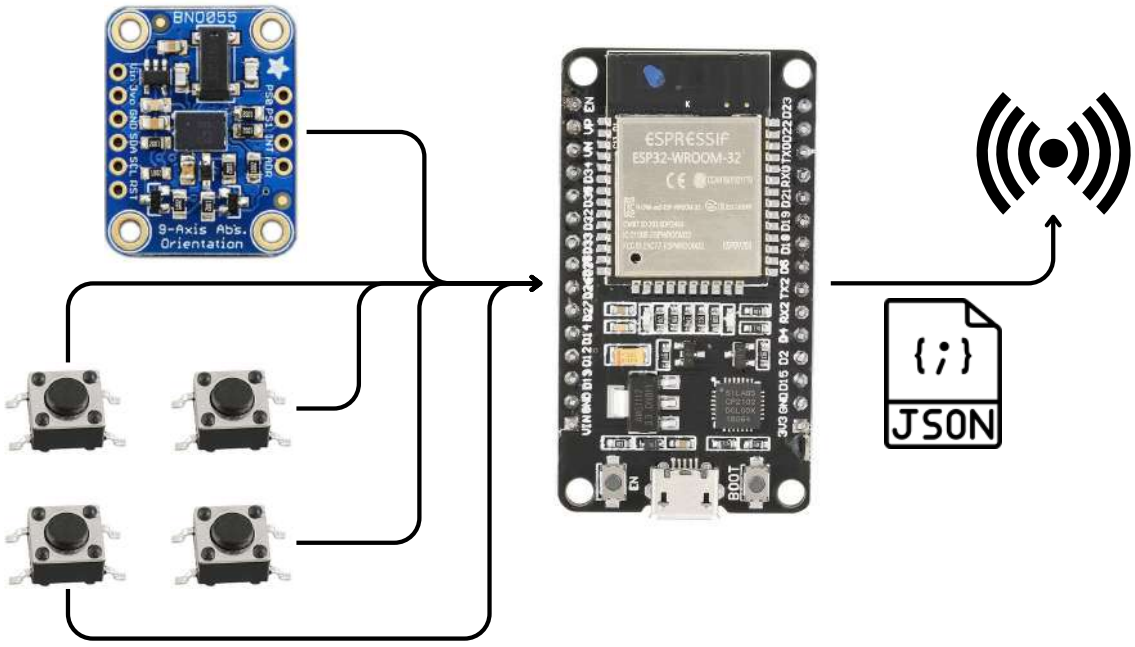
$$roll = euler.y() * \frac{180}{\pi}; \tag{16}$$

$$pitch = euler.z() * \frac{180}{\pi}; \tag{17}$$

2. Diseño y esquematizado de propuesta

2.1. Emisor

Figura 7. Diagrama del flujo de la información desde la parte emisora



Fuente:Propio

La comunicación entre la BNO055 y la ESP32 se establecerá a través de una conexión serial confiable. La BNO055, un sensor de orientación de alta precisión, transmitirá continuamente los datos relacionados con la orientación y los movimientos del objeto o dispositivo al que está conectada.

Al mismo tiempo, la ESP32 estará configurada para recibir las señales de pulso provenientes de los cuatro botones, los cuales actúan como entradas digitales.

Una vez que la ESP32 haya recopilado todos los datos necesarios, los procesará utilizando algorit-

mos específicos que se encargarán de filtrar y limpiar la información. Este proceso garantizará que los datos resultantes sean precisos y confiables, lo que a su vez mejorará la calidad y la exactitud. Una vez que la ESP32 haya completado el procesamiento de los datos y haya obtenido un valor filtrado y limpio, procederá a transmitirlos a través de una conexión Wi-Fi. Utilizando un archivo JSON, se estructurarán los datos de manera adecuada, lo que facilitará su interpretación y uso posterior.

La ESP32 enviará este trama de datos en formato JSON a una dirección IP específica usando el protocolo UDP, que actuará como destino de la transmisión. En esta dirección IP, se encontrará un dispositivo o una aplicación receptora preparada para recibir y procesar los datos entrantes de manera adecuada.

2.2. Receptor

Figura 8. Diagrama de posibles receptores

Fuente:Propio

En la parte receptora, el dispositivo debe estar conectado a Internet y compartir el mismo nodo o punto de acceso (Access Point) al que está conectado el dispositivo emisor. Esta conexión asegurará la comunicación fluida entre ambos dispositivos y permitirá la transferencia de datos de manera eficiente.

Dentro del dispositivo receptor, se ejecutará y operará la simulación correspondiente. Esta simulación estará en constante funcionamiento y recibirá los datos entrantes provenientes del dispositivo emisor. Los datos recibidos se procesarán y se utilizarán para mostrar en pantalla la simulación en tiempo real.

2.3. Diseño e implemntación de Filtro Kalman

Después de haber explorado el concepto del filtro Kalman junto con el fundamento matemático que soporta su implementación, vamos a describir como aplicar el filtro Kalman en un sistema real.

Primero vamos a definir el modelo y las ecuaciones que precisan su comportamiento. Luego, se especificarán los pasos seguidos para inicializar los vectores y las matrices, de la misma forma también se explicarán los métodos empleados para realizar el calculo de la matriz de covarianza, tanto en la etapa de predicción como en la de actualización. Finalmente, se van a presentar algunos gráficos que ilustran estos resultados que se obtuvieron, para así mismo ser analizados con el fin de determinar el nivel de precisión y correspondencia con los datos conocidos.

2.4. Modelamiento del movimiento del sistema

El filtro Kalman utiliza un modelo de movimiento en el que se consideran tanto el ángulo Yaw como la velocidad angular para actualizar la estimación del ángulo en dos pasos principales: la predicción y la actualización. El modelo del sistema es lineal y se representa mediante la matriz de transición de estado A, que se utiliza para modelar la relación entre el estado actual y el estado futuro.

Las ecuaciones que describen el comportamiento de este sistema son:

$$\theta(k+1) = \theta(k) + \omega(k) \cdot dt \tag{18}$$

$$\omega(k+1) = \omega(k) \tag{19}$$

Donde dt, es el tiempo de muestreo.

Estas ecuaciones describen cómo cambia el ángulo y la velocidad angular en función del tiempo, asumiendo que la velocidad angular es constante en el intervalo de tiempo *dt*. Al escribir estas ecuaciones en forma matricial, obtenemos la matriz de transición de estado A que se utiliza. La matriz de transición A que representa el modelo, es la siguiente:

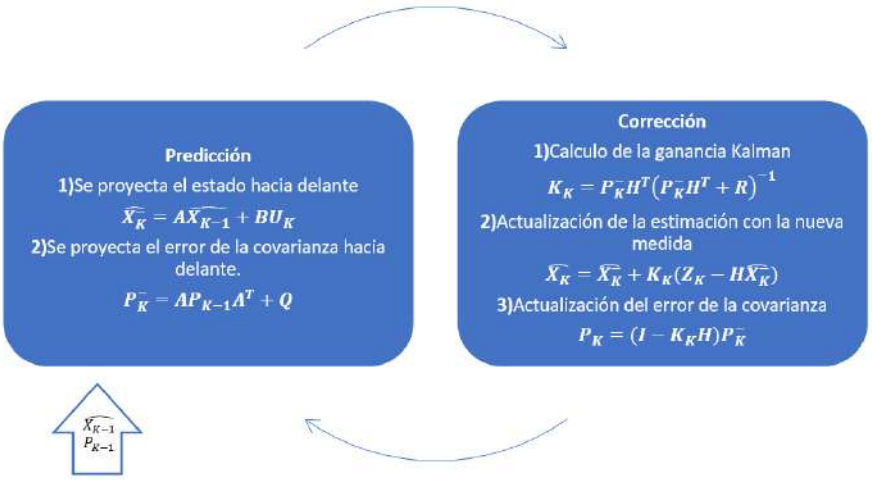
$$A = \begin{bmatrix} 1 & dt \\ 0 & 1 \end{bmatrix} \tag{20}$$

La matriz de transición A se basa en el modelo del sistema y las suposiciones realizadas. En este caso particular, se asume que la velocidad angular $\omega(t)$ es constante en el intervalo de tiempo dt, lo que nos permite simplificar el modelo del sistema lineal y facilita la estimación del ángulo de yaw y su velocidad angular $\omega(t)$.

2.5. Aplicación del filtro Kalman lineal

El filtro de Kalman presenta diversas variaciones, y en este trabajo se utilizará un modelo lineal que asume que el ruido del proceso y el ruido de medición son gaussianos y blancos. El objetivo es estimar el estado instantáneo del sistema, el cual está perturbado por estos ruidos de carácter aleatorio. Para ello, se estima el vector de estado x_k mediante un proceso iterativo que se basa en un modelo del sistema dinámico y en las medidas realizadas por los sensores de las variables que modelan el sistema. De esta manera, el filtro de Kalman busca reducir el efecto del ruido aleatorio o blanco en las estimaciones del estado del sistema.El filtro Kalman esta descrito por el siguiente algoritmo iterativo,

Figura 9. Etapas de predicción y corrección del filtro Kalman



Nota. Esta figura muestra las etapas de predicción y corrección del filtro Kalman.

2.5.0.1. Creación de un estado anterior. La correcta creación del estado inicial es un paso crucial en la implementación del filtro de Kalman, dado que proporciona una estimación preliminar del estado del sistema y minimiza la incertidumbre en la estimación. La calidad de la estimación inicial del estado previo puede influir notablemente en la precisión del filtro de Kalman. Por ejemplo, si la estimación inicial resulta ser muy inexacta, el filtro de Kalman podría tardar más tiempo en converger hacia una estimación precisa del estado del sistema. Por consiguiente, resulta esencial seleccionar valores adecuados para la estimación inicial del estado previo.

En el caso de la primera iteración, se emplean los siguientes vectores iniciales:

$\hat{X}_{k-1}^-$: Vector de estado inicial, el cual almacenará los valores iniciales de las variables que serán filtradas, este vector corresponde al estado anterior. Donde ambos se inicializan en cero, lo que significa que se asume que, al comienzo, tanto el ángulo Yaw como la velocidad angular son cero.

A medida que se adquieran mediciones del sensor y se actualice el filtro de Kalman, estos valores cambiarán para reflejar las estimaciones actualizadas del ángulo Yaw y la velocidad angular.

$$\hat{X}_{k-1}^{-} = \begin{bmatrix} 0 \\ \\ 0 \end{bmatrix} \tag{21}$$

P_{k-1}: Matriz de covarianza del proceso inicial: Esta matriz tiene como función principal corregir los errores en el cálculo del vector de estado inicial en la iteración anterior. Se inicializa con los valores de 1 en la matriz diagonal para sugerir una cierta incertidumbre en la estimación del estado actual.

$$P_{k-1} = \begin{bmatrix} 1 & 0 \\ \\ 0 & 1 \end{bmatrix} \tag{22}$$

En el paso de predicción, el modelo de movimiento lineal se utiliza para estimar el estado futuro del sistema que incluye el ángulo yaw y la velocidad angular.

2.5.0.2. Paso de predicción. Basándose en el modelo lineal, este paso tiene como objetivo predecir el estado X y la covarianza P del estado del sistema en el tiempo k. Para ello, multiplicamos el estado actual $\hat{x}k$ por la matriz A para obtener el estado predicho $\hat{x}k+1^{-}$. Observamos que el ángulo de Yaw predicho se calcula sumando el producto de la velocidad angular actual y

el intervalo de tiempo dt al ángulo de Yaw actual. La velocidad angular predicha simplemente se mantiene igual a la velocidad angular actual como se muestra en la siguiente ecuación:

$$\begin{bmatrix} \theta(k+1) \\ \omega(k+1) \end{bmatrix} = \begin{bmatrix} 1 & dt \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \theta(k) \\ \omega(k) \end{bmatrix} \tag{23}$$

A continuación, se detalla lo que representan cada una de las matrices y la función que desempeñan:

$\hat{X}_k^-$: Vector de estado filtrado o predicho en la iteración actual del filtro de Kalman: Este vector recoge el valor que se predijo en las variables de estado, el cual se obtuvo a partir del modelo lineal.

$$\hat{X}_k^- = \begin{bmatrix} \theta_k \\ \omega_k \end{bmatrix} \tag{24}$$

A: La matriz de transición de estado, la cual representa el modelo. Esta matriz establece la relación entre las variables de estado en dos instantes de tiempo consecutivos y se utiliza para predecir el estado del sistema en el siguiente paso de tiempo (t+1) a partir del estado actual (t).

B: Matriz de entrada de control que representa el efecto de las entradas de control en el estado del sistema, es decir, de aquellas variables de estado de las cuales el sistema no depende. Por lo tanto, su función es modificar las dimensiones de U_K. En el sistema no se utiliza esta matriz porque estamos usando un modelo de movimiento lineal simplificado que no considera entradas de control

externas.

U_K: Vector de variables de control o entrada de control, utilizado para modelar el efecto de las entradas en el estado del sistema, sin embargo, como ya se dijo antes, en esta implementación no se consideran estas entradas de control externas.

P_{k-1}^{-}: Una vez predicho el vector de estado, se procede a realizar la predicción de la matriz de covarianza. Esta matriz de covarianza corresponde al estado anterior, y nos permite conocer el grado de dispersión de la estimación del filtro de Kalman.

Q: Matriz que representa el ruido de la covarianza del proceso o la incertidumbre en el modelo del sistema. Este parámetro debe conocerse antes de aplicar el filtro. La matriz introduce un error relacionado con el ruido del proceso y permite comparar la predicción con la medida, ya que se suma a la predicción de la matriz de covarianza del vector de estado.

2.5.0.3. Paso de actualización. Este paso tiene como objetivo principal utilizar la información más reciente de la medición del sistema para actualizar la estimación previa del estado del sistema y, de esta manera, mejorar la estimación. Al hacerlo, se reduce la incertidumbre en la estimación del sistema, proporcionando una estimación mucho más precisa del estado del sistema. Primero se ajusta la estimación predicha utilizando las mediciones del sensor, luego calculamos la diferencia entre la medición del ángulo de Yaw y la estimación predicha del ángulo de Yaw, para ello utilizamos la matriz de observación H para extraer la estimación predicha del ángulo de Yaw del estado predicho *x_pred*.

Luego, calculamos la ganancia de Kalman K. Esta ganancia determina la importancia relativa que

se le da a la medición del sensor en comparación con la estimación predicha. La ganancia de Kalman se calcula en base a la matriz de covarianza predicha P_pred, la matriz H y el ruido de la medición R, que es la varianza de las mediciones del sensor obtenida a partir de un análisis de las mismas.

Finalmente, se actualiza la estimación del ángulo de Yaw y la velocidad angular utilizando la ganancia de Kalman K_k y el residuo. La ecuación que realiza esta actualización es la siguiente:

$$\hat{x}_k = \hat{x}_k^- + K_k(Z_k - H\hat{x}_k^-) \tag{25}$$

Donde $\hat{x}_k$ es el estado actualizado, $\hat{x}_k^-$ es el estado predicho, K_k es la ganancia de Kalman, y $Z_k - H\hat{x}_k^-$ es el residuo (diferencia entre la medición del sensor Z_k y la estimación predicha del ángulo de Yaw $H\hat{x}_k^-$).

La función principal de esta ecuación es actualizar el estado actual $\hat{x}_k$ utilizando la información de la medición actual y el estado predicho $\hat{x}k^-$ para obtener una estimación más precisa del estado actual. La forma de esta ecuación se deriva de las ecuaciones de movimiento lineal que describen cómo varían el ángulo y la velocidad angular en función del tiempo. La ganancia de Kalman, K_k, se utiliza para ponderar la importancia de la medición actual en la actualización del estado, equilibrando la información de la predicción y la medición.

Además, se actualiza la matriz de covarianza P_k, que representa el grado de dispersión de los valores estimados del vector de estado con respecto a la media de estos. En otras palabras, es una medida de la precisión de la estimación del estado. Si esta matriz es grande, significa que el filtro

no está seguro de la estimación del estado del sistema; por el contrario, si esta matriz es pequeña, indica que la incertidumbre en la estimación del estado del sistema es baja.

2.6. Matrices constantes del filtro

Las matrices H, Q, y R juegan roles fundamentales en la implementación del filtro de Kalman. A continuación, se describen estas matrices y se explica su utilización.

2.6.1. Matriz de Observación H. La matriz H es la matriz de observación y se utiliza para mapear el estado real al estado observado. En este caso, es una matriz identidad y no altera el estado. También se utiliza en el cálculo del residuo, que es la diferencia entre las mediciones observadas y las mediciones predichas por el modelo del sistema.

$$H_{[2x2]} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{26}$$

2.6.2. Matriz de Ruido del Proceso Q. Esta matriz es la que representa el ruido del proceso del sistema y la utilizamos para actualizar la covarianza del estado predicho durante el paso de predicción del filtro Kalman.

$$Q_{[2x2]} = \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix} \tag{27}$$

2.6.3. Matriz de Covarianza de Ruido de la Medición R. La matriz de covarianza de ruido del proceso describe la incertidumbre en las mediciones observadas, que son aquellas que representan el ruido en las mediciones y se usan para calcular la ganancia K para luego actualizar la covarianza del estado durante el paso de actualización del filtro de Kalman.

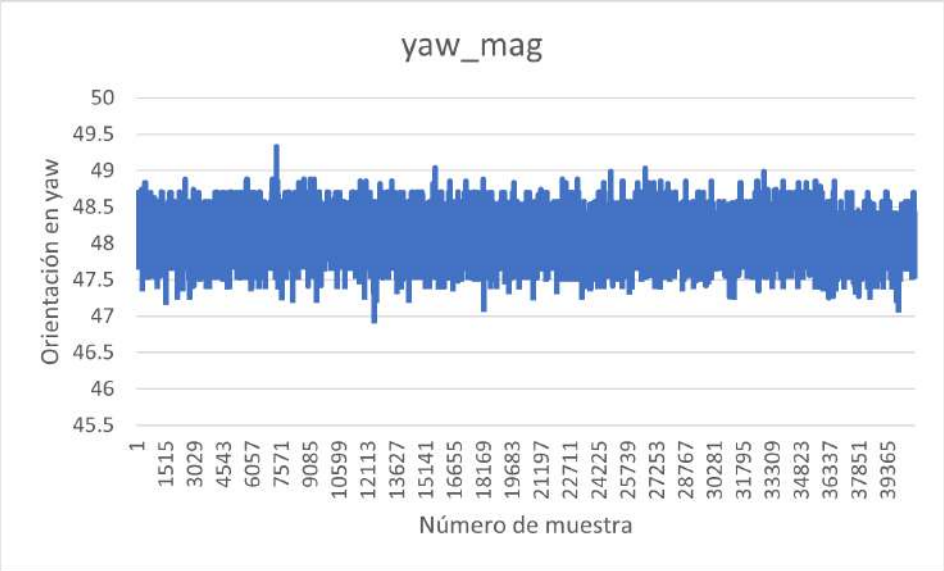
$$R_{[2x2]} = \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix} \tag{28}$$

2.7. Cálculo de las matrices de covarianza Q y R

Se llevó a cabo un método que consiste en considerar matrices diagonales constantes a lo largo de toda la implementación del filtro, colocando en cada elemento de la diagonal el valor que representa la varianza de la variable asociada. Estas matrices permiten determinar el grado de dispersión entre los valores predichos y medidos, así como observar el mecanismo de adaptación del filtro.

