

UNIVERSIDAD AUTÓNOMA DE CIUDAD JUÁREZ

EXTENSIÓN CIUDAD UNIVERSITARIA

DEPARTAMENTO DE INGENIERÍA INDUSTRIAL Y MANUFACTURA

PLATAFORMA DE ADQUISICIÓN Y PUBLICACIÓN DE DATOS

Redes Industriales

Alumnos:

Jeshua Tecalco Maldonado 215677
José Jaime Estrada Muñoz 208170
Jorge Ivan Ruiz Espinoza 201621
Irán Saúl Torres Alvarado 192836
Kevin Ramos Rivera 180070

Equipo 2

Docente:

Dr. Francesco José García Luna

21 de noviembre de 2025

Índice

1. Introducción.	4
2. Objetivo	4
2.1. Objetivos Particulares	4
2.2. Materiales y hardware	4
3. Desarrollo	6
3.1. Diseño	6
3.2. Cálculo del centro de masa	7
3.3. PCB	10
3.3.1. Esquemático	11
3.4. Programación	13
4. Resultados	13
4.1. Diseño final	13
4.2. PCB terminada	14
4.3. Programación y comunicación	14
5. Conclusiones	16
6. Anexos	17
6.1. PCB	17
6.1.1. Componentes	22
6.2. ESP32 MQTT.	26
6.3. Raspberry MQTT -> ROS2.	28
6.4. Gráficas de los resultados	30

Índice de figuras

1. Centro de masa y cálculo euclíadiano.	9
2. Diagrama esquemático completo del PCB mostrando las interconexiones entre el ESP32, Raspberry Pi y sensor MPU-9250.	12
3. Vistas de la esfera física con sus diferentes partes.	13
4. Centro de masa en SolidWorks	15
6. Diseño y características físicas de la PCB.	18
7. Diseño de la Esfera (parte inferior).	19
8. Diseño de la base	20
9. Diseño de las Partes de la estructura	20
10. Base	21
11. Diseño y dimensiones del MPU-9250	22
12. Diseño y dimensiones de la fuente de voltaje.	23
13. Diseño y dimensiones da la PCB con el ESP32 y el Raspberry Pi Zero 2W. .	24

14. Diseño de la esfera completa.	25
15. Gráfica de la Aceleración Lineal en X	30
16. Gráfica de la Aceleración Lineal en Y	30
17. Gráfica de la Aceleración Lineal en Z	31
18. Gráfica de la Velocidad Angular en X	31
19. Gráfica de la Velocidad Angular en Y	32
20. Gráfica de la Velocidad Angular en Z	32
21. Gráfica de la Rotación Pitch	33
22. Gráfica de la Rotación Roll	33
23. Gráfica de la Rotación Yaw	34
24. Gráfica de la Rotación X	34
25. Gráfica de la Rotación Y	35
26. Gráfica de la Rotación Z	35
27. Gráfica de la Rotación W	36

1. Introducción.

Este documento presenta el diseño de una plataforma de adquisición de datos, en donde tiene como objetivo el ser capaz de detectar y representar su posición y su orientación en tiempo real, por lo que mediante el uso de ROS 2 y microcontroladores, tal como la Raspberry Pi Zero 2 W que se utilizará como la unidad principal de procesamiento. La estructura se fabricará y se modelará a través del software Solidworks para la fabricación y su posterior montaje de componentes que se usarán en el sistema. De forma en que los sensores iniciales integrados serán quienes captarán los datos de movimiento, en donde, serán procesados y visualizados en ROS 2 a través de una interfaz gráfica interactiva.

A lo largo del desarrollo se consideraron elementos críticos del diseño en los que se incluyen la distribución del peso, la ubicación del centro de masa, la fabricación de la PCB y la interconexión de los módulos electrónicos, con el fin de asegurar una estructura resistente, junto a un sistema equilibrado en términos de procesamiento y ensamblaje. Por lo tanto la plataforma propuesta no solo cumple los requisitos funcionales del sistema, sino que también ofrece una base sólida para una mayor iteración.

2. Objetivo

Crear una plataforma capaz de medir y mostrar su posición y orientación usando sensores, una Raspberry Pi y ROS 2, asegurando que la estructura y los componentes estén bien organizados para que el sistema funcione de manera estable y confiable

2.1. Objetivos Particulares

- Diseñar la estructura física de la plataforma de adquisición de datos, es decir, la carcasa o soporte donde irán montados la Raspberry Pi, la IMU y los demás componentes.
- Armar el circuito electrónico de la plataforma, conectando la Raspberry Pi, la IMU y los módulos necesarios para su funcionamiento.
- Programar un IMU para la adquisición y publicación de datos .
- Verificar los resultados obtenidos.

2.2. Materiales y hardware

Los materiales requeridos para el funcionamiento del proyecto son los siguientes:

- **Raspberry Pi Zero 2 W:** Es el microcomputador principal que es encargado del procesamiento de los datos y tambien de la ejecución del sistema operativo gracias a su cpu arm Cortex-A53 de cuatro núcleos su conectividad inalámbrica tambien permite correr ROS 2 y manejar tareas como la comunicación con sensores publicación de tópicos y visualización de datos en herramientas, tambien su bajo consumo energético la hace adecuada para sistemas portátiles.

- **IMU (Unidad de Medición Inercial)** Sensor que combina acelerómetro y giroscopio tambien en algunos modelos, magnetómetro estos dispositivos permiten medir y mandar informacion :
 - Aceleración lineal en x, y, z.
 - Velocidad angular en x, y, z.
 - Campo magnético.
- **Placa PCB diseñada para el proyecto:** integración ordenada y segura de los componentes Permite la distribuir la energía y ayuda reducir ruido eléctrico en señales y asegurar conexiones estables para la IMU, la Raspberry Pi.
- **Batería LiPo** Fuente de energía portátil, se utiliza por la capacidad de entregar de corriente estable en sistemas embebidos esto nos ayuda manteniendo el funcionamiento continuo de la Raspberry Pi y la IMU
- **Módulo regulador de voltaje** componente que se encarga de estabilizar el volatje y ayuda a regular su capacidad de energía de un sistema mantenido una función continua de la Raspberry Pi y la IMU.
- **Cableado y conectores** garantiza comunicación eléctrica dando una adecuado funcionamiento del sistema
- **Estructura física de la plataforma de adquisición de datos** diseñada en SolidWorks, donde se montarán los componentes.
- **Herramientas de ensamble** como soldador, cautín, tornillos, soportes y elementos de montaje.