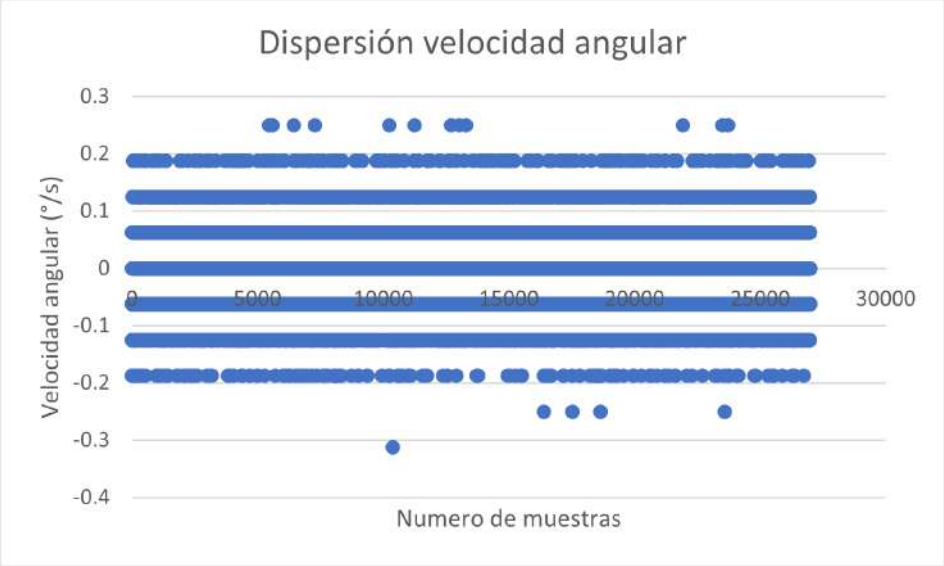
2.7.0.1. Matriz de Covarianza de Ruido de la Medición R. Las gráficas siguientes presentan la dispersión de los datos medidos en yaw y en velocidad angular a lo largo de un período. En el eje horizontal (eje X), se muestra el numero de datos, mientras que en el eje vertical (eje Y) se representan los valores de las mediciones de yaw en grados y las mediciones de velocidad angular en rad/s, respectivamente.

Figura 10. Dispersión del ángulo yaw



Nota. Esta figura muestra la dispersión del ángulo yaw.

Figura 11. Dispersión de la velocidad angular



Nota. Esta figura muestra la dispersión de la velocidad angular.

Al estudiar las gráficas, podemos evaluar la consistencia y variaciones en las mediciones del yaw y la velocidad angular a lo largo del tiempo. Esto nos proporciona información valiosa sobre la exactitud de nuestras mediciones y nos ayuda a identificar posibles fuentes de errores o ruido en los datos recolectados.

Se evaluó las mediciones utilizando la varianza, esta medida estadística se encarga de cuantificar la variabilidad de los datos alrededor de su media:

$$Var_{(x)} = \frac{\sum_1^n (x_i - \overline{X})^2}{n} \tag{29}$$

Esta ecuación de la varianza nos permite estimar la consistencia de las mediciones que se realizaron para así comprender el grado de incertidumbre asociado a ellas.

Durante un período prolongado, se registraron mediciones del ángulo de yaw y de la velocidad angular utilizando un magnetómetro y un giroscopio, respectivamente. Estos datos, recopilados a través del puerto serie, se archivaron en un CSV. Posteriormente, se determinó la varianza de ambas medidas. Esto facilitó la creación de la matriz R, en la que los elementos diagonales representan las varianzas calculadas del ángulo de alabeo y de la velocidad angular.

$$R_{[2x2]} = \begin{bmatrix} 0.00429 & R_{12} \\ R_{21} & 0.00791 \end{bmatrix} \tag{30}$$

En donde se puede observar que cada elemento en la diagonal representa la varianza asociada a cada variable de estado, es decir, la varianza del ángulo de yaw y la velocidad angular.

2.7.0.2. Matriz de Ruido del Proceso Q. La matriz Q se utiliza para representar el ruido en el modelo del sistema, es decir, la incertidumbre que no puede ser modelada explícitamente. Los valores en Q fueron determinados por medio de un proceso de ensayo y error. Hicimos esto iterando sobre diferentes valores de varianza del proceso estimado y observando cómo estos afectan la varianza de la salida del filtro de Kalman. Se comenzó con valores bajos, y luego se ajustaron según fue necesario para lograr un rendimiento óptimo del filtro. Si se observa que el filtro reacciona demasiado lentamente a los cambios, puede ser beneficioso incrementar los valores de Q.

Al Optar por valores bajos en la matriz Q, lo que resultó en un rendimiento notablemente mejorado. Esto sugiere que nuestro modelo del sistema es bastante preciso y la mayoría de la

Tabla 1
Evaluación de la Varianza del Filtro Kalman para Diferentes Valores de la Varianza del Proceso

Varianza del proceso estimada	**Varianza de Kalman**
0.001	0.0079
0.01	0.1015
0.1	0.1432
1	0.1607

Nota. Los valores son presentados para cuatro niveles diferentes de la varianza del proceso estimada.

incertidumbre proviene de las mediciones, no del modelo en sí.

$$Q_{[2x2]} = \begin{bmatrix} 0.001 & Q_{12} \\ Q_{21} & 0.001 \end{bmatrix} \tag{31}$$

A través de este método iterativo, pudimos identificar el valor de varianza del proceso estimada que resulta en la mejor coincidencia entre nuestras predicciones y las mediciones reales. Este enfoque nos permite afinar nuestras suposiciones sobre la incertidumbre en el modelo del proceso y mejorar la eficacia de nuestro filtro de Kalman.

3. Implementación y construcción del prototipo y simulador

3.1. Librerías Arduino

3.1.1. Wire.

La librería **Wire** simplifica la programación de la comunicación I2C al proporcionar una interfaz sencilla y funciones predefinidas para enviar y recibir datos. Permite a los dispositivos Arduino actuar tanto como maestros (iniciadores de la comunicación) como esclavos (dispositivos que responden a las solicitudes del maestro).

Para utilizar la librería **Wire**, es necesario incluirla en el programa de Arduino mediante la instrucción **#include <Wire.h>.** A continuación, se pueden utilizar las funciones de la librería, como **Wire.begin()** para inicializar la comunicación I2C y **Wire.write()** y **Wire.read()** para enviar y recibir datos, respectivamente.

3.1.2. ArduinoJson.

La librería **ArduinoJson.h** facilita la creación, manipulación y análisis de datos en formato JSON en proyectos de Arduino. Permite tanto la generación de datos JSON como la extracción de información de estructuras JSON existentes.

Al utilizar la librería **ArduinoJson.h**, puedes crear objetos JSON, añadir pares clave-valor, arrays y otros elementos de JSON. También puedes serializar estructuras y variables de Arduino en formato JSON. Además, la librería proporciona funciones para analizar y extraer datos de cadenas JSON, lo que facilita la lectura y el procesamiento de información recibida desde fuentes externas, como servicios web o sensores que envían datos en formato JSON.

La librería **ArduinoJson.h** utiliza una cantidad mínima de memoria y se optimiza para trabajar en

sistemas con recursos limitados, como las placas Arduino. Esto la hace adecuada para proyectos que requieren el uso eficiente de la memoria y el procesamiento.

3.1.3. AsyncUDP. La librería **AsyncUDP.h** facilita la implementación de la comunicación UDP en proyectos de Arduino, especialmente cuando se requiere un enfoque asíncrono. La comunicación asíncrona permite enviar y recibir datos UDP sin bloquear la ejecución del programa principal, lo que es especialmente útil cuando se necesitan realizar otras tareas simultáneamente.

Al utilizar la librería **AsyncUDP.h**, puedes crear objetos de socket UDP y utilizarlos para enviar y recibir datagramas de manera asíncrona. Puedes especificar el puerto local en el que escuchar, enviar datos a una dirección IP y puerto remotos, y recibir datos entrantes en un bucle sin bloquear la ejecución principal del programa.

La librería **AsyncUDP.h** también proporciona funciones y eventos para gestionar eventos de recepción y envío de datos, así como para manejar errores de comunicación UDP. Además, es compatible con IPv4 y IPv6, lo que permite la comunicación en redes IP de última generación.

3.1.4. Streaming. Simplifica la impresión de datos en el puerto serie de forma más conveniente y legible. Proporciona una sintaxis similar a la del flujo de datos, lo que permite una manipulación más sencilla y eficiente de los datos que se envían a través del puerto serie.

Al utilizar la librería **Streaming.h**, puedes concatenar y enviar datos de diferentes tipos (como números enteros, flotantes, cadenas de texto, etc.) al puerto serie utilizando el operador de inserción

´. Esto simplifica la tarea de imprimir y formatear múltiples variables en una sola línea de código, en lugar de utilizar múltiples llamadas a la función **Serial.print()**.

3.1.5. Adafruit BNO055 y Sensor. La librería **Adafruit_BNO055.h** se encarga de proporcionar una interfaz fácil de usar para comunicarse con el sensor BNO055. Permite configurar el sensor, leer datos de orientación en diferentes formatos (como ángulos de Euler, cuaterniones o matrices de rotación) y acceder a información adicional, como la temperatura o el estado del sistema. También ofrece funciones para calibrar los sensores internos y mejorar la precisión de las mediciones.

Por otro lado, la librería **Adafruit_Sensor.h** es una librería auxiliar utilizada por la librería **Adafruit_BNO055.h** (y otras librerías de sensores de Adafruit) para proporcionar una interfaz común y consistente para acceder a los datos de los sensores. Esta librería define una clase base llamada **Adafruit_Sensor** que incluye métodos para leer datos y obtener información sobre los sensores, como el rango de medición y la resolución. Al utilizar la librería **Adafruit_Sensor.h**, se facilita el uso conjunto de diferentes sensores de Adafruit en un proyecto, ya que todos ellos siguen la misma interfaz y se pueden manejar de manera consistente.

3.1.6. WiFi. La librería **WiFi.h** permite a los dispositivos Arduino conectarse a redes WiFi existentes, configurar y gestionar conexiones, enviar y recibir datos a través de la red WiFi, y realizar diversas tareas relacionadas con la comunicación inalámbrica.

Al utilizar la librería **WiFi.h**, puedes realizar las siguientes tareas:

- Configurar y establecer una conexión WiFi: La librería proporciona funciones para establecer una conexión a una red WiFi específica, proporcionando el nombre de la red (SSID) y la contraseña (clave de acceso).

- Realizar acciones en la conexión: Puedes verificar si la conexión está establecida, obtener información sobre la dirección IP asignada al dispositivo Arduino, y realizar acciones como desconectar o reconectarse a la red WiFi.

- Enviar y recibir datos: La librería **WiFi.h** ofrece métodos para enviar y recibir datos a través de la conexión WiFi establecida, lo que permite la comunicación con otros dispositivos o servicios en la red.

- Configurar parámetros de red: Puedes configurar parámetros de red como la dirección IP estática, la máscara de subred, la puerta de enlace predeterminada, entre otros.

3.2. Variables y Constantes

Varias variables están declaradas para almacenar datos y configuraciones específicas, como los parámetros del filtro de Kalman, configuraciones de red WiFi, dirección del servidor, puertos, estado de los botones, nivel de batería, lecturas de sensores IMU, entre otros.

También se definen constantes y macros para ajustar valores de tiempo, límites y parámetros de conexión.

3.3. Funciones

- **setup**(): Esta función se ejecuta una vez al inicio del programa y se utiliza para realizar configuraciones iniciales, como configurar pines, establecer la velocidad de comunicación serie y realizar la inicialización de componentes.

- **loop**(): Esta función se ejecuta en bucle y es el punto central del programa. En cada iteración del bucle, se realiza la lectura del tiempo actual y se verifica si se ha alcanzado el periodo de muestreo (Ts). Luego, se llama a la función updateStateMachine() para realizar la transición de estado y ejecutar las tareas correspondientes según el estado actual.

- **connectToWiFi**(): Intenta conectarse a una red WiFi específica y muestra el estado de la conexión.

- **protocoI2C**(): Escanea los dispositivos conectados al bus I2C y verifica la detección del sensor BNO055.

- **LecturaBotones**(): Lee el estado de los botones y actualiza las variables correspondientes.

- **lecturaBno055**(): Realiza la lectura de los datos del sensor BNO055, incluyendo la calibración y los ángulos de orientación (roll, pitch, yaw).

- **String GetJSONString**: Se utiliza para formatear los datos proporcionados en un objeto JSON con una estructura específica y retornarlos como una cadena de texto en formato JSON.

- **UDPConnect()**: Esta función se encarga de establecer una conexión UDP con el servidor especificado mediante la dirección IP y el puerto definidos en las constantes SERVER y PORT. Utiliza el objeto udpClient para realizar la conexión. Si la conexión ya está establecida, no realiza ninguna acción adicional. Si la conexión no está establecida, intenta establecerla llamando a connect() en el objeto udpClient con la dirección IP y el puerto del servidor.

- **UDPSendData(String message)**: Esta función se encarga de enviar los datos proporcionados como una cadena de texto (message) a través de la conexión UDP establecida. Utiliza el objeto udpClient para enviar los datos mediante la función broadcastTo(), que envía los datos a todos los dispositivos en la red local que estén escuchando en el puerto especificado. Si la conexión UDP está establecida, envía los datos proporcionados. Si la conexión no está establecida, llama a UDPConnect() para establecer una nueva conexión antes de enviar los datos.

- **matmul()**: Esta función realiza la multiplicación de matrices.

- **kalman_update()**: Esta función realiza la actualización del filtro de Kalman. Recibe un valor (Z) que representa la medición realizada. Utiliza los valores y parámetros del filtro de Kalman (matrices de transición, observación, ruido del proceso, ruido de la medición, estado y covarianza) para calcular el estado estimado y la covarianza posterior.

3.4. Maquina de Estados

Es una técnica utilizada en la programación para controlar el flujo de ejecución de un programa a través de diferentes estados o etapas. En el código que has proporcionado, se implementa una máquina de estados con cuatro estados: Ïnicio", "Lectura", "Procesamientoz "Transmisión".

Cada estado representa una etapa específica en el programa y se realiza la transición entre estados según ciertas condiciones o eventos. La función **updateStateMachine()** se encarga de determinar el estado actual y llamar a la función correspondiente a ese estado. Luego, se realiza la transición al siguiente estado en función de ciertas condiciones.

Veamos una descripción más detallada de cada estado y las transiciones en el código.

3.4.1. Estado Inicio.

- Este estado se ejecuta al inicio del programa y cuando **conectado** es igual a 0.
- La función **outputInicio()** se ejecuta en este estado y se encarga de llamar a las funciones **connectToWiFi**() y **protocoI2C**() para establecer la conexión WiFi y verificar la detección del sensor BNO055.
- Si la conexión WiFi y la detección del sensor son exitosas (**WifiReady** e **I2CReady** son verdaderas), se actualiza la variable **conectado** a 1.
- En función de **conectado**, se realiza la transición al estado "Lectura"si es igual a 1.

3.4.2. Estado Lectura.

- Este estado se ejecuta después del estado Ïnicioz cuando **proceso** es igual a 0.
- La función **outputLectura()** se ejecuta en este estado y se encarga de llamar a las funciones **LecturaBotones()** y **lecturaBno055()** para leer los botones y los datos del sensor BNO055 respectivamente.
- Después de realizar las lecturas, se actualiza la variable **proceso** a 1.
- Si se ha leído correctamente tanto los botones como los datos del sensor (**leidoBoton** y **leidoIMU** son verdaderos), se realiza la transición al estado "Procesamiento".

3.4.3. Estado Procesamiento.

- Este estado se ejecuta después del estado "Lecturaz cuando **proceso** es igual a 1.
- La función **outputProcesamiento()**se ejecuta en este estado y se encarga de generar una cadena **JSON (json)** a partir de los datos leídos en el estado anterior.
- Después de generar el JSON, se actualiza la variable **proceso** a 2.

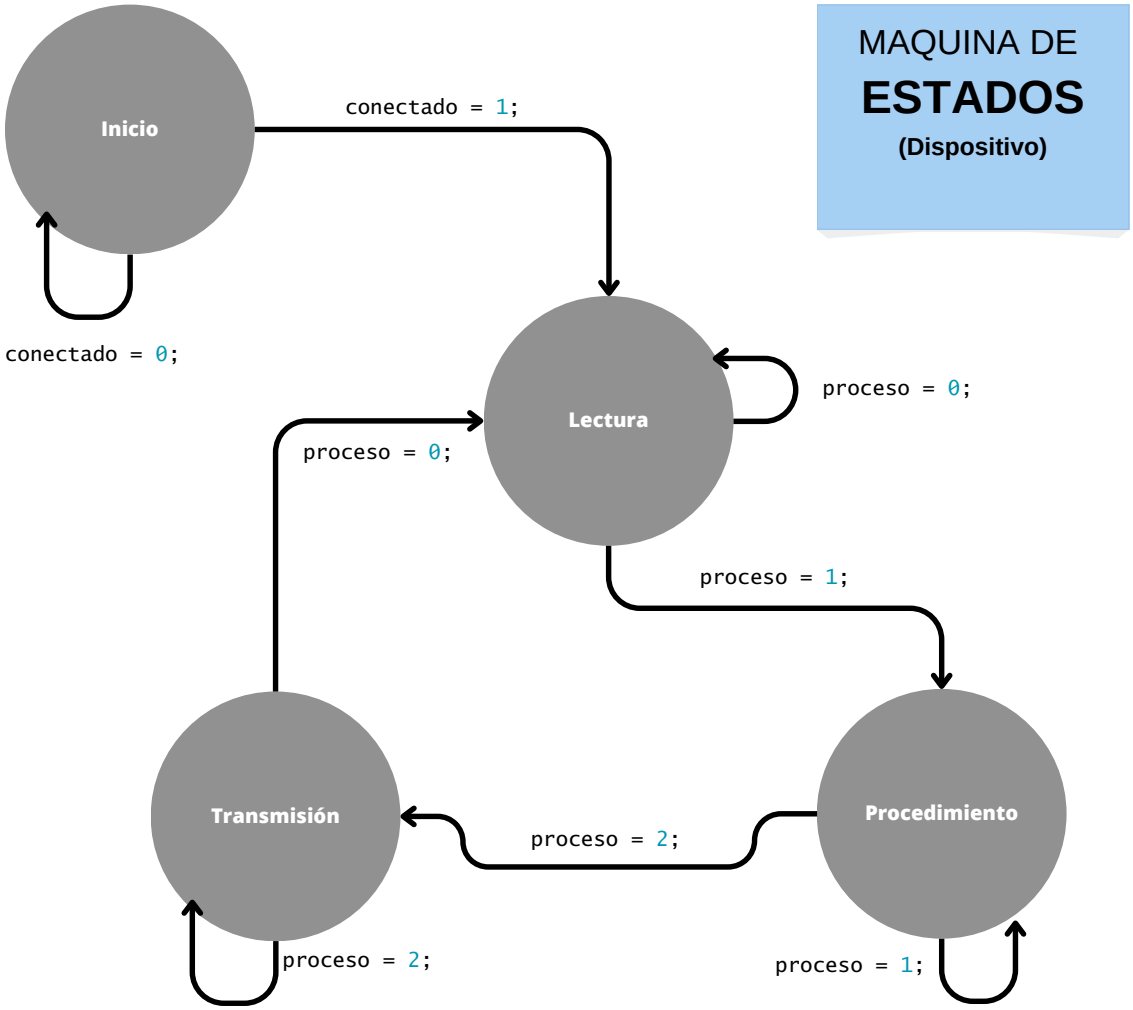
3.4.4. Estado Transmisión.

- Este estado se ejecuta después del estado "Procesamientoz cuando **proceso** es igual a 2.

- La función **outputTransmision()** se ejecuta en este estado y se encarga de enviar los datos **JSON (json)** a través de UDP utilizando la función **UDPSendData()**.
- Después de la transmisión, se actualiza la variable **proceso** a 0.
- A continuación, se realiza la transición al estado "Lectura"para comenzar un nuevo ciclo de lectura, procesamiento y transmisión de datos.

En cada transición de estado, se ejecuta la función de salida de estado correspondiente (outputInicio(), outputLectura(), outputProcesamiento(), outputTransmision()) para llevar a cabo las acciones específicas asociadas con ese estado.

Figura 12. Esquematico de la Maquina de Estados



Fuente:Propio

La máquina de estados permite controlar y organizar el flujo del programa de una manera estructurada y modular. Cada estado se encarga de realizar tareas específicas, y las transiciones entre estados se basan en condiciones o eventos que ocurren en el programa.

3.5. Simulador VR

El pipeline, o flujo de trabajo, para la creación de un juego en Unreal Engine 5 (UE5) abarca una serie de etapas fundamentales que guían el proceso de desarrollo desde la concepción hasta la realización final del juego. Aunque las necesidades y especificaciones de cada proyecto pueden variar, aquí se presenta una descripción general de las etapas comunes en el pipeline de desarrollo.

En la etapa de planificación y diseño, se establece la visión general del juego, definiendo su mecánica, historia, estilo visual y objetivos. Aquí se crea un diseño de alto nivel que establece los fundamentos del juego, como los niveles, personajes e interacciones, y se elabora un plan de desarrollo que incluye hitos, plazos y recursos necesarios.

Posteriormente, en la creación del mundo y los niveles, se utiliza el editor de UE4 para construir el entorno del juego. Esto implica diseñar y construir los niveles, incluyendo la disposición del terreno, la colocación de objetos, edificios y elementos de interacción. Además, se añade la iluminación, efectos atmosféricos y elementos visuales para establecer la atmósfera deseada.

La creación de los assets es otra etapa crucial, donde se crean o importan modelos 3D, texturas, efectos de sonido y otros elementos necesarios para el juego. Aquí se aplican texturas y materiales a los modelos 3D y se animan personajes y elementos interactivos, si es necesario.

En paralelo, la programación y la implementación de la lógica del juego son esenciales. Ya sea utilizando el lenguaje de programación visual Blueprint de UE5 o el lenguaje de programación C++, se define la lógica del juego, incluyendo el movimiento de personajes, detección de colisiones, IA de enemigos y sistemas de juego como puntuación e inventario. Esta etapa es fundamental para

dotar al juego de sus reglas y mecánicas.

A medida que los assets y la lógica del juego se van desarrollando, se procede a la integración de los elementos y la programación. Esto implica la conexión de la lógica del juego con los assets y elementos del mundo, ya sea a través de Blueprints o código en C++. Aquí se realizan pruebas y ajustes para garantizar que todo funcione correctamente y cumpla con las expectativas establecidas.

La optimización y el rendimiento son consideraciones importantes en cualquier proyecto. En esta etapa, se aplican técnicas para mejorar el rendimiento del juego, como el uso de niveles de detalle (LODs), culling para eliminar objetos no visibles y optimización de texturas. Se realizan pruebas exhaustivas en diferentes configuraciones de hardware para asegurar un buen funcionamiento en diversas plataformas.

Finalmente, se llevan a cabo pruebas y se realiza un pulido final del juego. Las pruebas exhaustivas permiten identificar errores, glitches y problemas de jugabilidad, y los comentarios de los testers se utilizan para realizar ajustes y mejoras. En esta etapa, se trabaja en el pulido final del juego, mejorando la estética, la jugabilidad y corrigiendo los últimos errores para lograr una experiencia de juego sólida.

3.6. Diseño del juego

3.6.1. Desarrollo en el Engine.

Se realizó la instalación de la versión 5.1.1. Previo a ello, se requiere Visual Studio 2023 para garantizar su correcto funcionamiento y depuración de código.

Figura 13. Arranque del Motor Gráfico

Fuente:Unreal Engine 5

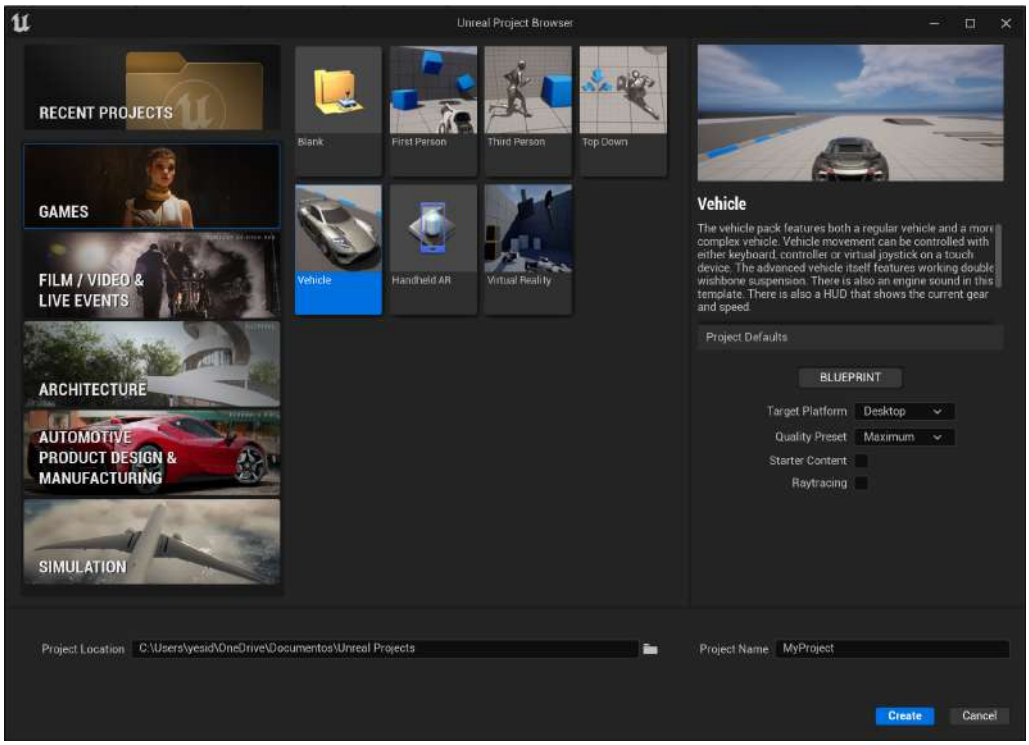
Figura 14. Seleción de versión de motor gráfico

Fuente:Unreal Engine 5

Una vez abierta la suite de Unreal, nos encontramos con la sección de plantillas y tipos de proyectos. Aquí, podemos acceder a una variedad de opciones predefinidas que nos brindan una base sólida para desarrollar nuestro proyecto. Estas plantillas incluyen diversos géneros de juegos, como acción, aventura, disparos, entre otros, así como también aplicaciones interactivas y de visualización arquitectónica. Cada plantilla viene con una configuración predeterminada, que incluye elementos como niveles, personajes, animaciones, sistemas de física y lógica de juego, entre otros. Estas plantillas nos permiten ahorrar tiempo y esfuerzo, ya que proporcionan un punto de partida sólido desde el cual podemos comenzar a construir nuestro proyecto. Además, también

podemos personalizar y modificar estas plantillas según nuestras necesidades específicas, lo que nos brinda flexibilidad en el proceso de desarrollo.

Figura 15. Hub de apertura para la creación de proyectos



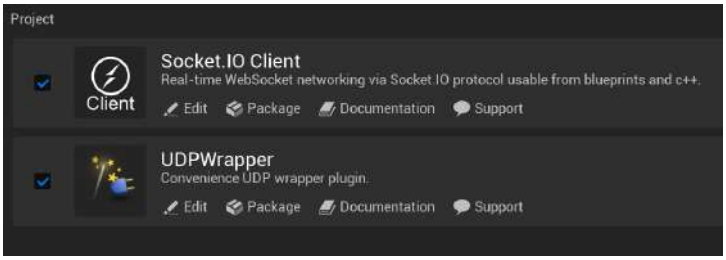
Fuente:Unreal Engine 5

Dentro de la plantilla de juego de carros, realicé modificaciones en el actor del carro creando uno nuevo con una malla diferente. Además de cambiar su apariencia visual, también ajusté las físicas y parámetros de acción del vehículo.

3.6.2. Plugins. Socket IO Client y UDPwrapper son dos plugins populares utilizados en el desarrollo de aplicaciones y juegos en Unreal Engine para facilitar la comunicación en

red.

Figura 16. Plugins usados en el proyecto para su funcionamiento



Fuente:Unreal Engine 5

3.6.2.1. Socket IO Client. Socket IO Client es un plugin para Unreal Engine que permite la comunicación en tiempo real basada en sockets utilizando el protocolo Socket.IO. Socket.IO es una biblioteca de JavaScript que permite la comunicación bidireccional entre servidores y clientes web. El plugin Socket IO Client brinda la capacidad de conectar Unreal Engine con un servidor Socket.IO y enviar y recibir eventos en tiempo real.

Al utilizar el plugin Socket IO Client en Unreal Engine, puedes establecer una conexión con un servidor Socket.IO y enviar y recibir mensajes o eventos personalizados. Esto puede ser útil para la creación de aplicaciones multijugador, chats en tiempo real, actualizaciones de estado en vivo, entre otros casos en los que se requiera una comunicación en tiempo real entre el cliente de Unreal Engine y un servidor Socket.IO.

3.6.2.2. UDPwrapper. UDPwrapper es otro plugin para Unreal Engine que proporciona una interfaz para utilizar el protocolo UDP (User Datagram Protocol). UDP es un protocolo de transporte en la capa de transporte del modelo OSI que se caracteriza por ser sin conexión y no

confiable, pero rápido y eficiente para la transmisión de datos en tiempo real.

Con el plugin UDPwrapper, puedes utilizar el protocolo UDP en Unreal Engine para enviar y recibir paquetes de datos a través de la red. Esto es útil para aplicaciones o juegos que requieren una baja latencia y una transmisión rápida de datos, como juegos multijugador en tiempo real, aplicaciones de streaming de video, sistemas de intercambio de datos en red, etc.

Ambos plugins ofrecen funcionalidades diferentes y se utilizan en diferentes contextos según las necesidades de tu proyecto. Pueden ser descargados e integrados en Unreal Engine desde sus respectivos repositorios o fuentes de origen.

3.7. Blueprints

3.7.1. Actor Principal. Previamente este actor de carro ya tenía programadas sus animaciones, eventos, colisiones, rigging y materiales. Solo fue necesario modificar las entradas de input, conservando únicamente aquellas que deseaba utilizar, como las teclas del teclado o los controles del mando. Estas entradas de input activarían los eventos necesarios para generar movimiento e interacción con el entorno del juego.

Figura 17. Objeto Clase Character que contendrá al personaje y su código

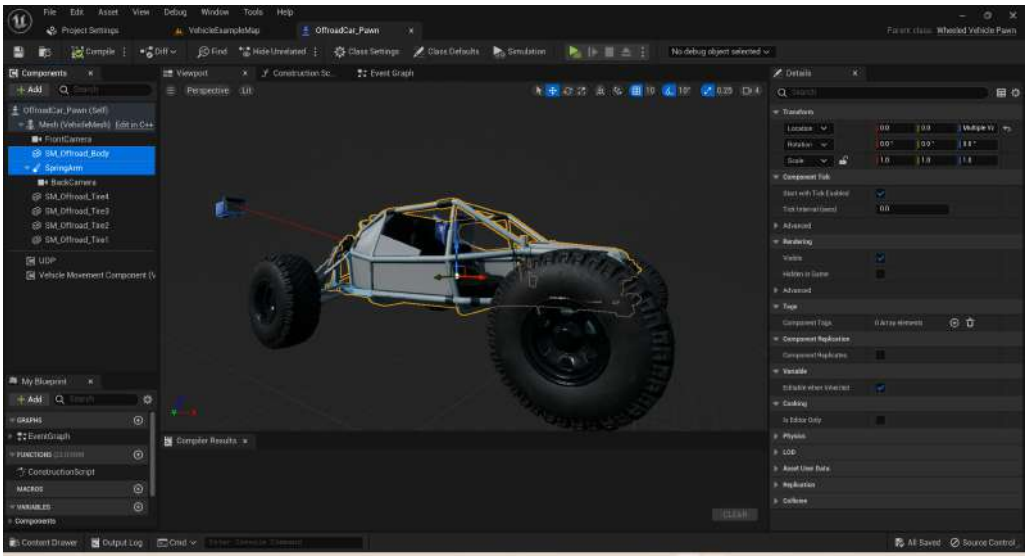
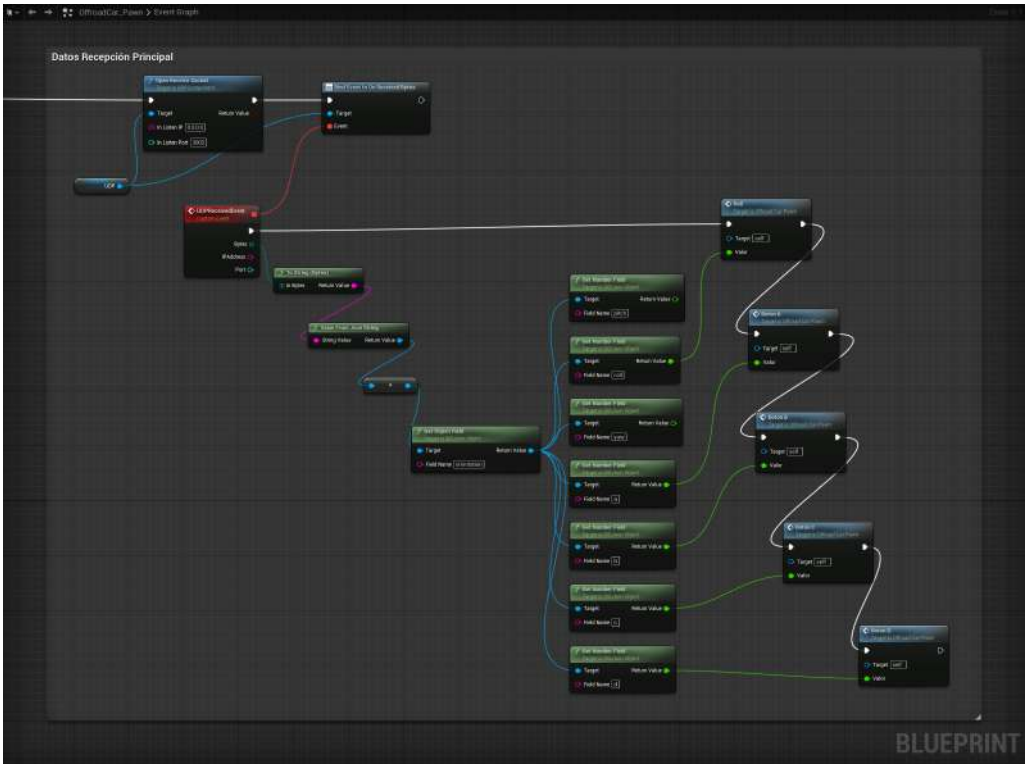


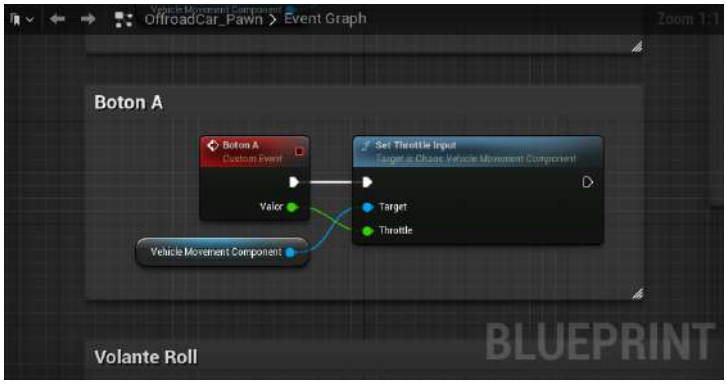
Figura 18. Hoja de Gráficos donde el se elabora el código



3.7.2. Gráfico de Eventos. Aquí podemos observar el bloque .ºpen Receive Socket", el cual permite la recepción de datos enviados. A continuación, se pasa al siguiente bloque, el cual detecta datos recibidos y activa un evento llamado ÜDPReceivedEvent". A partir de este evento, debemos convertir los bytes recibidos en una cadena de datos. Luego, convertimos esta cadena en formato JSON y comenzamos a extraer los valores necesarios según el campo solicitado. Una vez que tenemos los datos de cada campo listos, los ingresamos en las respectivas funciones y eventos previamente creados.

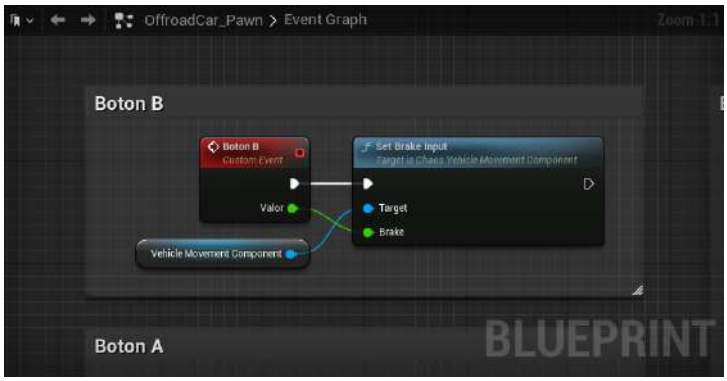
3.7.3. Evento Boton A. Activa la aceleración del vehículo.