Parámetro	Valor
A	1
B	2

Tabla 1: Especificaciones técnicas de la Raspberry Pi Zero 2W

- **MPU-9250:** Este componente se encarga de leer los 9 grados de libertad (giroscopio, acelerómetro y magnetometro).

Parámetro	Valor
A	1
B	2

Tabla 2: Especificaciones técnicas del MPU-9250.

- **ESP32:** Este componente se encarga de leer los datos obtenidos del MPU-9250.

Parámetro	Valor
A	1
B	2

Tabla 3: Especificaciones técnicas del ESP-32

- **Fuente de voltaje de 5V:** Esta fuente será la que proporcione energía para todos los componentes (MPU-9250, ESP32 y Raspberry Pi Zero 2W).

3. Desarrollo

El proyecto consiste en el diseño de una plataforma inteligente que puede adquirir y analizar en tiempo real su posición y orientación dentro del entorno ROS 2. La estructura mecánica se modelará en SolidWorks como una carcasa esférica en 3D en donde integrará los componentes electrónicos. Y por último el sistema empleará una Raspberry Pi como unidad principal y el ESP32 para la adquisición de datos provenientes del sensor inercial MPU-9250 que dispone de acelerómetro, giroscopio y magnetómetro, por lo que permite la visualización y control de movimiento en tiempo real.

En el apartado electrónico, el diseño de la PCB permitirá tener una mejor organización en las conexiones, al igual que permite la eliminación de cables sueltos dentro de la esfera, en la cual fue validado durante la construcción, esto es debido a que las pistas del circuito si cumplieron con los requerimientos de sección, y gracias a esto el ensamblaje tuvo un resultado más limpio y seguro. Pero aún con los resultados satisfactorios obtenidos hasta el momento, todavía se pueden identificar varios aspectos a mejorar, siendo uno de ellos la verificación precisa usando el peso de los elementos.

3.1. Diseño

Para el diseño de la estructura se tomaron en cuenta las medidas de los componentes que se utilizaran, con el fin de realizarlo de un tamaño lo suficientemente grande para ensamblarlos en la esfera. La esfera tiene una medida de 20 cm de diámetro, la cual está dividida en 2 partes. Tiene cortes en la superficie para poder maniobrar mejor los componentes que están dentro de la esfera sin la necesidad de abrirse. Además de tener una vista total al interior, incluso con la esfera cerrada. En la mitad de la esfera se agregaron 4 roscas que permite el poder atornillarse, cada una de las roscas, cuenta con la medida de M4x0.7x12mm. También se agrego una base en el centro para sujetar todos los componentes, en donde se realizó un corte para el MPU-9250 con el fin de que quedara justo en el centro de masa de la esfera.

3.2. Cálculo del centro de masa

Para determinar la posición del centro de masa del sistema se utiliza el modelo CAD desarrollado en SolidWorks. Sin embargo, para obtener un valor analítico es necesario conocer las masas reales de cada componente y su ubicación a lo largo del eje Z , como se muestra en la Figura ???. El centro de masa total se calcula mediante la ecuación (1):

$$Z_{CM} = \frac{\sum m_i z_i}{\sum m_i} \quad (1)$$

Sustituyendo las masas y posiciones individuales de los componentes, expresadas en milímetros:

$$Z_{CM} = \frac{(98)(+Z_{sup}) + (40)(+Z_{ext}) + (91)(0) + (98)(-Z_{inf}) + (160)(-Z_{bat})}{98 + 40 + 91 + 98 + 160} \quad (2)$$

Finalmente, la expresión se simplifica como se muestra en la ecuación (3):

$$Z_{CM} = \frac{98Z_{sup} + 40Z_{ext} - 98Z_{inf} - 160Z_{bat}}{487} \quad (3)$$

Nomenclatura

- m_i : masa del componente i (g).
- z_i : posición del componente i sobre el eje Z (mm).
- Z_{sup} : posición del componente superior.
- Z_{ext} : posición del componente externo o lateral.
- Z_{inf} : posición del componente inferior.
- Z_{bat} : posición de la batería dentro del sistema.
- Z_{CM} : centro de masa total del sistema.

Cálculo de las distancias euclidianas

La distancia euclíadiana entre dos puntos $A(x_1, y_1, z_1)$ y $B(x_2, y_2, z_2)$ se define como:

$$D_{AB} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \quad (4)$$

Dado que los componentes se encuentran distribuidos únicamente sobre el eje Z , la expresión se simplifica a:

$$D_{AB} = |z_2 - z_1| \quad (5)$$

A continuación se muestran las distancias entre los componentes principales del sistema, expresadas en milímetros (mm):

$$D_{sup-centro} = |Z_{sup} - 0| = |50 - 0| = 50 \text{ mm} \quad (6)$$

$$D_{ext-centro} = |Z_{ext} - 0| = |40 - 0| = 40 \text{ mm} \quad (7)$$

$$D_{inf-centro} = |Z_{inf} - 0| = |50 - 0| = 50 \text{ mm} \quad (8)$$

$$D_{bat-centro} = |-Z_{bat} - 0| = |-60 - 0| = 60 \text{ mm} \quad (9)$$

$$D_{sup-bat} = |Z_{sup} - (-Z_{bat})| = |50 - (-60)| = 110 \text{ mm} \quad (10)$$

$$D_{ext-bat} = |Z_{ext} - (-Z_{bat})| = |40 - (-60)| = 100 \text{ mm} \quad (11)$$

Por lo tanto, las distancias euclidianas entre los componentes del sistema quedan resumidas en la siguiente tabla:

Tabla 4: Distancias euclidianas entre los componentes del sistema.