Figura 19. Evento/Función Boton A



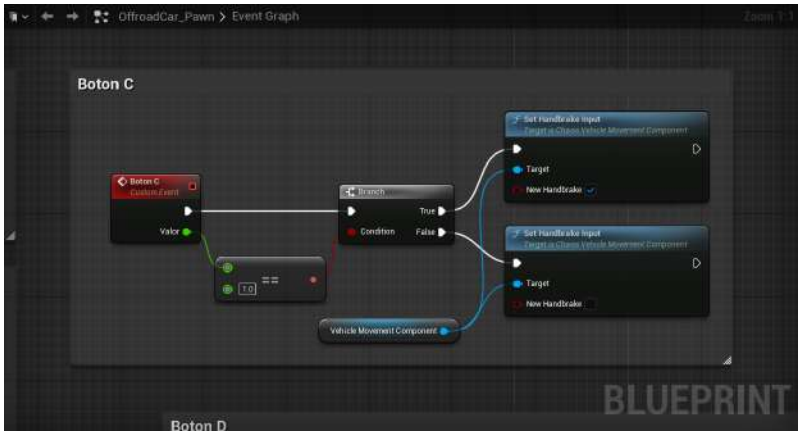
3.7.4. Evento Boton B. Activa el freno de pedal del vehículo.

Figura 20. Evento/Función Boton B



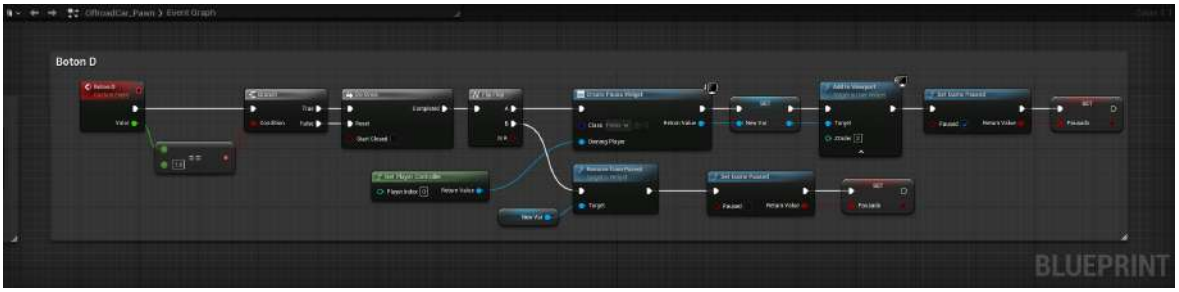
3.7.5. Evento Boton C. Activa el freno de mano del vehículo.

Figura 21. Evento/Función Boton c



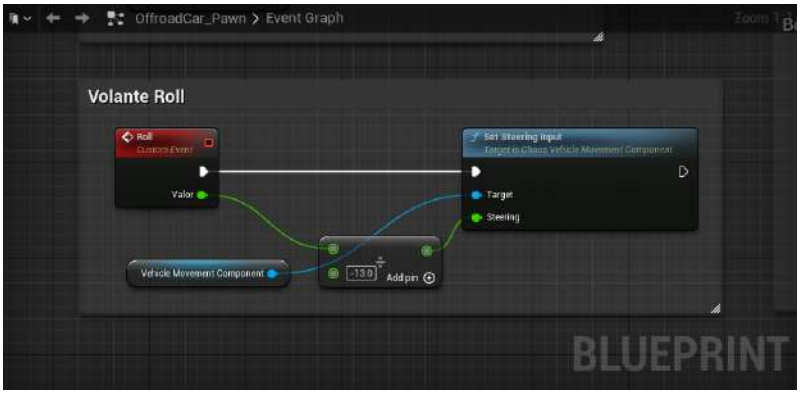
3.7.6. Evento Boton D. Pausa el juego.

Figura 22. Evento/Función Boton D



3.7.7. Evento Volante Roll. Dirección y ángulo en la que el vehículo girará.

Figura 23. Evento/Función Volante Roll



3.7.8. Evento Finalizar la Comunicación. Cierra el puerto en el equipo donde recibe los datos.

Figura 24. Evento/Función Finalizar la Comunicación

3.8. Esquemático

Para iniciar el proceso de creación de una PCB, es fundamental contar con un esquemático que muestre los componentes y las conexiones necesarias para el correcto funcionamiento del circuito. En nuestro caso, utilizamos una herramienta especializada llamada EasyEDA para diseñar tanto el esquemático como el diseño de la PCB de manera eficiente y precisa.

Comenzamos con el esquemático, que representa visualmente el circuito que vamos a implementar. En él, identificamos y ubicamos las diferentes partes del circuito, como resistencias, capacitores, microcontroladores, entre otros componentes. Además, muestra cómo se interconectan, lo que facilita la detección de posibles errores o problemas de diseño.

El esquemático muestra los componentes principales utilizados en el diseño, incluyendo la ESP32 (Kit v1) y la IMU BNO055. Es importante tener en cuenta que ambos componentes tienen un rango de tensión de consumo de 2.5 V a 3.6 V, por lo cual se decidió alimentar la ESP32 a través de su puerto Vin en lugar de utilizar el puerto microUSB.

Para suministrar la energía necesaria al circuito, empleamos una batería de litio 18650(Large, 2020) con una tensión nominal de 3.6 V, la cual se carga hasta un máximo de 4 V. Con el fin de facilitar su uso, incorporamos un módulo de carga y descarga TP4056, que permite cargar las baterías sin necesidad de extraerlas.

Para alimentar la ESP32 con la batería de litio 18650, fue necesario reducir la tensión a 3.3 V. Para lograrlo, utilizamos un regulador de voltaje, que es un componente electrónico capaz de tomar una tensión de entrada y regularla a una tensión de salida fija.

En nuestro caso, incorporamos un regulador de voltaje LM2596(Netto, Saldanha, y Dsouza, 2021), el cual tiene un amplio rango de voltaje de entrada, lo que lo hace útil para convertir voltajes provenientes de fuentes de alimentación variables en el tiempo, como las baterías. Además, cuenta con protecciones incorporadas, como sobrecalentamiento y cortocircuito, y funciona con una tensión mínima de 4.5 V. Como resultado, nos vimos en la necesidad de utilizar dos baterías conectadas en serie para alcanzar una tensión de 8 V, y así ajustar el trimmer para obtener una salida de 3.6 V que permite alimentar la ESP32.

Cada batería tiene una capacidad de corriente de 6800 mA, lo que permite que el circuito funcione de manera continua y sin interrupciones inesperadas. El diseño tiene en cuenta la alimentación y el flujo de energía del circuito, asegurando un funcionamiento correcto y eficiente de los componentes. En base a cálculos realizados utilizando una herramienta virtual, se determinó que la autonomía de este circuito, con la ESP32 funcionando con WiFi, es de 37.7 horas, considerando que la ESP32 consume 180 mA.(de Ingeniería, s.f.)

Para medir el nivel de carga de las baterías, implementamos un divisor de tensión utilizando dos resistencias de 10k de igual valor. A una de las resistencias se le agregó en paralelo un capacitor y un diodo Zener de referencia 1N4002. El diodo Zener se coloca en paralelo con la resistencia R2 del divisor de tensión y actúa como un regulador de voltaje, evitando que la tensión de entrada supere un valor determinado y protegiendo el circuito contra sobretensiones. El capacitor y el diodo Zener se incorporaron con el objetivo de mejorar la precisión de la medición del divisor de tensión y reducir el ruido eléctrico.

Además, se utilizó un filtro paso bajo para el divisor de tensión. Esto se debe a que el divisor de ten-

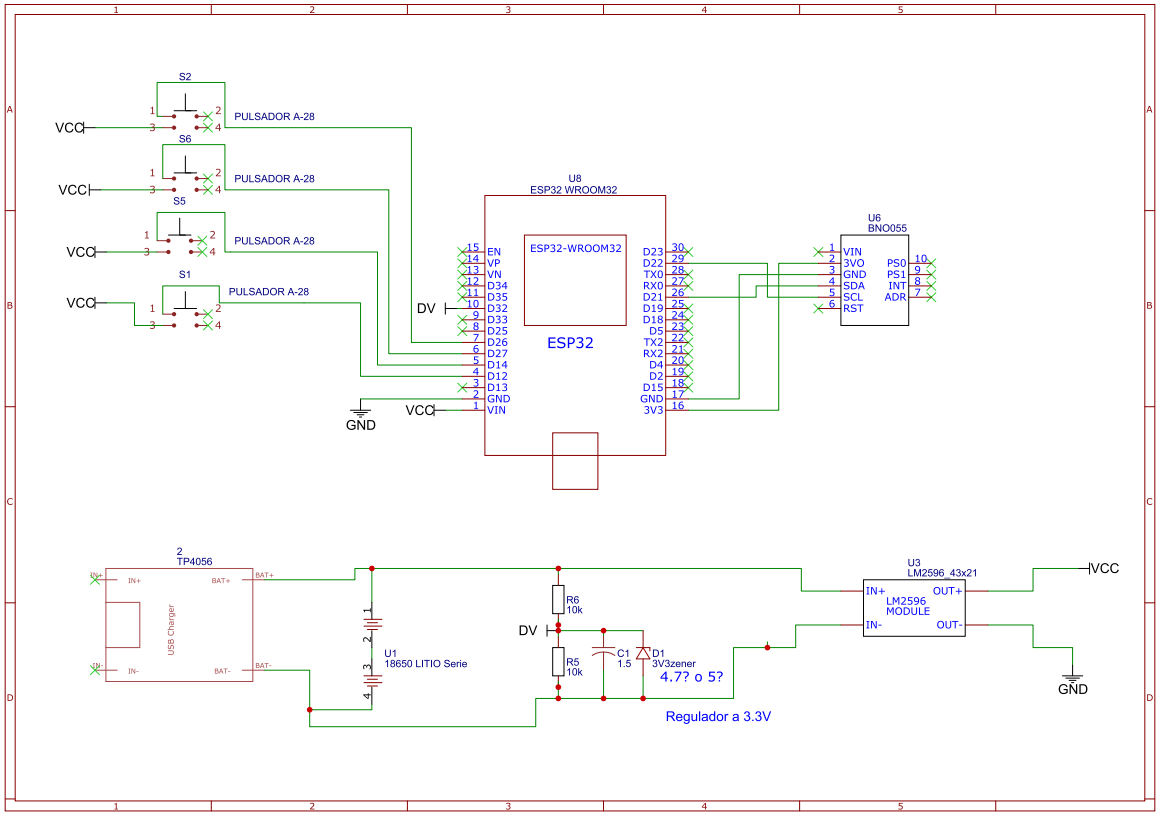
sión puede generar ruido o interferencia no deseada en la señal. El filtro paso bajo es especialmente importante cuando se emplean señales analógicas, ya que son más sensibles a ruidos e interferencias. El filtro de paso bajo ayuda a eliminar las frecuencias no deseadas y garantiza que las señales analógicas se transmitan sin los efectos negativos del ruido de alta frecuencia. En resumen, el filtro paso bajo reduce el ruido y mejora la calidad de la señal.(Juan Antonio Jiménez Tejada, 2008)

Para el calculo del capacitor se usó el valor de las resistencias que ambas son de 10kΩ, de esta manera se calculó el valor del capacitor de la siguiente manera(Sadiku, 2006):

$$C = \frac{1}{2\pi f R_2} = \frac{1}{2\pi(10Hz)(10K\Omega)} = 1.59\mu F \tag{32}$$

En términos de valores comerciales de capacitancia, teníamos la opción de seleccionar un capacitor de $1.5\mu F$ o $2.2\mu F$. Por lo tanto, decidimos utilizar un capacitor de $1.5\mu F$. Este divisor se conecta al pin 32 del Arduino, a través del cual, mediante programación, se determinará el porcentaje correspondiente.

Figura 25. Diseño esquemático del Circuito.



Con respecto a la conexión de la IMU BNO055 con la ESP32, la alimentación VIN de la IMU se conectó a la salida de tensión constante de 3.3V de la ESP32. Las entradas SCL y SDA se refieren a los pines de comunicación del bus I2C del sensor BNO055. SCL significa "serial clock.º reloj en serie, mientras que SDA significa "serial data.º datos en serie. A través de estos pines, el sensor BNO055 se comunica con la ESP32 mediante el protocolo I2C. Los pines correspondientes en la ESP32 son el pin D21 para el puerto SDA y el pin D22 para el puerto SLC, en esta referencia de ESP32.

Los pulsadores están conectados a la ESP32 a través de los pines de entrada analógica. En el

nivel de programación se ha utilizado la resistencia pull-down, donde se hizo uso de la resistencia interna de la ESP32 para mantener la entrada en estado bajo (LOW) cuando no se recibe una señal. Esto asegura que la entrada se mantenga en estado bajo cuando el pulsador no está presionado.(del Valle Hernández, 2022)

Al presionar el pulsador, la entrada se conecta a Vcc y cambia a un estado alto (HIGH). Esto garantiza que la señal de entrada sea clara y evita las fluctuaciones eléctricas no deseadas. Se elimina la necesidad de utilizar componentes electrónicos adicionales al aprovechar la resistencia interna de pull-down de la ESP32. Para habilitar la resistencia interna de pull-down, es necesario configurar el modo de entrada del pin correspondiente (por ejemplo, mediante el uso de la función pinMode()). A continuación, se puede activar la resistencia interna de pull-down utilizando el método esp32.GPIO.PULL_DOWN_EN.

```
1  pinMode(botonA,INPUT_PULLDOWN);
2  pinMode(botonB,INPUT_PULLDOWN);
3  pinMode(botonC,INPUT_PULLDOWN);
4  pinMode(botonD,INPUT_PULLDOWN);
```

Una vez creado y revisado el esquemático en EasyEDA, avanzamos a la siguiente etapa que es el diseño de la PCB.

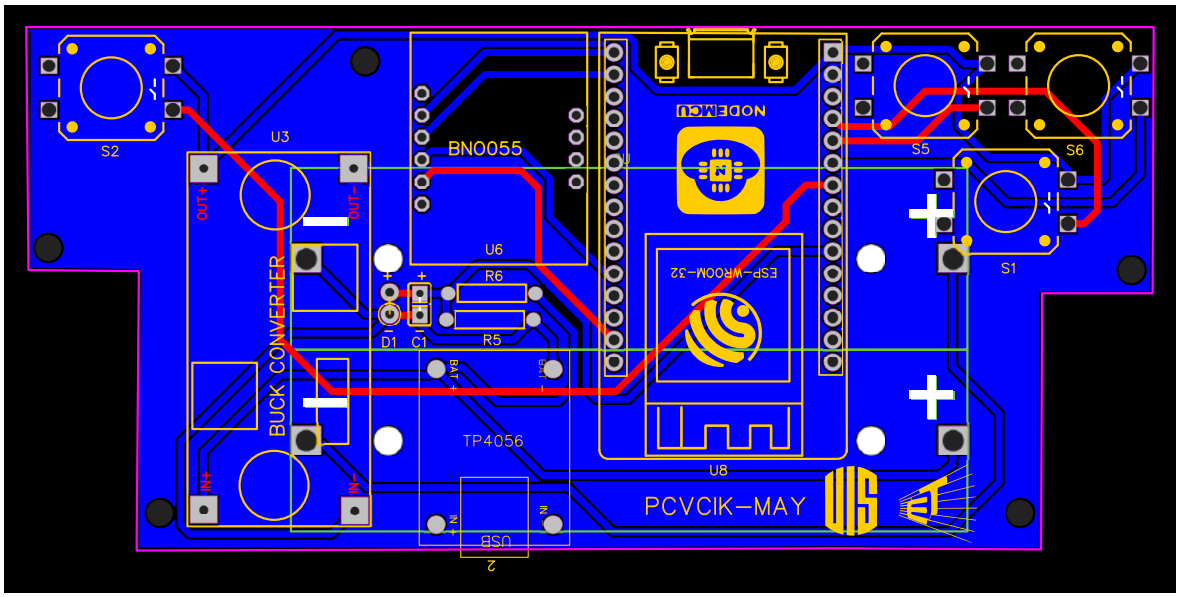
3.9. PCB

La PCB (Printed Circuit Board) es la placa de circuito impreso donde se montarán y conectarán físicamente los componentes electrónicos.

Utilizando EasyEDA, podemos transferir automáticamente el esquemático al diseño de la PCB. Esta herramienta nos permite colocar y conectar los componentes de manera ordenada y eficiente en la placa, siguiendo las conexiones definidas en el esquemático. Además, cuenta con herramientas de diseño avanzadas que facilitan el proceso y mejoran la calidad del diseño.

Durante el diseño de la PCB, se tuvieron en cuenta aspectos como la disposición física de los componentes, la longitud de las pistas, la impedancia y la ubicación de los puntos de conexión. Estos elementos son fundamentales para asegurar un rendimiento óptimo y una funcionalidad adecuada del circuito.

Figura 26. Diseño PCB



El proceso de creación de la PCB implica la transición desde el esquemático hasta el diseño físico en la placa de circuito impreso. Gracias a la interfaz EasyEDA, pudimos llevar a cabo esta tarea de manera eficiente y precisa, asegurándonos de que el circuito esté correctamente represen-

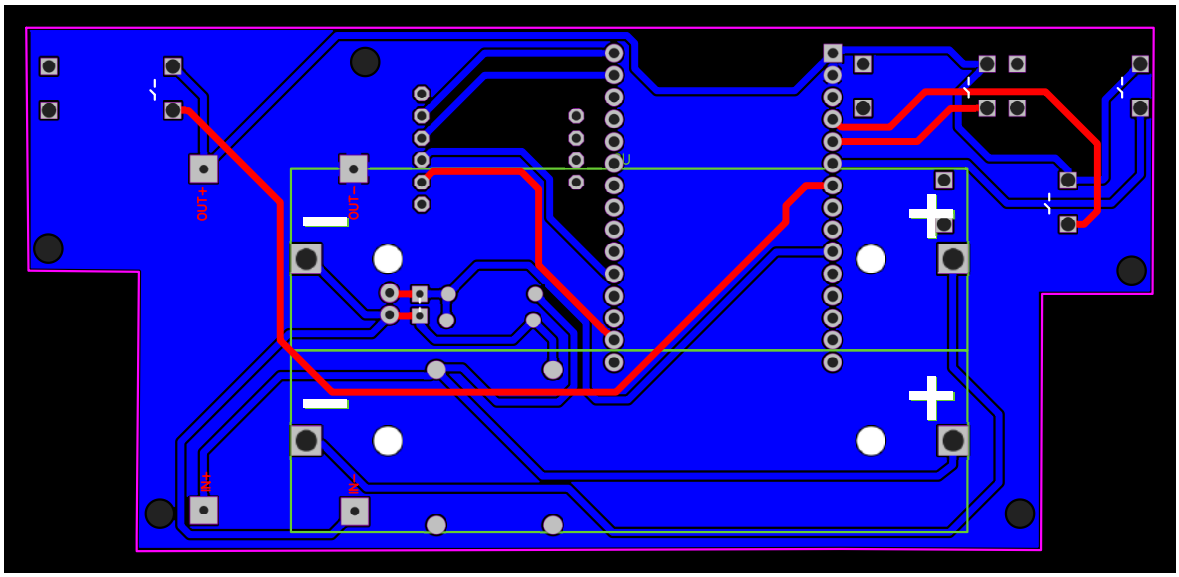
tado en la PCB y cumpla con los requisitos técnicos necesarios. Por ejemplo, las dimensiones de la PCB son una anchura máxima de 129.540 mm y una altura de 59.690 mm.

En cuanto a la configuración de la PCB, decidimos que sea de 2 capas. En la capa inferior se encuentra el plano de tierra y el routeo del circuito. Para determinar el ancho de una pista de routeo de la PCB en función de la corriente que debe soportar, se pueden utilizar diferentes cálculos y fórmulas que dependen de diversos factores. En general, a mayor corriente, mayor debe ser el ancho de la pista para garantizar un rendimiento óptimo y evitar problemas como la generación de calor excesivo o el riesgo de sobrecalentamiento.(Salmony, 2022)

Dependiendo de otros factores como la longitud de la pista, pueden ser necesarios diferentes anchos de pista.

Generalmente, una pista de 0.2 mm de ancho puede soportar hasta aproximadamente 1 amperio de corriente para evitar problemas térmicos y de desgaste en la pista. Sin embargo, para una corriente de 6 amperios y para lograr un rendimiento óptimo, es necesario utilizar una calculadora que aplique la fórmula adecuada,(DigiKey, s.f.) para una vía externa el requerimiento del tamaño de la pista es de 0.6mm, sin embargo, decidimos hacerlo de 0,8mm, de color azul están las pistas de la capa inferior.

Figura 27. Diseño PCB sin capa serigrafía superior



En la capa superior de la PCB se encuentra el enrutamiento en color rojo. Idealmente, para facilitar la soldadura, todo el enrutamiento debería haber estado en la capa inferior. Sin embargo, fue necesario realizar algunas rutas en la capa superior para evitar ubicarlas debajo de la antena de la ESP32. Esto se hizo para evitar posibles interferencias en la señal y así mantener un rendimiento óptimo.

Dejar pistas debajo de la antena puede actuar como una extensión de la misma, lo que afecta su impedancia y puede causar reflexión de la señal, pérdida de potencia y una reducción en el alcance efectivo de la antena. Por lo tanto, se evitó colocar cualquier otro componente o pista debajo o cerca de la antena para no afectar su rendimiento.

La ubicación de los elementos se planificó de manera que la IMU quedara lo más centrada posible, al igual que la ESP32, para reducir la distancia entre ellos y los demás componentes. Esto se hizo considerando que la mayoría de los elementos tienen una conexión directa con la ESP32.

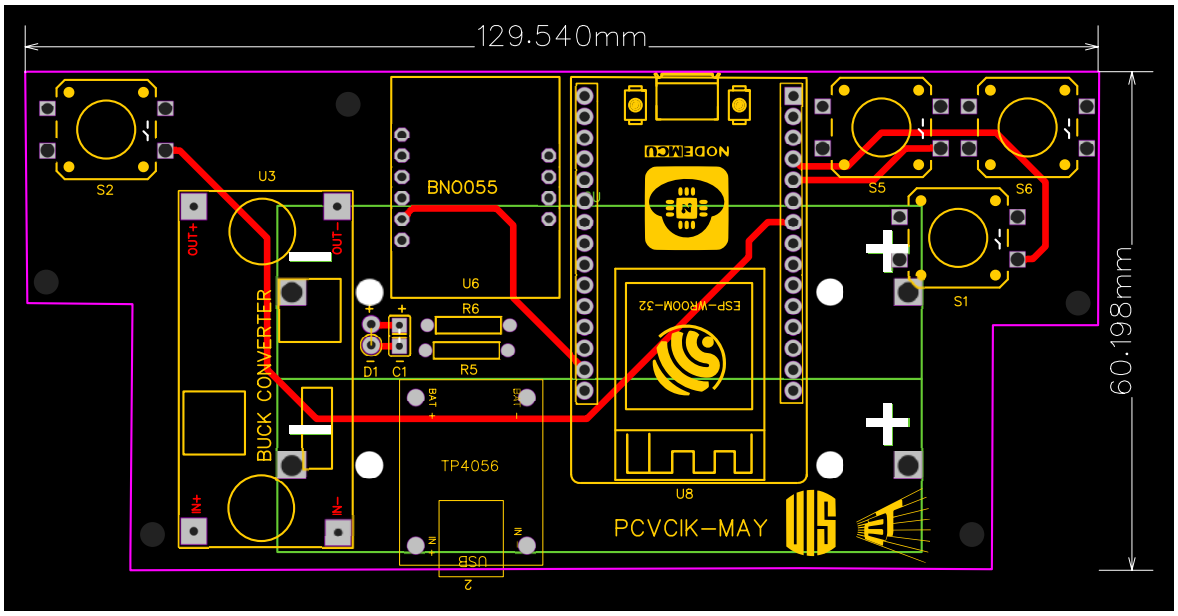


Figura 28. Medidas de la Propuesta de PCB

En el diseño de la PCB, se ha tenido en cuenta la accesibilidad de los puertos de la ESP32 y el módulo TP4056. Para lograrlo, se han ubicado en el borde de la PCB. Esta disposición facilita la conexión y desconexión de los dispositivos, brindando una mayor comodidad al usuario.

Además, se ha aprovechado la capa inferior de la PCB para ubicar la batería en la parte trasera. Esta decisión permite optimizar el espacio disponible en la placa, asegurando una distribución eficiente de los componentes. Al colocar la batería en la capa inferior, se reduce el espacio ocupado en la capa superior, lo cual es beneficioso para el enrutamiento y la organización general del diseño.

Este enfoque de diseño considerado y estratégico asegura una mayor practicidad y eficiencia en el uso de la PCB, al tiempo que maximiza el espacio disponible en la placa.

Figura 29. Modelo 3D de la propuesta visto de frente

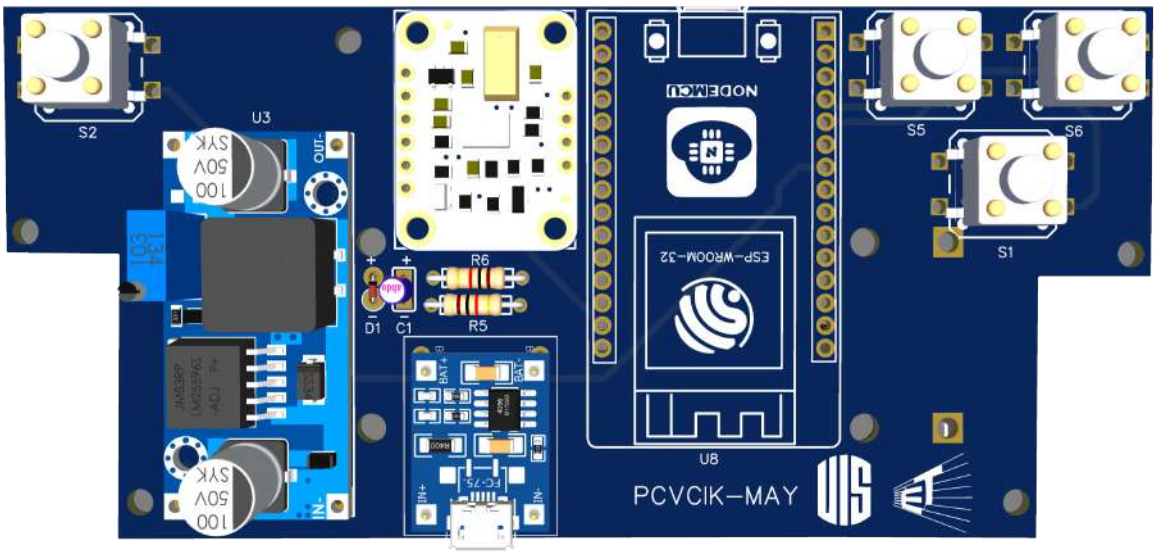
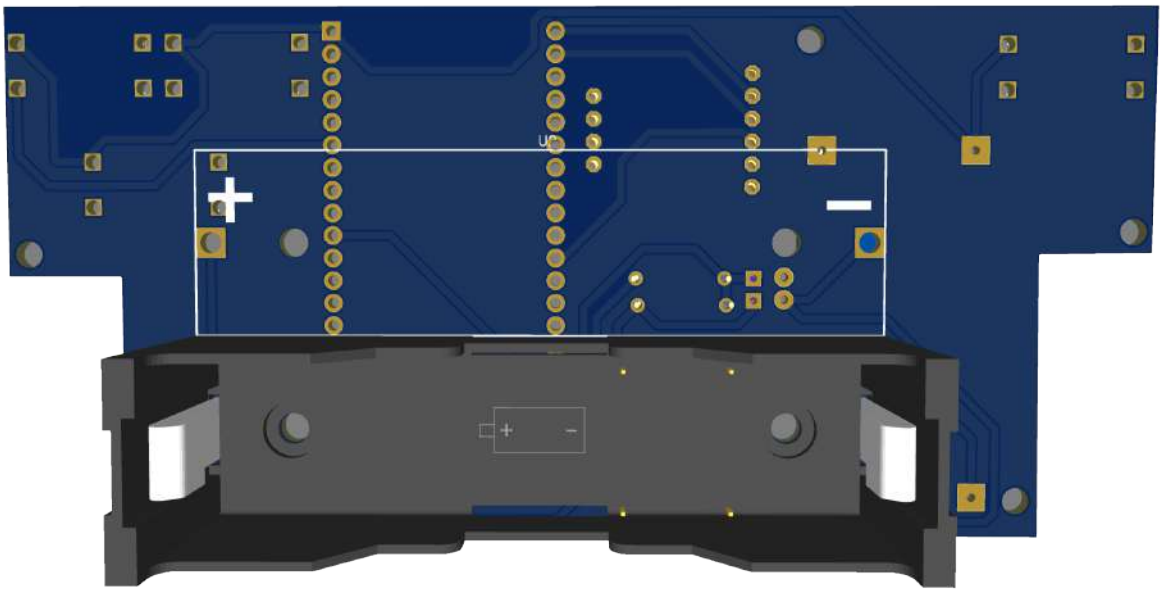


Figura 30. Modelo 3D de la propuesta visto desde atrás



La distribución de los botones se ha planificado cuidadosamente para ofrecer una experiencia de usuario cómoda y ergonómica. Se han ubicado estratégicamente para que sean fáciles de

usar y se adapten bien a la posición de la mano al sostener la PCB. Además, se han dejado orificios de 3 mm para poder atornillar y asegurar la PCB en la carcasa, brindando una mayor estabilidad.
En cuanto a la soldadura de los elementos en la PCB, se ha optado por utilizar pines header macho para permitir un montaje y desmontaje sencillos de la ESP32 y la IMU BNO055. Esta decisión se ha tomado considerando que este es el primer prototipo de la PCB y podría requerir cambios en el futuro. Además, estos elementos son los más costosos en la construcción de la PCB, por lo que se necesita prestar una mayor atención al manipularlos.

3.9.1. Planos de la propuesta de mando. La construcción de la carcasa que cubrirá la PCB fue un aspecto esencial en el diseño de nuestro proyecto. Nuestro objetivo principal fue desarrollar una carcasa que no solo brindara protección física a la PCB, sino que también ofreciera facilidad de manipulación al utilizarla como un control de manejo.
Para lograr esto, nos embarcamos en un diseño personalizado de la carcasa que se adaptara perfectamente a las dimensiones y forma de la PCB. Para esto, estuvimos utilizando la aplicación de SolidWorks la cual nos permitió la elaboración total de esta, para esto, tuvimos en cuenta varios aspectos durante este proceso:

Protección física: La carcasa fue diseñada para proteger la PCB de posibles daños causados por golpes, caídas o contacto accidental con elementos externos. De esta forma se diseñó un modelo 3D impreso en filamento que conta de 3 piezas.

Ergonomía y facilidad de uso: Tuvimos en cuenta la ergonomía al diseñar la carcasa para asegurarnos de que fuera cómoda de sostener y manipular, ya que este dispositivo al ser inalámbrico no posee ningún sistema de soporte. Consideramos factores como el tamaño, la forma, el peso y la disposición de los controles y botones para facilitar el uso intuitivo y evitar la fatiga del usuario durante la manipulación.

Accesibilidad a los componentes: Diseñamos la carcasa con aberturas de carga estratégicamente ubicadas que permitieran acceder a los componentes de la PCB sin dificultad como lo son el módulo de carga y la ESP32. Esto facilita el mantenimiento, las actualizaciones o cualquier ajuste necesario en el futuro, sin comprometer la integridad de la carcasa.

La construcción de la carcasa fue un aspecto crucial en nuestro diseño de la PCB. Buscamos crear una carcasa personalizada que brindara protección física, facilidad de uso, accesibilidad a los componentes y una estética atractiva. Al tener en cuenta estos aspectos, logramos desarrollar una carcasa que complementa y realza la funcionalidad de nuestra PCB como un control de manejo eficiente y confiable.

En este trabajo, se presentan los resultados de la fabricación de tres piezas impresas en 3D utilizando filamento como material de construcción. Estas piezas fueron diseñadas para cumplir con requisitos específicos y han sido acotadas y dimensionadas cuidadosamente para garantizar su precisión y funcionalidad. Las imágenes acotadas de las piezas impresas en 3D proporcionan una representación visual clara de los resultados obtenidos. La acotación en el diseño de objetos es de vital importancia, ya que permite comunicar de manera precisa las medidas y detalles necesarios

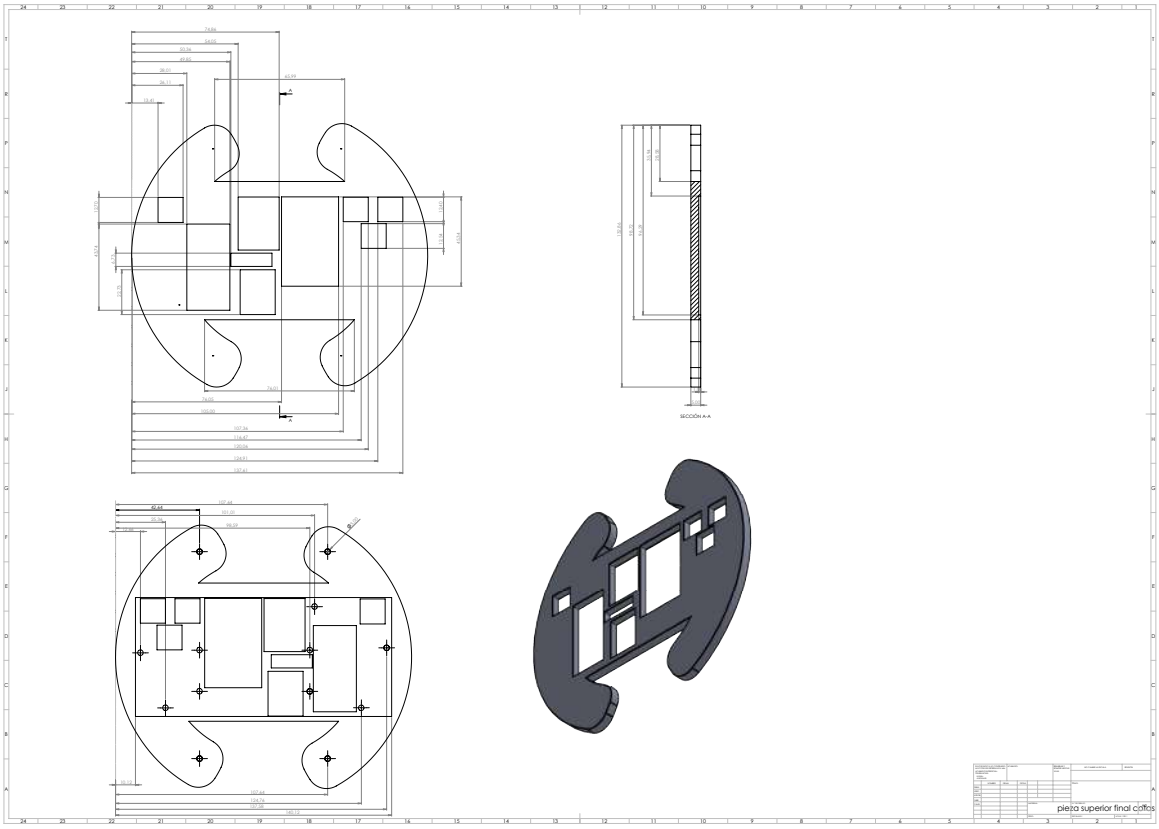
para la fabricación, construcción o diseño de un proyecto. En el caso de nuestro diseño, decidió aplicar acotaciones cuidadosas para asegurar la correcta interpretación y construcción del objeto.

La inclusión de acotaciones en los dibujos técnicos contribuye a evitar errores y malentendidos en la interpretación de los planos. Al proporcionar medidas claras y precisas, se facilita la comprensión de la geometría del objeto y se asegura que se construya de manera adecuada, cumpliendo con los requerimientos de diseño para esto nos basamos en las recomendaciones de acotaciones(Diaz, 2006) y sus normas(de Tecnologías Educativas y Formación del Profesorado, 2009).Además, la acotación es esencial para mantener la calidad y la precisión en la representación de objetos técnicos en los planos y dibujos técnicos.

La acotación en el diseño de objetos desempeña un papel fundamental al garantizar la claridad, precisión y calidad en la representación de las medidas y detalles necesarios para la construcción del proyecto. Al aplicar acotaciones adecuadas, se evitan errores y se asegura la correcta interpretación de los planos, contribuyendo al éxito del proyecto en términos de funcionalidad, estética y cumplimiento de los requisitos de diseño.

Estas imágenes permiten una evaluación precisa de las dimensiones y geometría de las piezas, así como la apreciación de los materiales y tecnología utilizados en su fabricación. Estas piezas impresas en 3D son un testimonio del potencial de la impresión 3D en la creación de componentes funcionales y personalizados.

Figura 31. Pieza superior



Se presenta en la Figura 31 es la primera pieza impresa en 3D, la cual fue diseñada para ser la carcasa superior de la PCB. El diseño de esta pieza se basó en la disposición y el tamaño de los componentes de la PCB. Se tuvo en cuenta la posición y el espacio requerido por los componentes para asegurar un ajuste perfecto y una funcionalidad óptima para esta exactitud se usó la aplicación de EASYEAD donde ahí se elaboró la PCB y tenía las medidas exactas de la construcción, se hizo variación entre 2 y 3 mm entre cada componente para poder encajar la pieza en la PCB y que los elementos pudieran sobresalir.