Distancia	Fórmula	Resultado (mm)
$D_{sup-centro}$	$ Z_{sup} - 0 $	50
$D_{ext-centro}$	$ Z_{ext} - 0 $	40
$D_{inf-centro}$	$ Z_{inf} - 0 $	50
$D_{bat-centro}$	$ -Z_{bat} - 0 $	60
$D_{sup-bat}$	$ Z_{sup} - (-Z_{bat}) $	110
$D_{ext-bat}$	$ Z_{ext} - (-Z_{bat}) $	100

Cálculo Euclíadiano del Centro de Masa

El centro de masa del ensamblaje se encuentra en las coordenadas proporcionadas por SolidWorks:

$$X = -68.56 \text{ mm}, \quad Y = 3.20 \text{ mm}, \quad Z = -46.99 \text{ mm}$$

Para obtener la **distancia euclíadiana** del centro de masa respecto al origen, se utiliza la siguiente expresión:

$$d = \sqrt{x^2 + y^2 + z^2}$$

Sustituyendo los valores del centro de masa:

$$d = \sqrt{(-68.56)^2 + (3.20)^2 + (-46.99)^2}$$

Cálculo de cada término:

$$\begin{aligned} (-68.56)^2 &= 4690.6 \\ (3.20)^2 &= 10.24 \\ (-46.99)^2 &= 2200.5 \end{aligned}$$

Suma de los términos:

$$4690.6 + 10.24 + 2200.5 = 6901.34$$

Finalmente, se obtiene la distancia euclíadiana:

$$d = \sqrt{6901.34} \approx 83.10 \text{ mm}$$

Resultado final:

$$d \approx 83.10 \text{ mm}$$

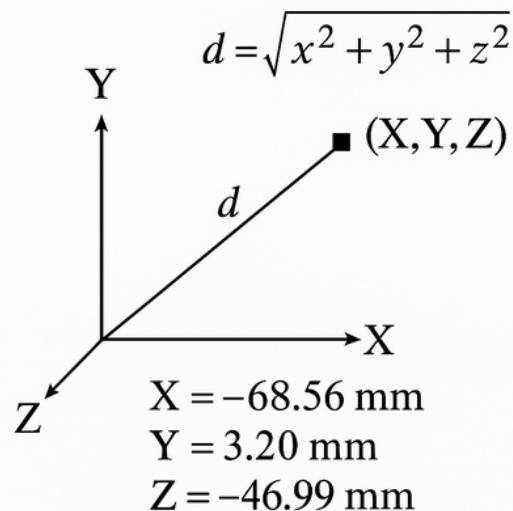


Figura 1: Centro de masa y cálculo euclíadiano.

3.3. PCB

Para la construcción del PCB se utilizó el software Fritzing, diseñando un circuito impreso de doble capa cuyas pistas se distribuyen en la capa superior (color amarillo) e inferior (color naranja), estas pistas tienen el tamaño estándar (24 milipulgadas). Las dimensiones de la placa son 7.25 cm de ancho por 6.55 cm de alto, lo que permite tener un diseño compacto de los componentes y a su vez manteniendo una distribución optimizada en las señales. Esta configuración de doble capa fue esencial para acomodar la complejidad de las interconexiones requeridas por el sistema, permitiendo una organización más eficiente del espacio disponible dentro de la esfera.

El diseño tiene conexiones específicas para el ESP32 y la Raspberry Pi Zero W, en donde las pistas de potencia tienen una mayor anchura (32 milipulgadas) con el fin de manejar adecuadamente la corriente requerida por el sistema. También se incluyeron orificios de sujeción estratégicamente ubicados para un montaje mecánico robusto dentro de la esfera, asegurando que los componentes permanezcan fijos durante el movimiento de la esfera. Al igual que se implementó un plano de tierra continuo que mejora la estabilidad eléctrica y reduce la susceptibilidad a interferencias electromagnéticas, lo cual es crucial para el correcto funcionamiento en un entorno dinámico.

La disposición de los componentes fue cuidadosamente planificada para minimizar la longitud de las trayectorias críticas y reducir las posibles interferencias entre señales digitales y analógicas. Esto es importante, ya que permite garantizar la integridad de las lecturas del sensor MPU-9250, que requiere un entorno eléctricamente estable para proporcionar mediciones precisas. Al igual que se priorizó la ubicación de componentes sensibles lejos de las fuentes de ruido y se implementaron técnicas de filtrado en las líneas de alimentación para asegurar lecturas consistentes del sensor inercial.

Desde el punto de vista de la fabricación, se consideraron estándares industriales para el espaciamiento entre pistas y el tamaño de los pads, lo que facilita el proceso de fabricación y ensamblaje. Todos los componentes utilizados están disponibles en formatos estándar, lo que simplifica la adquisición y reduce costos. La orientación de los componentes fue optimizada para permitir un ensamblaje eficiente, tanto manual como automatizado, considerando la accesibilidad para posibles reparaciones o modificaciones futuras.

Como se observa en las imágenes, el diseño final logra integrar todos los componentes esenciales en un espacio reducido, manteniendo al mismo tiempo una organización que facilita el ensamblaje y las posibles reparaciones. Los orificios de montaje periféricos permiten fijar firmemente la placa a la base interna de la esfera, mientras que la distribución de componentes evita concentraciones de calor y asegura una adecuada disipación térmica durante el funcionamiento prolongado.

3.3.1. Esquemático

En este apartado se revela la arquitectura de interconexiones del sistema, donde destaca la comunicación I2C entre el ESP-32 y los sensores, así como las líneas de alimentación diferenciadas para 3.3V y 5V según los requerimientos de cada componente. También se aprecian los circuitos de reinicio y arranque del ESP32, esenciales para la programación y reinicio del microcontrolador, junto con pines de expansión que permiten futuras mejoras o la conexión de sensores adicionales sin necesidad de modificar el diseño base de la placa.

Por otra parte, se definieron rutas de señal con longitudes moderadas para evitar retardos innecesarios y mantener la sincronización entre los distintos módulos. Esto fue especialmente importante en las líneas que manejan comunicación digital, donde un trazo excesivamente largo o mal referenciado podría introducir pequeñas variaciones eléctricas capaces de afectar la lectura del sensor o la estabilidad del microcontrolador. Asimismo, se procuró que las pistas de alimentación mantuvieran recorridos directos y sin bifurcaciones complejas, asegurando que cada dispositivo recibiera un voltaje estable incluso bajo condiciones de carga variable durante el funcionamiento del sistema.

Finalmente, se incorporaron puntos de prueba en ubicaciones estratégicas del esquemático para facilitar las etapas de diagnóstico y calibración. Estos puntos permiten medir tensiones clave en la placa, verificar continuidad en líneas críticas y confirmar el correcto funcionamiento del bus de datos sin necesidad de desmontar componentes. Esta previsión resulta especialmente útil en proyectos que pasan por múltiples iteraciones de diseño, ya que simplifica la detección de fallas y brinda mayor flexibilidad al momento de ajustar parámetros o reemplazar módulos individuales sin comprometer el resto del circuito.