Esta pieza fue concebida con el propósito de mejorar la comodidad en el uso de la PCB. Su diseño

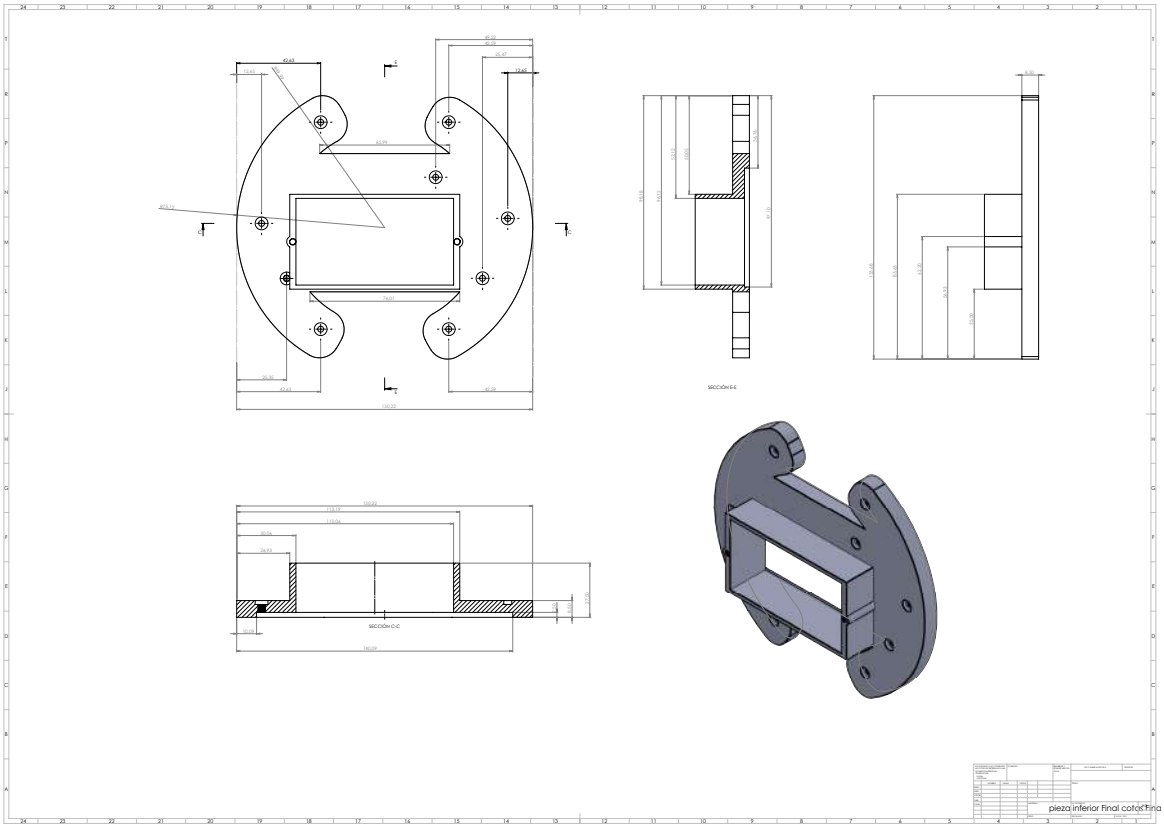
permite un acceso fácil a los componentes y facilita la manipulación de la PCB durante su funcionamiento. La decisión de dejar los elementos a la vista en la carcasa fue tomada estratégicamente para permitir una visibilidad clara de los componentes y su funcionamiento, lo cual es especialmente útil durante el proceso de depuración y mantenimiento.

La elección de una carcasa impresa en 3D ofrece ventajas significativas, como la posibilidad de personalizar el diseño y lograr una integración perfecta con la PCB. Además, el material utilizado en la impresión fue el filamento plástico de alta calidad con un diámetro de 1.75mm, proporciona una protección adecuada contra daños físicos y aislamiento eléctrico.

Esta primera pieza impresa en 3D es un componente clave en el diseño de la carcasa de la PCB, ya que combina funcionalidad y estética. Su diseño meticuloso y su adaptación precisa a la PCB garantizan una experiencia de uso cómoda y eficiente, al tiempo que permiten una visualización clara de los componentes y su rendimiento.

Es importante resaltar que la Figura 31 muestra la pieza impresa en 3D, destacando su diseño basado en los componentes de la PCB y su capacidad para hacer más accesible y visualmente atractivo el uso de la PCB.

Figura 32. Pieza inferior



La pieza inferior de la carcasa, representada en la Figura 32, desempeña un papel fundamental en el ensamblaje de la PCB y en la protección de la batería ubicada en la parte trasera de la PCB. Esta pieza está diseñada de manera que se puedan atornillar tanto la pieza 1 de la carcasa como la propia PCB, garantizando una sujeción segura y estable.

Para lograr esto, se incorporaron 9 agujeros en la pieza inferior de la carcasa, los cuales se alinean perfectamente con la PCB y la pieza 1 de la carcasa. Estos agujeros fueron dimensionados cuidadosamente para permitir el paso de tornillos de 3 mm sin sobresalir o dificultar la manipulación de la pieza. Además, se añadieron dos agujeros adicionales en el área de la batería, los cuales encajan

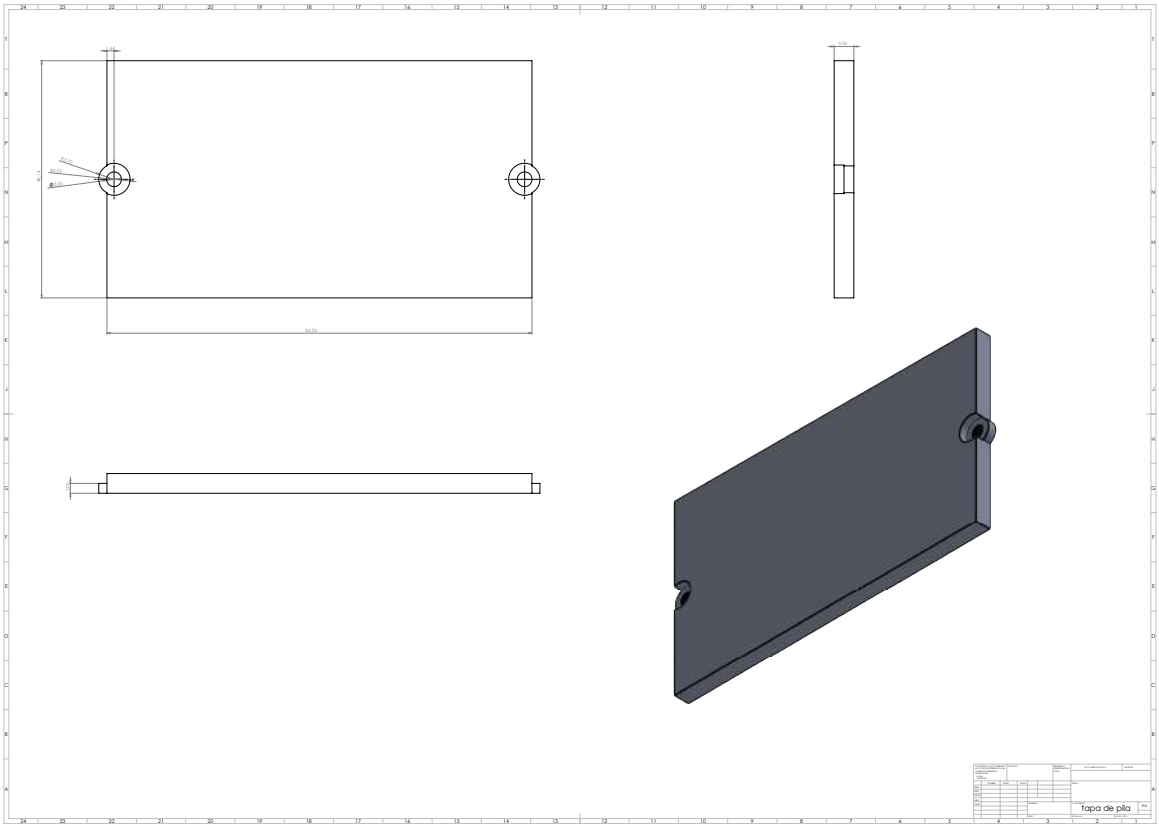
con precisión en el relieve de la pieza 3 de la carcasa.

Durante el proceso de diseño de la pieza inferior, se utilizó la función de taladro avanzado en el software Solidworks, con la referencia 3M, para garantizar la compatibilidad con los tornillos de 3 mm que se utilizarán en el ensamblaje. Esto asegura que los tornillos se ajusten perfectamente a los agujeros sin generar molestias ni interferencias en la manipulación de la pieza.

En conjunto, la pieza inferior de la carcasa juega un papel crucial al proporcionar una conexión segura y estable entre la PCB y las demás partes de la carcasa. Además, ofrece una cobertura protectora a la batería, garantizando su seguridad y minimizando cualquier riesgo de daño o interferencia externa, sin embargo, ninguna de las piezas protege a la PCB del agua y/o polvo.

La pieza inferior de la carcasa se ha diseñado con precisión para permitir el ensamblaje adecuado de la PCB y proporcionar protección a la batería. La incorporación de agujeros estratégicos y la utilización de la función de taladro avanzado en el software de diseño han permitido lograr un ajuste perfecto con los tornillos de 3 mm, asegurando la integridad y funcionalidad del conjunto de la carcasa.

Figura 33. Pieza Tapa para las baterias



La Figura 33 muestra la tercera pieza impresa en 3D, que desempeña un papel crucial al completar la protección de la batería. Esta pieza está diseñada para ser atornillada a la pieza 2 de la carcasa, en lugar de ser encajable, debido a la intención de evitar que las baterías sean extraíbles. El diseño de esta pieza se basó en la necesidad de mantener la batería asegurada y protegida dentro de la carcasa. Al atornillar la tercera pieza a la pieza 2, se garantiza una sujeción firme y segura de la batería, evitando posibles movimientos o deslizamientos indeseados.

Además, se tuvo en cuenta que las baterías cuentan con un sistema de carga integrado en la PCB. Al hacer que las baterías no sean extraíbles, se asegura que permanezcan conectadas al sistema de

carga de manera constante y segura, evitando cualquier interrupción en el proceso de carga.

Este enfoque de diseño ofrece una solución eficiente para la protección y carga de las baterías dentro de la carcasa. Al mantener las baterías fijas y seguras, se minimiza el riesgo de desconexiones accidentales o problemas de carga, lo que garantiza un funcionamiento confiable y continuo del dispositivo.

Figura 34. Vista frontal del prototipo en su etapa final

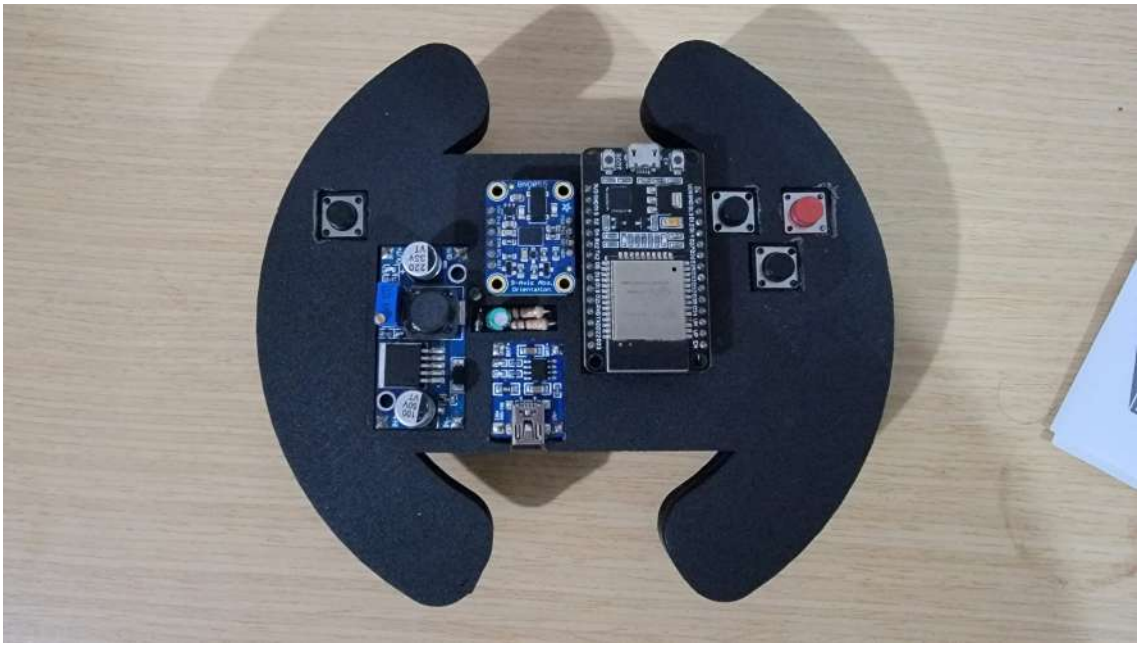


Figura 35. Vista posterior del prototipo en su etapa final

El diseño de la PCB es crucial para el éxito del prototipo, y por eso se ha considerado la comodidad y facilidad de uso en la soldadura y el ensamblaje. El uso de pines header macho garantiza la estabilidad de los componentes y facilita la corrección de errores o modificaciones en el diseño en el futuro, si fuera necesario. La elección de los pines header macho es una precaución necesaria para garantizar comodidad y facilidad de uso durante el proceso de soldadura de la PCB, y permite una mayor flexibilidad para realizar cambios y mejoras en el diseño en el futuro.

El diseño realizado en SolidWorks ha logrado una carcasa que se ajusta de manera precisa y eficiente a la PCB. La imagen 34 evidencia el resultado satisfactorio de este diseño, demostrando una integración óptima y un enfoque funcional y eficiente en el desarrollo del proyecto, además cumple con las restricciones de peso, teniendo un peso total de 228g.

Tras completar la construcción del prototipo de control, se han incurrido en diversos gastos ne-

cesarios para materializar esta innovadora solución. A lo largo del proceso de desarrollo, se han considerado múltiples elementos para garantizar la calidad y eficiencia del control virtual, lo que ha resultado en un total de 397.800 pesos colombianos invertidas en el prototipo de control final del proyecto que lo podemos ver de forma detallada en la tabla 2.

Tabla 2
Costos de los Componentes

Componente	**Costo (COP)**
Divisor de tensión	2.000
Pulsadores	4.000
ESP32	36.800
TP4056	8.000
LM2596	12.000
BNO055	180.000
Impresión Carcasa	95.000
Impresión PCB	60.000
Total	**397.800**

Los gastos se han distribuido en diversas áreas clave, que incluyen la adquisición de componentes electrónicos de alta calidad para la IMU BNO055 y el filtro de Kalman, garantizando así mediciones precisas y confiables. Además, se ha destinado una parte del presupuesto a la obtención de la placa de desarrollo ESP32, que permite la transmisión de datos de manera inalámbrica al motor gráfico Unreal Engine 5, enriqueciendo la experiencia de conducción virtual.

4. Análisis de resultados

Con el objetivo de valorar la efectividad del prototipo e interpretar su comportamiento, implementamos un enfoque de pruebas dividido en dos etapas. Inicialmente, realizamos pruebas a los componentes esenciales individualmente con el propósito de garantizar su funcionamiento adecuado. Posteriormente, una vez que cada componente fue probado y validado, estos se integraron para constituir el prototipo completo. Finalmente, llevamos a cabo una prueba exhaustiva que nos permitió obtener una visión integral del desempeño y eficiencia del prototipo en su totalidad.

4.1. Evaluaciones del rendimiento del Filtro de Kalman.

En esta sección, se llevará a cabo un análisis de los datos obtenidos, en el cual se realizarán una serie de comparaciones para resaltar los resultados obtenidos por el algoritmo del filtro Kalman.

Tabla 3
Comparación de Métricas de Yaw

Métricas	**Filtro Kalman**	**Sin Filtro**
Varianza	0.00791332	0.069787784
Desviación Estándar	0.08895685	0.264173776

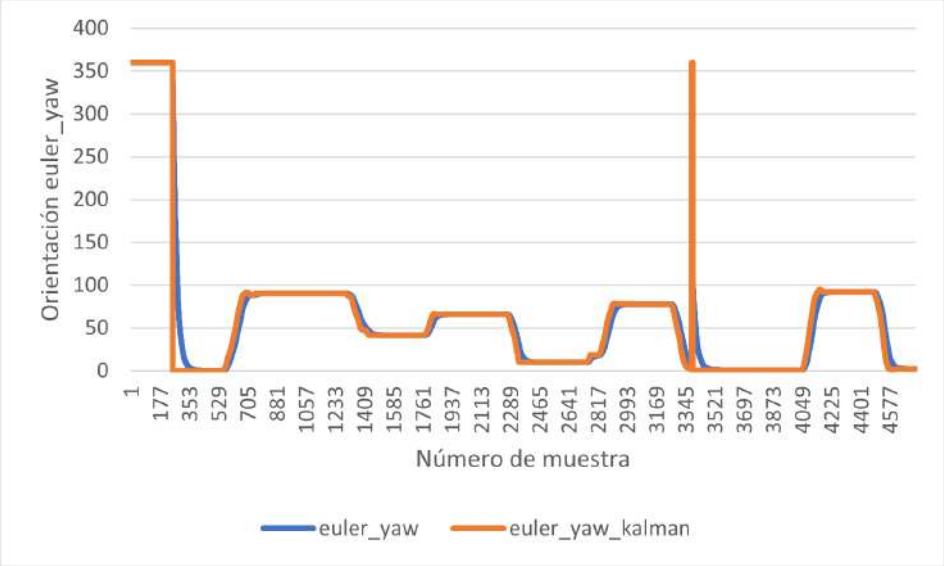
Nota. La tabla compara las métricas del Yaw cuando se utiliza el Filtro Kalman y cuando no se utiliza. Se muestran los valores de la varianza y la desviación estándar para ambos métodos.

En la tabla 3 se puede observar cómo, al comparar las medidas, resulta evidente que al aplicar el filtro Kalman sin ajustar adecuadamente la matriz de covarianza del ruido de las mediciones

(R), se obtiene una varianza y una desviación estándar mayores en comparación con la medida directa del ángulo Euler Yaw proporcionada por el sensor BNO055. *"Es importante destacar que el sensor emplea un algoritmo de fusión de sensores basado en un filtro de Kalman para entregar datos de orientación en ángulos de Euler con alta precisión"* (PRAJWAL SHENOY y VARADHAN, s.f.). La eficiencia de este algoritmo se manifiesta en la nula desviación estándar y la varianza en los datos del ángulo Euler.

Esta comparación es importante porque nos demuestra que es crucial ajustar correctamente la matriz de covarianza para lograr resultados precisos y confiables, o incluso para mejorar las mediciones de la orientación. Este experimento nos permite evidenciar la alta precisión de la medida del ángulo Euler proporcionada por el BNO055, ya que el algoritmo de fusión de sensores está optimizado para ofrecer un rendimiento bastante alto en la determinación de la orientación.

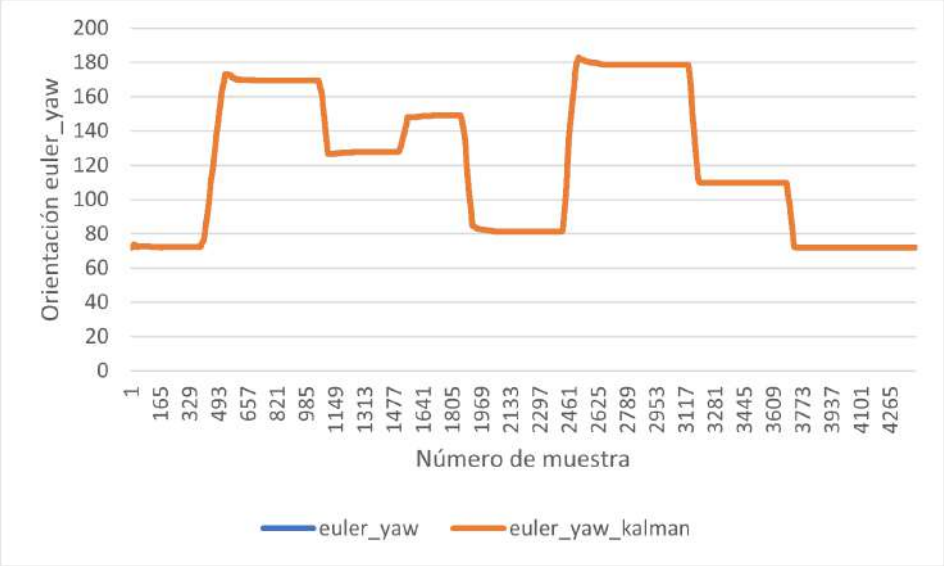
Figura 36. Inestabilidad de Yaw sin Matriz de Covarianza Adecuada



Nota. Esta figura muestra la inestabilidad del ángulo calculado Yaw cuando no se utiliza una matriz de covarianza adecuada.

La gráfica representada muestra el comportamiento de dos enfoques para la estimación de la orientación. Al analizarla, se observa la falta de estabilización en la salida del estado de Kalman cuando la matriz de covarianza del ruido no se ajusta correctamente. Por otro lado, la salida del algoritmo de fusión de sensores del BNO055 exhibe una estabilización mucho mejor y menor dispersión de los datos.

Figura 37. Estabilidad de Yaw con Matriz de Covarianza Adecuada



Nota. Esta figura muestra la estabilidad del Yaw cuando se utiliza una matriz de covarianza adecuada.

Cuando ajustamos la matriz de covarianza del ruido R con los valores adecuados, se observa que el filtro Kalman optimizado y el algoritmo de fusión de sensores del BNO055 siguen una misma trayectoria y presentan una estabilidad similar en los datos de orientación. De esta manera, se logra una mayor precisión en la estimación de la orientación.

Para evaluar la eficacia del filtro de Kalman en la mejora de las mediciones, se realizó un experimento en el que se estimó el ángulo yaw utilizando el magnetómetro del IMU.

La siguiente tabla muestra la comparación entre las métricas del ángulo sin filtrar y el ángulo después de aplicar el filtro Kalman. Las dos métricas clave son la varianza y la desviación estándar.

Al examinar detenidamente la tabla 4 es notable que tanto la varianza como la desviación

Tabla 4
Comparación de las métricas con y sin el uso del Filtro de Kalman

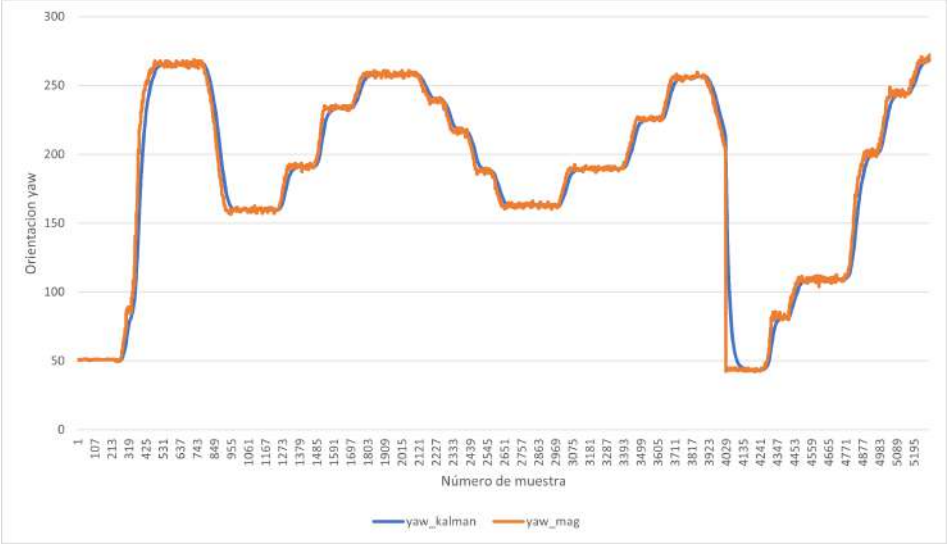
Métrica	**Filtro Kalman (Yaw)**	**Sin Filtro (Yaw)**	**Reducción (%)**
Varianza	0.00791	0.06979	88.66
Desviación Estándar	0.08896	0.26417	66.33

Nota. Esta tabla compara las métricas del Yaw con y sin el uso del Filtro de Kalman. Se registra la varianza y la desviación estándar para ambos métodos y se calcula la reducción porcentual cuando se utiliza el Filtro de Kalman.

estándar muestran una mejora significativa en las mediciones del ángulo procesadas por el filtro de Kalman. Esta mejora se evidencia en una reducción en la varianza del ángulo calculado de más del 88.66 % y una disminución en la desviación estándar de un 66.332 %.

Figura 38

Comparación del ángulo yaw con y sin Filtro Kalman



Nota. Esta figura muestra la comparación del ángulo yaw con y sin el uso del Filtro Kalman.

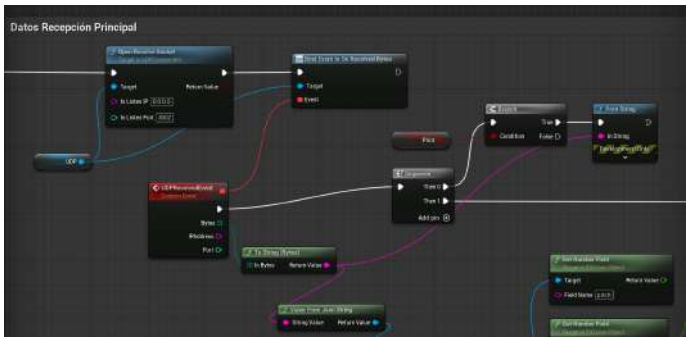
En la gráfica Figura 38, podemos ver la comparativa entre la medición del ángulo obtenido por el magnetómetro **yaw_mag** y la medición del ángulo después de aplicar el filtro Kalman **yaw_kalman**. Al analizar esta gráfica, se observa que la medición del ángulo filtrado presenta mayor estabilidad y menor ruido en comparación con la medición del ángulo yaw obtenido por el magnetómetro. La línea correspondiente al ángulo filtrado es más suave y sigue una trayectoria más consistente a lo largo del tiempo, lo que nos demuestra una reducción en la variabilidad y el ruido de la medición.

4.2. Pruebas de operación del prototipo

Durante las pruebas de operación del prototipo de simulador de conducción virtual en Unreal Engine, se llevaron a cabo diversas pruebas para evaluar el funcionamiento y la precisión del sistema.

Para verificar la recepción de los datos y si estas estaban correctos, se necesita literalmente ver la información que llegaba, el código en Unreal abre el puerto de recepción y que llegue la información que pueda llegar, pero no se conoce el valor que llegaba.

Figura 39. Código en Unreal para imprimir en pantalla el objeto JSON



Con esto, se imprime en pantalla en la parte superior izquierda de la pantalla el valor que se le ingrese a ese bloque, que en este caso era el JSON que recibía, cada dos segundos en pantalla se actualiza la impresión en la pantalla el valor que se recibe.

4.2.1. Pruebas de Funcionalidad de los Botones. Se comprobó que los 4 botones del simulador respondieron correctamente a las interacciones del usuario, cada botón activó las funciones específicas de acuerdo con su mapeo en Unreal Engine, a respuesta fue instantánea y sin retrasos notables.

4.2.2. Pruebas de la IMU y el Filtro de Kalman. Para este proyecto, se utilizó el sensor BNO055, que viene equipado con un algoritmo de fusión sensorial basado en el Filtro de Kalman Extendido. Este algoritmo provee una estimación precisa del ángulo de guiñada yaw, la cual es crucial para simular con exactitud la posición angular del volante en el simulador de conducción.

Es posible obtener una medida del ángulo de guiñada a través del magnetómetro, sin em-

bargo, esta medida solo es confiable si el dispositivo se encuentra en una orientación horizontal y su precisión tiende a ser inferior a la obtenida a través del algoritmo de fusión sensorial.

El algoritmo de fusión sensorial del BNO055 toma en cuenta las orientaciones de inclinación pitch y balanceo roll, lo que permite obtener una medición precisa del ángulo de guiñada, sin importar la orientación del dispositivo.

4.2.3. Pruebas de Transmisión y Recepción de Datos JSON. La ESP32 transmitió correctamente los datos del simulador en formato JSON a través de la conexión UDP. Los plugins UDPwrapper y SocketIOClient en Unreal Engine recibieron los datos de manera fiable y sin pérdidas significativas.

La ESP32 transmitió correctamente los datos del simulador en formato JSON a través de la conexión UDP. Los plugins UDPwrapper y SocketIOClient en Unreal Engine recibieron los datos de manera fiable y sin pérdidas significativas.

4.2.4. Pruebas de Interacción con el Simulador. Las acciones del usuario, detectadas a través de los botones y la IMU, afectaron adecuadamente la simulación de conducción en Unreal Engine. Los eventos y funciones correspondientes se activaron con precisión, y el vehículo virtual respondió de manera coherente a las acciones del usuario.

4.2.5. Pruebas de Compilación y Ejecución en Diferentes Plataformas. El simulador se compiló exitosamente tanto para dispositivos Android como para sistemas operativos Windows. Las pruebas en ambas plataformas mostraron un rendimiento estable y consistente, sin problemas de compatibilidad.

Figura 40. Verificación de transmisión y recepción del objeto JSON.

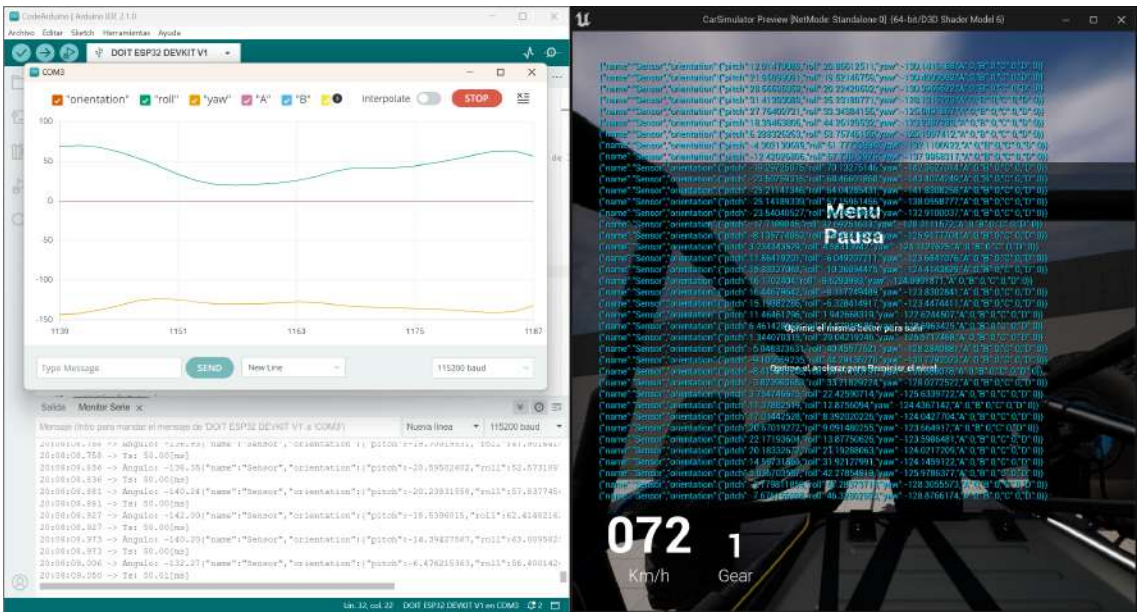
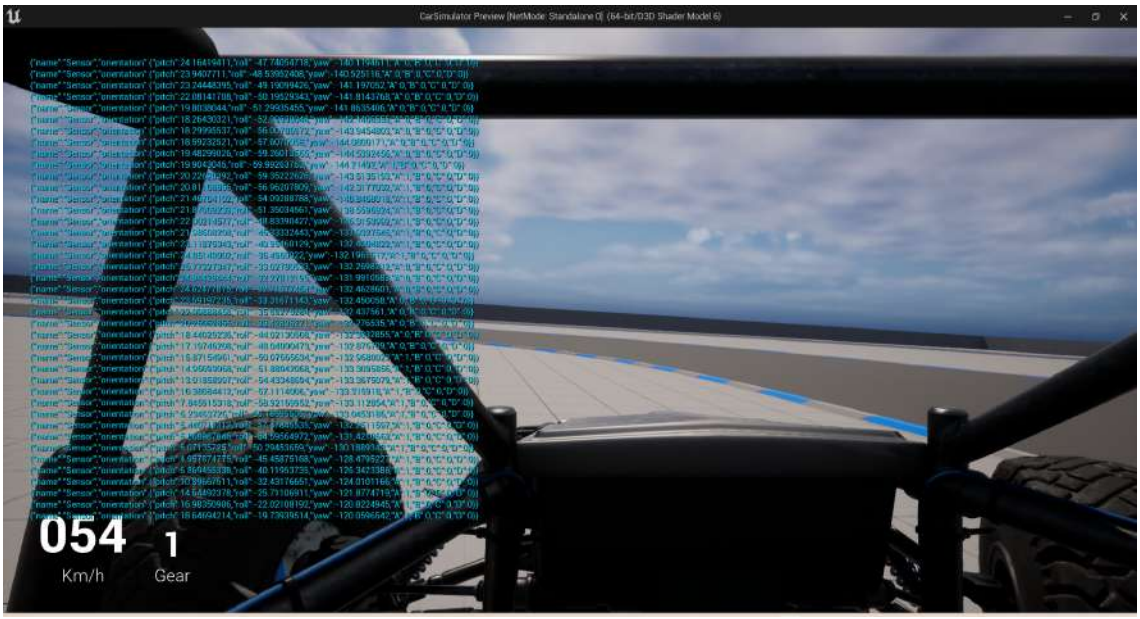


Figura 41. Interfaz del simulador imprimiendo el objeto JSON en consola.



5. Recomendaciones

- Para asegurar una base de datos funcional se propone la gestión de un servidor alojado en la nube para poder realizar consultas a la base de datos desde internet y poder enlazar esta a diferentes aplicativos. Además, se recomienda realizar el diseño relacional de la base de datos con las mediciones de electrodomésticos grupales.

- Con el propósito de mejorar los resultados de clasificación de todos los elementos se plantea un aumento en el número de muestras por dispositivo, incluyendo la base de datos de mediciones individuales completa. Es importante mencionar, que a medida que se aumente la cantidad de muestras, el tiempo que dura la extracción de características de las señales para organizar la matriz que se usa para el entrenamiento será mayor (actualmente con las muestras presentadas tiene una duración aproximada de 33 horas).

- Si las baterías están cargadas y se conecta la ESP32 al computador, es posible que se produzca una sobrecarga en las baterías. La sobrecarga puede dañar las baterías y, en casos extremos, incluso provocar fugas o incendios. Para evitar estos riesgos, es recomendable esperar hasta que las baterías estén descargadas o al menos en un nivel bajo de carga antes de conectar la ESP32 al computador. Además, es importante utilizar cables y conexiones en buen estado y seguir las especificaciones del fabricante para una correcta carga y uso de las

baterías. Siempre es conveniente tomar precauciones adicionales para garantizar la seguridad de los dispositivos y las personas involucradas.

6. Conclusiones

La implementación del Filtro de Kalman para estimar la orientación yaw utilizando datos del magnetómetro demostró ser una herramienta poderosa para aumentar la precisión y fiabilidad de las mediciones. Este método logró reducir la varianza causada por el ruido inherente de los sensores en más del 88 %. Además, se observó una disminución significativa en la desviación estándar del error, la cual se redujo en un 66 %. Esto indica que la dispersión de los errores de medición disminuyó en gran medida, mejorando aún más la precisión de las estimaciones.

El establecimiento de la comunicación entre la aplicación del simulador de automóvil en Unreal Engine y el mando de control virtual vía WiFi fue uno de los aspectos fundamentales del proyecto. Para lograr una interacción fluida y en tiempo real entre el usuario y el simulador de conducción, se implementó una conexión inalámbrica basada en WiFi utilizando la placa ESP32 y la transmisión de datos en formato JSON mediante el protocolo UDP (User Datagram Protocol).

Una vez que se estableció la conexión WiFi, se configuró la ESP32 para formatear los datos del control del usuario en un objeto JSON. Este objeto JSON contiene información sobre las acciones del usuario, como el estado de los botones y los datos de la IMU, la ESP32 utiliza el protocolo UDP para enviar los datos JSON al simulador de conducción en Unreal Engine.

En la aplicación del simulador de automóvil en Unreal Engine, se implementó una recepción adecuada de los datos enviados por la ESP32. Se utilizaron los plugins UDPwrapper y SocketIOClient para recibir y desconcatenar los datos JSON transmitidos por la ESP32. Una vez que los datos

JSON fueron recibidos y desconcatenados en Unreal Engine, se mapearon los valores específicos a eventos y funciones dentro del motor gráfico. Por ejemplo, en uno de los botones se utilizaron para controlar la aceleración del vehículo virtual, mientras que los datos de orientación influyeron en la dirección del vehículo.

Los datos transmitidos por la ESP32 a través de la conexión WiFi permitieron una simulación de conducción envolvente y altamente responsiva, lo que mejoró significativamente la experiencia del usuario y la sensación de inmersión en el entorno virtual. La conexión inalámbrica y la transmisión de datos en formato JSON utilizando UDP fueron fundamentales para lograr esta comunicación eficiente y en tiempo real entre ambos dispositivos.

El propósito de este prototipo fue mejorar la interactividad y realismo en la simulación de conducción, brindando a los usuarios una experiencia más inmersiva y auténtica. Al combinar tecnologías como la IMU, el filtro de Kalman y la conectividad con Unreal Engine 5, se logró una buena respuesta en la simulación de movimientos del vehículo, lo que permitió una experiencia de conducción más realista a la hora de usar, además el peso del prototipo de volante al ser ligero de 228g y ergonómico brindó comodidad durante su uso.

El uso de la IMU BNO055 ha sido fundamental y protagonista en este proyecto, proporcionando una detección de movimiento casi sin ruido y altamente precisa. Gracias a su integración con Unreal Engine, los movimientos del usuario se reflejan de manera realista en la interfaz de simulación de conducción. La combinación con el filtro de Kalman ha mejorado aún más la estabilidad de los datos.