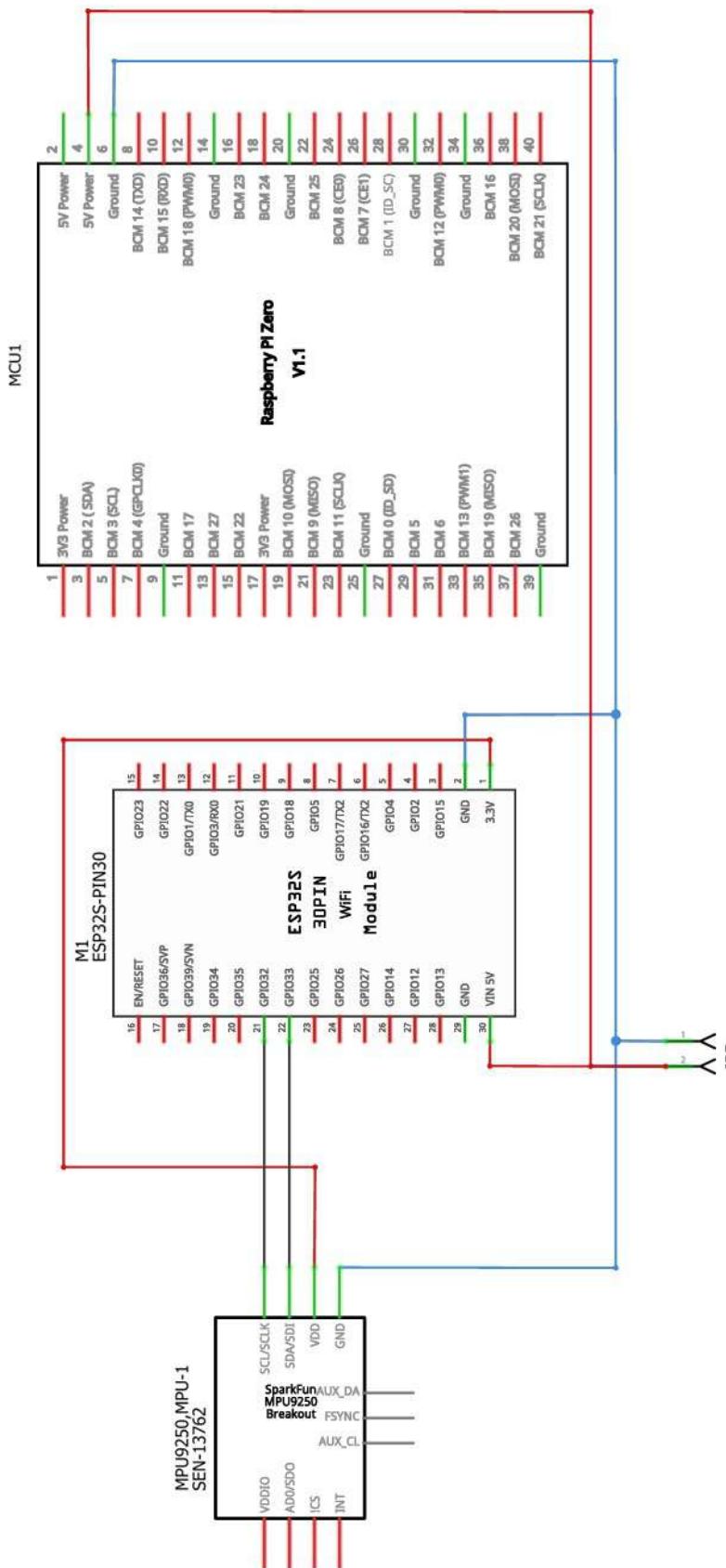


Figura 2: Diagrama esquemático completo del PCB mostrando las interconexiones entre el ESP32, Raspberry Pi y sensor MPU-9250.¹²

3.4. Programación

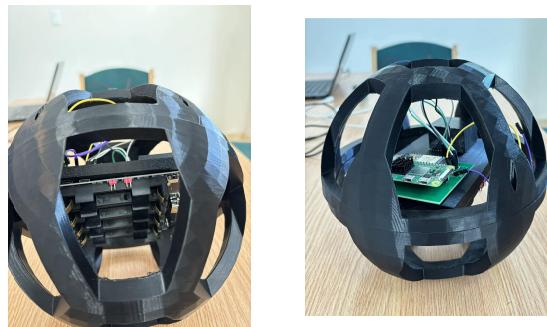
En la programación se dividen 3 secciones, la primera se enfoca en obtener los datos del MPU-9250 a través de un microcontrolador ESP32 por medio de I2C y que este procese los datos en crudo para poder mandarlos al Raspberry por medio de un nodo MQTT el cual estará en la ip local del Raspberry para que así este se suscriba al topico de mqtt, dando paso a la segunda sección la cual funcionara por medio del mismo programa para ros2 el cual estará en un entorno de Ubuntu server 24.04.03, el cual se encarga de qué la Raspberry se conecte al nodo de MQTT generado por el ESP32 y procese los datos para poder mandarlos a través de un nodo de ros2, lo cual finalmente abrirá paso a la tercera sección la cual funcionara a través de las herramientas de ros 2 mas en específico el filtro de madgwick

4. Resultados

La estructura física diseñada en SolidWorks y fabricado en impresión 3D. Al final, como se puede apreciar, es ligera, resistente y también permite colocar los módulos electrónicos ya mencionados. Cumpliendo así los resultados esperados del diseño modelado.

4.1. Diseño final

El diseño final de la plataforma logró integrarse correctamente dentro de la estructura esférica impresa en 3D, permitiendo validar que las dimensiones modeladas en SolidWorks fueron precisas para alojar cada uno de los componentes electrónicos. La distribución interna facilitó el montaje de la Raspberry Pi, el ESP32, la batería y la PCB sin generar tensiones mecánicas ni interferencias entre ellos. Además, la apertura y cierre de la esfera se realizaron de manera estable gracias al sistema de roscas M4, lo que permitió acceder fácilmente al interior para pruebas y montaje. La estructura resultó ligera, rígida y con buena estabilidad al posicionarse sobre diferentes superficies, confirmando que el centro de masa ajustado contribuye a un comportamiento más equilibrado del sistema durante su operación.



(a) Esfera física vista 1 (b) Esfera física vista 2

Figura 3: Vistas de la esfera física con sus diferentes partes.

4.2. PCB terminada

La PCB fabricada cumplió con los requerimientos eléctricos establecidos en el diseño, confirmándose que las pistas de alimentación, comunicación y señal mantienen una distribución adecuada para el entorno embebido dentro de la esfera. Durante las pruebas iniciales, tanto el ESP32 como la Raspberry Pi lograron alimentarse correctamente desde la fuente de 5V, sin presentar caídas de tensión. La comunicación I2C entre el MPU-9250 y el ESP32 se mantuvo estable aun cuando la PCB fue sometida a movimiento, asimismo, el tamaño compacto de la tarjeta permitió un ensamblaje ordenado, evitando el uso de cableado excesivo y mejorando la fiabilidad del sistema.

4.3. Programación y comunicación

En cuanto a la programación y la comunicación, la comunicación entre el MPU-9250 hacia el ESP32 se llevó a cabo sin pérdidas, permitiendo transmitir de manera continua los valores de aceleración, giro y magnetometría. El nodo MQTT configurado en la Raspberry Pi Zero 2W logró recibir los mensajes publicados por el ESP32 siendo estable dentro de la red local. La integración con ROS2 permitió visualizar las lecturas en un tópico dedicado y aplicar el filtro de Madgwick, verificando que el procesamiento en la Raspberry era suficiente para generar una orientación fluida y coherente con el movimiento físico de la esfera. Estos resultados muestran la arquitectura de comunicación seleccionada es útil para aplicaciones de adquisición de datos iniciales.

Centro de masa: Se muestra justo en el centro, en donde ubicamos el MPU-9250. Aunque colocando los componentes con su peso exacto el centro de masa se mueve ligeramente del centro, por ello se coloco un contrapeso para ajustar nuevamente el centro del masa.

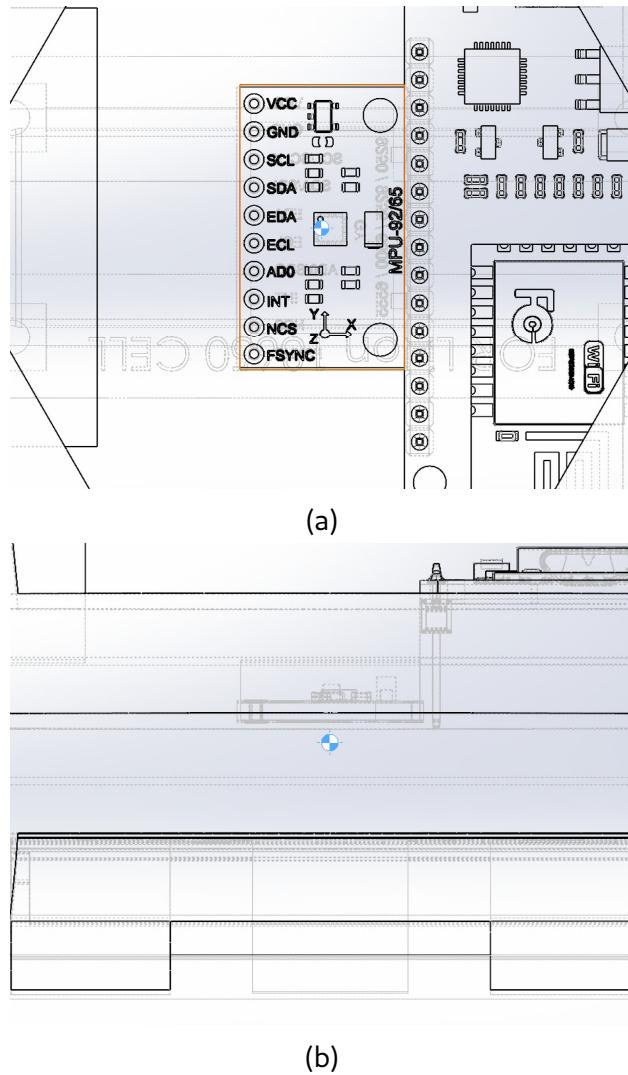


Figura 4: Centro de masa en SolidWorks

La etapa de simulación en fritzing indica que el diseño cumple con los requisitos de distribución de potencia y manejo de señales. Las pistas de alimentación tienen el ancho adecuado para los niveles de corriente esperados, y las longitudes de las trayectorias críticas han sido minimizadas para preservar la integridad de las señales de comunicación I2C entre el ESP32 y los periféricos. Este modelo nos permitirá seguir mejorando el diseño de la implementación.

5. Conclusiones

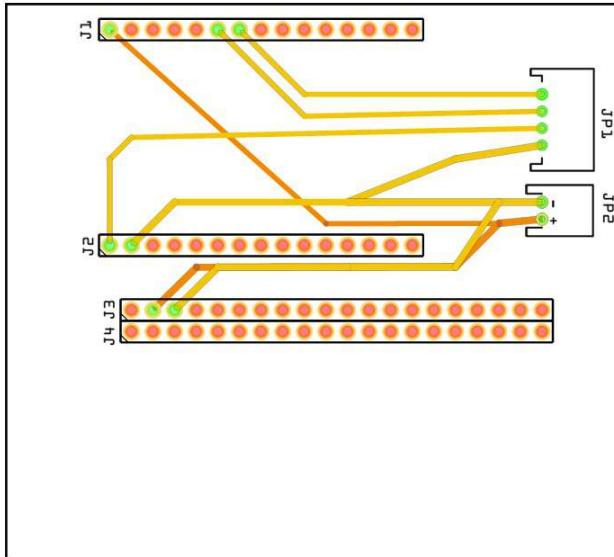
El proyecto, se ha obtenido un prototipo que permite a la plataforma de adquisición de datos comprobar la Raspberry Pi Zero 2W y el microcontrolador ESP32, logrando una comunicación efectiva mediante el protocolo MQTT y su posterior integración con ROS2 para el procesamiento y visualización de datos.

El modelado tridimensional en SolidWorks facilitó la correcta distribución de los componentes internos, garantizando tanto la estabilidad estructural como el posicionamiento adecuado del centro de masa, lo que resulta esencial para obtener lecturas confiables del sensor inercial MPU-9250. Asimismo, el diseño del PCB en Fritzing demostró cumplir con los requisitos electrónicos necesarios para su producción, permitiendo un montaje compacto y seguro dentro de la estructura.