La simulación de condiciones físicas realistas ofrece un valor añadido al prototipo de control vir-

tual, ya que permite a los usuarios experimentar escenarios difíciles de manejar en la vida real, sin correr riesgos reales. Esta combinación de tecnologías avanzadas y condiciones físicas simuladas abre nuevas posibilidades para aplicaciones en la formación de conductores, pruebas de vehículos y en la investigación de comportamiento de vehículos en situaciones extremas.

Referencias Bibliográficas

de Ingeniería, H. (s.f.). Duración teórica de la carga de una batería. Descargado de https://www.herramientasingenieria.com/onlinecalc/spa/duracion-baterias/duracion-baterias.html

del Valle Hernández, L. (2022). Resistencia pull up y pull down. *Programar Facíl*. Descargado de https://programarfacil.com/blog/arduino-blog/resistencia-pull-up-y-pull-down/

de Tecnologías Educativas y Formación del Profesorado, I. N. (2009). Normalización. *intef.es*.

Diaz, J. A. (2006). Lección 5: Acotaciones. *Universidad de Granada*.

DigiKey. (s.f.). Calculadora de ancho de trazas de pcb. *Resource*. Descargado de https://www.digikey.com/es/resources/conversion-calculators/conversion-calculator-pcb-trace-width

Josefa Katiuska Toala-Palma, J. M. Q.-L. M. I. S.-V., Jéssica Lourdes Arteaga-Mera. (2020). La realidad virtual como herramienta de innovación educativa. *EPISTEME KOINONIA*, *3*(5).

Juan Antonio Jiménez Tejada, J. A. L. V. (2008). Problemas de electrÓnica bÁsica.

Large. (2020). Especificaciones de la batería de litio 18650-mah y especificaciones importantes. *Large News*.

Lozé, S. (2020a). From defense to healthcare training: building vr on a common framework. *Spotlight - Unreal Engine*. Descargado de https://www.unrealengine.com/en-US/spotlights/from-defense-to-healthcare-training-building-vr-on-a-common

-framework

Lozé, S. (2020b). Vr medical simulation from precision os trains surgeons five times faster. *Spotlight - Unreal Engine*. Descargado de https://www.unrealengine.com/en-US/spotlights/vr-medical-simulation-from-precision-os-trains-surgeons-five-times-faster

Netto, C. M., Saldanha, C. C., y Dsouza, D. (2021). Room light intensity control with temperature monitoring system using arduino. En *2021 ieee international conference on distributed computing, vlsi, electrical circuits and robotics (discover)* (p. 160-163). doi: 10.1109/DISCOVER52564.2021.9663580

Pimentel, K. (2021). Meet the real-time bim app for designing and constructing buildings. *Spotlight - Unreal Engine*. Descargado de https://www.unrealengine.com/en-US/spotlights/meet-the-real-time-bim-app-for-designing-and-constructing-buildings

PRAJWAL SHENOY, A. G., y VARADHAN. (s.f.). Design and validation of an imu based full hand kinematic measurement system. *Resource*. Descargado de https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9870778

Sadiku, M. N. (2006). *Fundamentos de circuitos eléctricos* (3ra ed. ed.). McGraw-Hill.

Salmony, P. (2022). Errores de principiante en el diseño de pcb y cómo evitarlos. *Altium Resource*. Descargado de https://resources.altium.com/es/p/top-5-pcb-design-mistakes-and-how-fix-them

Townsend, K. (2015). Adafruit bno055 absolute orientation sensor - overview. *learn Adafruit*.

Apéndices

Apéndice A. Código principal ejecutado en la ESP32

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BNO055.h>
#include <utility/imumaths.h>
#include <math.h>
#include <WiFi.h>
#include "ArduinoJson.h"
#include "AsyncUDP.h"
#include "Streaming.h"

#define BNO055_DELAY_MS (100)
#define WIFI_NETWORK "yesud"
#define WIFI_PASSWORD "yeye2000"
#define WIFI_TIMEOUT_MS 20000
#define SERVER "192.168.1.121"
#define PORT 3002

// Declaraci n de variables
// Variables del filtro de Kalman
// Variables del filtro de Kalman
float dt = 0.05; // Intervalo de tiempo entre mediciones (segundos)
float A[2][2] = {{1, dt}, {0, 1}}; // Matriz de transici n de estado
float H[2][2] = {{1, 0}, {0, 1}}; // Matriz de observaci n
float Q[2][2] = {{0.001, 0}, {0, 0.001}}; // Matriz de ruido del proceso
float R[2][2] = {{0.09522, 0}, {0, 0.00429}}; // Matriz de ruido de la
    medici n
float P[2][2] = {{1, 0}, {0, 1}}; // Matriz de covarianza inicial
float x[2] = {0, 0}; // Estado inicial ( ngulo  y velocidad angular)

// Periodo de muestro
double t, t0 = 0;
const float Ts = 50000; //Periodo de muestreo en microsegundos

// Botones
const int botonA = 12;
const int botonB = 14;
const int botonC = 27;
const int botonD = 26;
int Ab = 0;
int Bb = 0;
int Cb = 0;
int Db = 0;


// Nivel Bateria
const int MAX_ANALOG_VAL = 4095;
const float MAX_BATTERY_VOLTAGE = 4.2; // El voltaje LiPoly m ximo de una
    bater a 3.7 es 4.2

//Sensor IMU BNO055
Adafruit_BNO055 sensor = Adafruit_BNO055();
float roll = 0; //  ngulo  de medici n de balanceo basado en manget metro
float pitch = 0; //  ngulo  de medici n de cabeceo basado en manget metro
float yaw = 0; //  ngulo  de medici n de gui ada basado en manget metro

//Json objeto
```

```
55 String json;
56
57 // Async UDPClient objeto
58 AsyncUDP udpClient;
59
60 // Variable boolena
61 bool isCal = false;
62 bool WifiReady = false;
63 bool I2CReady = false;
64 bool leidoBoton = false;
65 bool leidoIMU = false;
66
67 // Maquina de estados
68 enum State
69 {
70   Inicio,
71   Lectura,
72   Procesamiento,
73   Transmision
74 };
75
76 // Cambiador de estados
77 int conectado = 0;
78 int proceso = 0;
79
80 State currentState;
81
82 //Arranca Nayro Quintana
83 void setup() {
84   // Declaraci n de pines ESP32
85   // Entradas
86   pinMode(botonA,INPUT_PULLDOWN);
87   pinMode(botonB,INPUT_PULLDOWN);
88   pinMode(botonC,INPUT_PULLDOWN);
89   pinMode(botonD,INPUT_PULLDOWN);
90
91   //Reloj externo
92   sensor.setExtCrystalUse(true);
93
94   // Iniciar puertos seriales
95   Serial.begin(115200);
96
97   // Inicio sistema
98   currentState = Inicio;
99
100  // Inicio periodo de muestreo
101  t0 = micros();
102 }
103
104 void loop() {
105   t = micros();
106   //updateStateMachine();
107   if (t < t0)
108   {
109     t0 = 0;
110   }
111
112   if (t - t0 >= Ts)
113   {
114     updateStateMachine();
115     Serial.print("Ts: ");
116     Serial.print((t - t0)/1000);
117     Serial.println("[ms]");
118     t0 = t;
119   }
```

```
120
121 }
122
123 // Funciones
124
125 // Maquina de Estados
126
127 void updateStateMachine()
128 {
129   switch (currentState)
130   {
131     case Inicio: stateInicio(); break;
132     case Lectura: stateLectura(); break;
133     case Procesamiento: stateProcesamiento(); break;
134     case Transmision: stateTransmision(); break;
135   }
136 }
137
138 // Transiciones
139 void stateInicio()
140 {
141   if (conectado == 0)
142     changeState(State::Inicio);
143   if (conectado == 1)
144     changeState(State::Lectura);
145 }
146
147 void stateLectura()
148 {
149   if (conectado == 0)
150     changeState(State::Inicio);
151   if (proceso == 0)
152     changeState(State::Lectura);
153   if (proceso == 1)
154     changeState(State::Procesamiento);
155 }
156
157 void stateProcesamiento()
158 {
159   if (conectado == 0)
160     changeState(State::Inicio);
161   if (proceso == 0)
162     changeState(State::Lectura);
163   if (proceso == 1)
164     changeState(State::Procesamiento);
165   if (proceso == 2)
166     changeState(State::Transmision);
167 }
168
169 void stateTransmision()
170 {
171   if (conectado == 0)
172     changeState(State::Inicio);
173   if (proceso == 0)
174     changeState(State::Lectura);
175   if (proceso == 1)
176     changeState(State::Procesamiento);
177   if (proceso == 2)
178     changeState(State::Transmision);
179 }
180
181 void changeState(State newState)
182 {
183   currentState = newState;
```

```
184   switch (currentState)
185   {
186     case State::Inicio: outputInicio();    break;
187     case State::Lectura: outputLectura();    break;
188     case State::Procesamiento: outputProcesamiento();    break;
189     case State::Transmision: outputTransmision();    break;
190     default: break;
191   }
192 }
193
194 // Funciones estados
195 void outputInicio()
196 {
197   connectToWiFi ();
198   protocoI2C ();
199   delay(1000);
200   if (WifiReady == true && I2CReady == true)
201     conectado = 1;
202   Serial.println ("Conexi n");
203   Serial.println (conectado);
204 }
205
206 void outputLectura()
207 {
208   LecturaBotones();
209   lecturaBno055();
210  // bateriaRevision();
211   if (leidoBoton == true && leidoIMU == true)
212     proceso = 1;
213     leidoBoton = false;
214     leidoIMU = false;
215 }
216
217 void outputProcesamiento()
218 {
219   json = GetJSONString ("Sensor", pitch, roll, yaw, Ab, Bb, Cb, Db);
220   Serial.println(json);
221   proceso = 2;
222 }
223
224 void outputTransmision()
225 {
226   UDPSendData(json);
227   // delay(BNO055_DELAY_MS);
228   proceso = 0;
229 }
230
231 // Conectarse a WiFI
232 void connectToWiFi (){
233   Serial.print ("Conexi n a WiFI");
234   WiFi.mode (WIFI_STA);
235   WiFi.begin (WIFI_NETWORK, WIFI_PASSWORD);
236   unsigned long startAttemptTime = millis ();
237   while (WiFi.status () != WL_CONNECTED && millis() - startAttemptTime <
      WIFI_TIMEOUT_MS){
238     Serial.print (".");
239     delay(100);
240   }
241
242
243   // TODO: Tratamiento de errores
244   if (WiFi.status () != WL_CONNECTED){
245     Serial.print ("Fallido!");
246   } else {
```

```
247       Serial.print ("Conectado!");
248       Serial.print (WiFi.localIP());
249       WifiReady = true;
250       Serial.println ("Wifi= ");
251       Serial.println (WifiReady);
252     }
253   }
254
255   //
256   void protocoI2C (){
257     Serial.println ("I2C scaner. Scaneando ...");
258     Wire.begin();
259     byte i = 40;
260     Wire.beginTransmission (i);
261     if (Wire.endTransmission () == 0){
262       Serial.print ("Direcci n Encontrada : ");
263       Serial.print (i, DEC);
264       Serial.print (" (0x");
265       Serial.print (i, HEX);
266       Serial.println (")");
267     } // Posibles Repuestas
268     Serial.println ("Listo.");
269     Serial.println (" dispositivo(s).");
270     I2CReady = true;
271     Serial.println ("I2C = ");
272     Serial.println (I2CReady);
273     if(!sensor.begin()){
274       Serial.print("Ooops, no se ha detectado BNO055 ... Compruebe su cableado
          o I2C ADDR!");
275       while(1);
276     }
277   }
278
279   void LecturaBotones (){
280     Ab = digitalRead(botonA);
281     Bb = digitalRead(botonB);
282     Cb = digitalRead(botonC);
283     Db = digitalRead(botonD);
284     leidoBoton = true;
285   }
286
287
288   void lecturaBno055 (){
289     uint8_t system, gyro, accel, mg = 0;
290     sensor.getCalibration(&system, &gyro, &accel, &mg);
291
292     imu::Quaternion quat = sensor.getQuat();
293     quat.normalize ();
294     imu::Vector<3> euler = quat.toEuler();
295     yaw = euler.x() * 180/M_PI;
296     pitch = euler.y() * -180/M_PI;
297     roll = euler.z() * -180/M_PI;
298
299     // Calcular el  ngulo  de yaw a partir del campo magn tico
300     float mag_heading = atan2(mag.y(), mag.x());
301     float yaw_mag = mag_heading * 180 / M_PI;
302     if (yaw_mag < 0) {
303       yaw_mag += 360;
304     }
305
306
307     // Crear el vector de medici n Z
308     float Z[2] = {yaw_mag, gyro_mac.z()};
309
```

```
310   // Actualizar el filtro de Kalman
311   kalman_update(Z);
312
313   Serial.print("Angulo yaw filtrado");
314   Serial.println(x[0], 6);
315
316
317   //  Finaliza
318
319   imu::Vector<3> laccel = sensor.getVector(Adafruit_BNO055::VECTOR_LINEARACCEL
        );
320   //Serial  << system << "," << accel << "," << gyro << "," << mg << "," <<
        laccel.x() << "," << laccel.y() << "," << laccel.z() << ", Botones : A="
        << A << ", B=" << B << ", C=" << C << endl;
321
322   leidoIMU = true;
323 }
324
325 void bateriaRevision (){
326   int rawValue = analogRead(32);
327   // El Voltaje de referencia en ESP32 es 1.1V
328   float voltageLevel = (rawValue / 4095.0) * 2 * 1.1 * 3.3; // calcular el
        nivel de tensi n
329   float batteryFraction = voltageLevel / MAX_BATTERY_VOLTAGE;
330   Serial.println((String)"Sin procesar:" + rawValue + " Voltaje:" +
        voltageLevel + "V Porcentaje: " + (batteryFraction * 100) + "%");
331 }
332
333 String GetJSONString(String sensorName, float pitch, float roll, float yaw,
      int Ab, int Bb, int Cb,int Db)
334 {
335     DynamicJsonDocument doc(1024);
336     doc["name"] = sensorName;
337
338     DynamicJsonDocument orientation(768);
339     orientation["pitch"] = pitch;
340     orientation["roll"] = roll;
341     orientation["yaw"] = yaw;
342     orientation["A"] = Ab;
343     orientation["B"] = Bb;
344     orientation["C"] = Cb;
345     orientation["D"] = Db;
346     doc["orientation"] = orientation;
347     char docBuf[1024];
348     serializeJson(doc, docBuf);
349     return String(docBuf);
350 }
351
352 void UDPConnect()
353 {
354     IPAddress ipAddress = IPAddress();
355     ipAddress.fromString(SERVER);
356     udpClient.connect(ipAddress, PORT);
357     Serial.println ("Servidor conectado");
358 }
359
360 void UDPSendData(String message)
361 {
362     char charBuffer[1024];
363     message.toCharArray(charBuffer, 1024);
364     if (udpClient.connected()){
365       udpClient.broadcastTo(charBuffer, PORT);
366     } else {
367       Serial.println ("Servidor NO conectado");
```

```
368         UDPConnect();
369       }
370 }
371
372 // Funci n de multiplicaci n de matrices
373 void matmul(float A[][2], float B[][2], float C[][2], int m, int n, int p) {
374   float temp[m][p];
375   for (int i = 0; i < m; i++) {
376     for (int j = 0; j < p; j++) {
377       temp[i][j] = 0;
378       for (int k = 0; k < n; k++) {
379         temp[i][j] += A[i][k] * B[k][j];
380       }
381     }
382   }
383   for (int i = 0; i < m; i++) {
384     for (int j = 0; j < p; j++) {
385       C[i][j] = temp[i][j];
386     }
387   }
388 }
389 void kalman_update(float Z[2]) {
390   // Paso de predicci n
391   float x_pred[2] = {A[0][0] * x[0] + A[0][1] * x[1], A[1][0] * x[0] + A[1][1]
       * x[1]};
392   float P_pred[2][2];
393   float A_P[2][2];
394   matmul(A, P, A_P, 2, 2, 2);
395
396   // Calcular la traspuesta de A
397   float A_T[2][2] = {{A[0][0], A[1][0]}, {A[0][1], A[1][1]}};
398
399   // Calcular P_pred = A * P * A^T + Q
400   matmul(A_P, A_T, P_pred, 2, 2, 2);
401   P_pred[0][0] += Q[0][0]; P_pred[0][1] += Q[0][1];
402   P_pred[1][0] += Q[1][0]; P_pred[1][1] += Q[1][1];
403
404   // Paso de actualizaci n
405   float y[2] = {Z[0] - (H[0][0] * x_pred[0] + H[0][1] * x_pred[1]), Z[1] - (H
       [1][0] * x_pred[0] + H[1][1] * x_pred[1])};
406   float S[2][2];
407   matmul(H, P_pred, S, 2, 2, 2);
408   S[0][0] += R[0][0]; S[0][1] += R[0][1];
409   S[1][0] += R[1][0]; S[1][1] += R[1][1];
410
411   float S_inv[2][2];
412   float det_S = S[0][0] * S[1][1] - S[0][1] * S[1][0];
413   S_inv[0][0] = S[1][1] / det_S;
414   S_inv[0][1] = -S[0][1] / det_S;
415   S_inv[1][0]= -S[1][0] / det_S;
416   S_inv[1][1] = S[0][0] / det_S;
417
418   float K[2][2];
419   matmul(P_pred, H, K, 2, 2, 2);
420   matmul(K, S_inv, K, 2, 2, 2);
421
422   // Actualizar el estado y la matriz de covarianza
423   x[0] = x_pred[0] + K[0][0] * y[0] + K[0][1] * y[1];
424   x[1] = x_pred[1] + K[1][0] * y[0] + K[1][1] * y[1];
425
426   float I[2][2] = {{1, 0}, {0, 1}};
427   float KH[2][2] = {{I[0][0] - K[0][0] * H[0][0] - K[0][1] * H[1][0], I[0][1]
       - K[0][0] * H[0][1] - K[0][1] * H[1][1]},
428                     {I[1][0] - K[1][0] * H[0][0] - K[1][1] * H[1][0], I[1][1]
```

```
    - K[1][0] * H[0][1] - K[1][1] * H[1][1]}};
429
430   matmul(KH, P_pred, P, 2, 2, 2);
431 }
```

Apéndice B. Links y enlaces web

Link de Github https://github.com/YesidElYesud/PCVCIK-2022-2 Ahi esta todo el proyecto, archivos, imagenes y ejecutables.

Link del video https://www.youtube.com/watch?v=WTq3cihyyJE

Ejecutable del juego para Windows y Android https://github.com/YesidElYesud/PCVCIK-2022-2/releases/tag/ue5

Link del libro https://drive.google.com/file/d/14DhTeFGc9uvb3RIyDJdlWWCws_8kGQcd/view?usp=sharing

Link del Site de Google https://sites.google.com/e3t.uis.edu.co/pcvcik222/inicio

Lin de Presentación https://docs.google.com/presentation/d/1Fb_8u1o53WHOtNPu_97c_NpZpVyRNc3_VSmvXCB9BRQ/edit?usp=sharing