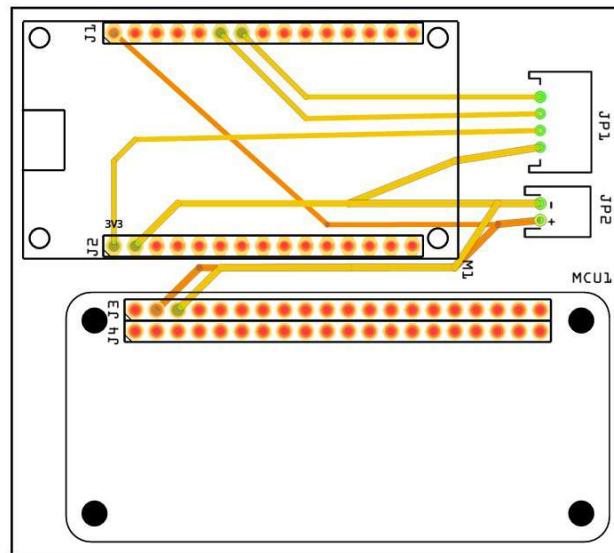
Aunque los resultados obtenidos son satisfactorios, se identifican áreas de mejora que serán abordadas en las siguientes fases, como la verificación experimental del peso de cada componente para ajustar con mayor precisión el centro de masa, la optimización de la transmisión de datos en tiempo real y la validación de las mediciones iniciales bajo diferentes condiciones dinámicas. En general, el progreso alcanzado sienta una base sólida para avanzar hacia la etapa de calibración, pruebas funcionales y optimización del sistema completo.

6. Anexos

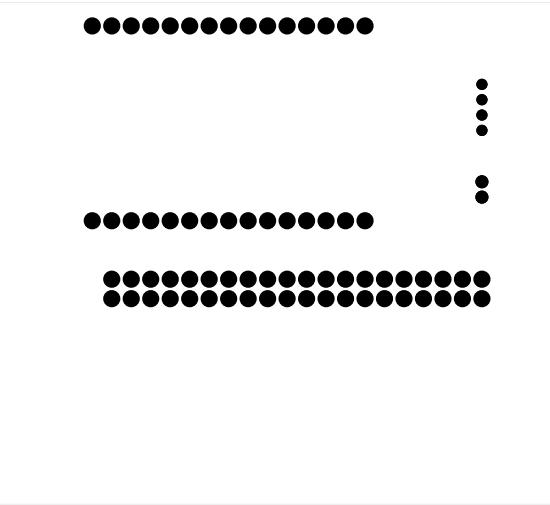
6.1. PCB



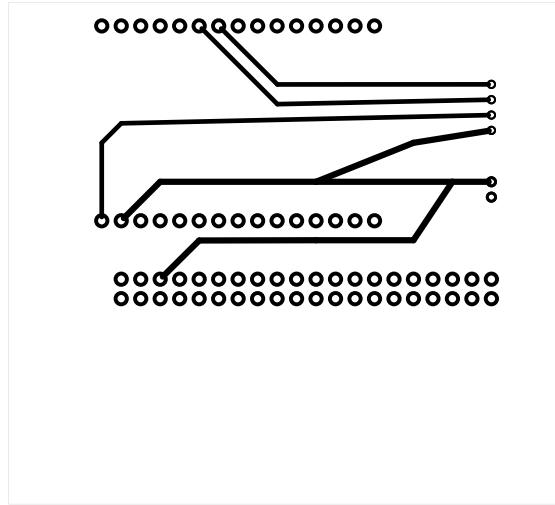
(a) PCB sin componentes insertados.



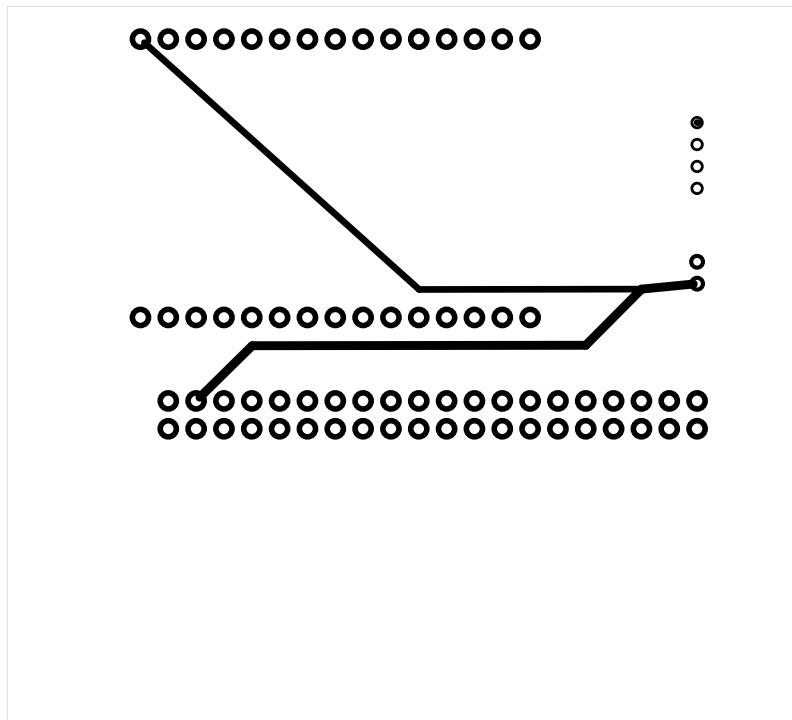
(b) PCB con ESP32 y Raspberry Pi Zero W.



(a) Orificios de conexión

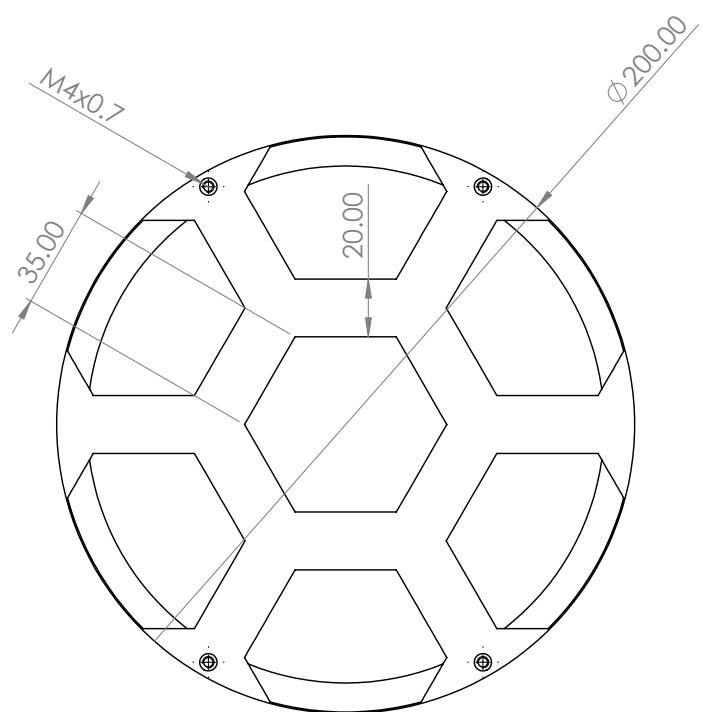


(b) Conexión en la parte superior



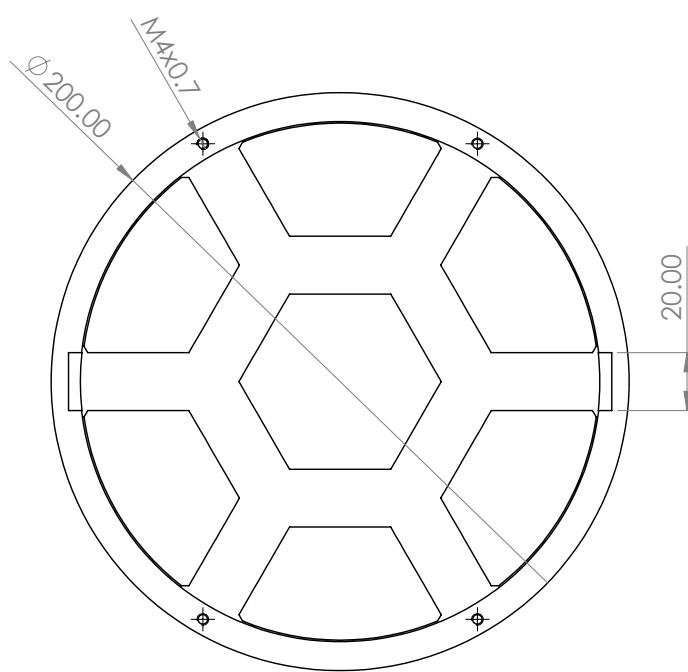
(c) Conexión en la parte inferior

Figura 6: Diseño y características físicas de la PCB.



Producto SOLIDWORKS Educational. Solo para uso en la enseñanza.

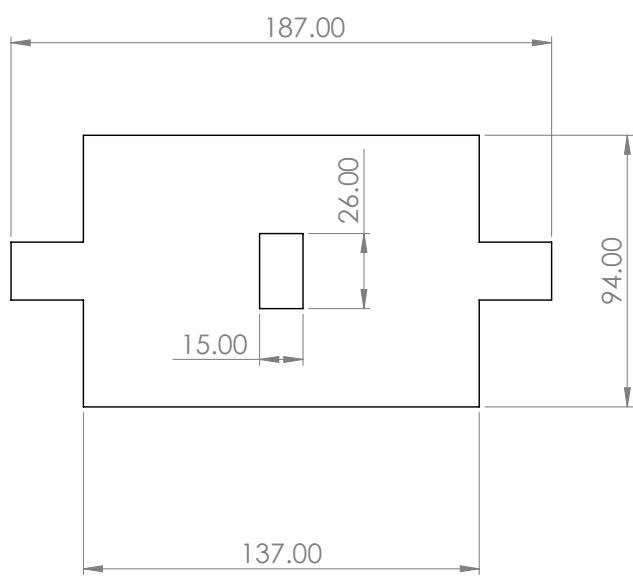
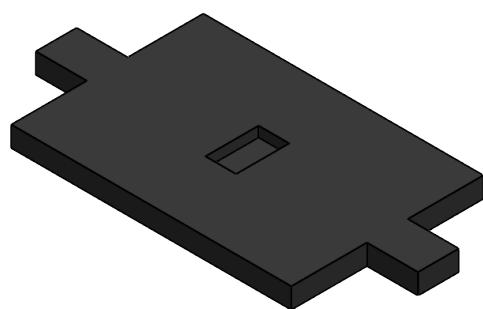
Figura 7: Diseño de la Esfera (parte inferior).



Producto SOLIDWORKS Educational. Solo para uso en la enseñanza.

Figura 8: Diseño de la base

Figura 9: Diseño de las Partes de la estructura

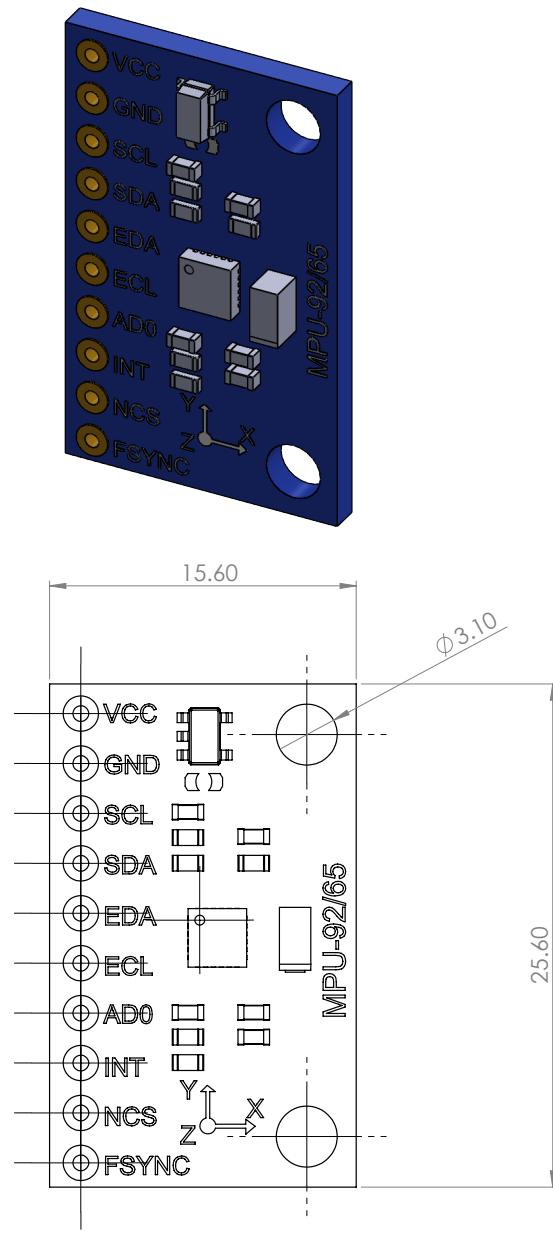


Producto SOLIDWORKS Educational. Solo para uso en la enseñanza.

Figura 10: Base

6.1.1. Componentes

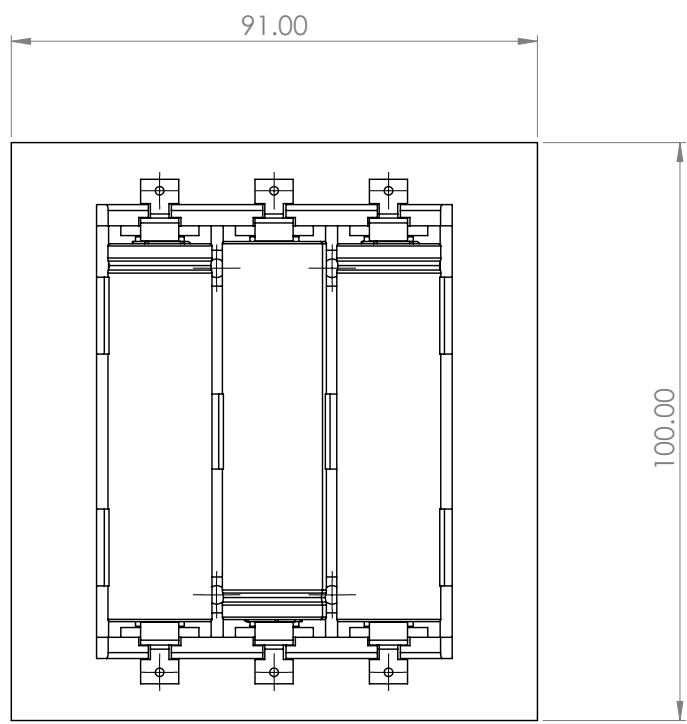
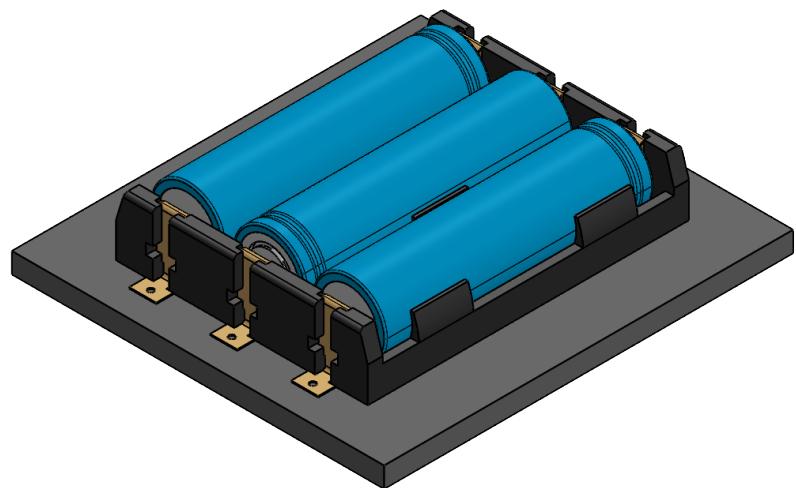
MPU-9250: Se tomo un modelo ya creado para poder integrarlo en el diseño digital final, pero tomando en cuenta las medidas que el componente tiene.



Producto SOLIDWORKS Educational. Solo para uso en la enseñanza.

Figura 11: Diseño y dimensiones del MPU-9250

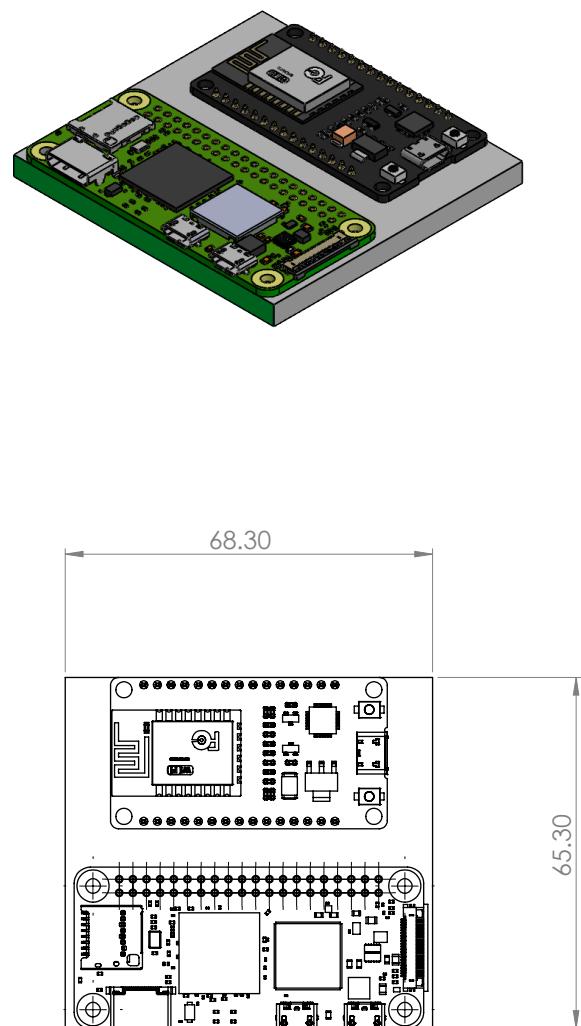
Fuente de voltaje: Se tomo un modelo de referencia para agregarlo al diseño digital final, pero se modifco su tamaño para que sus medidas sean acordes al sistema físico.



Producto SOLIDWORKS Educational. Solo para uso en la enseñanza.

Figura 12: Diseño y dimensiones de la fuente de voltaje.

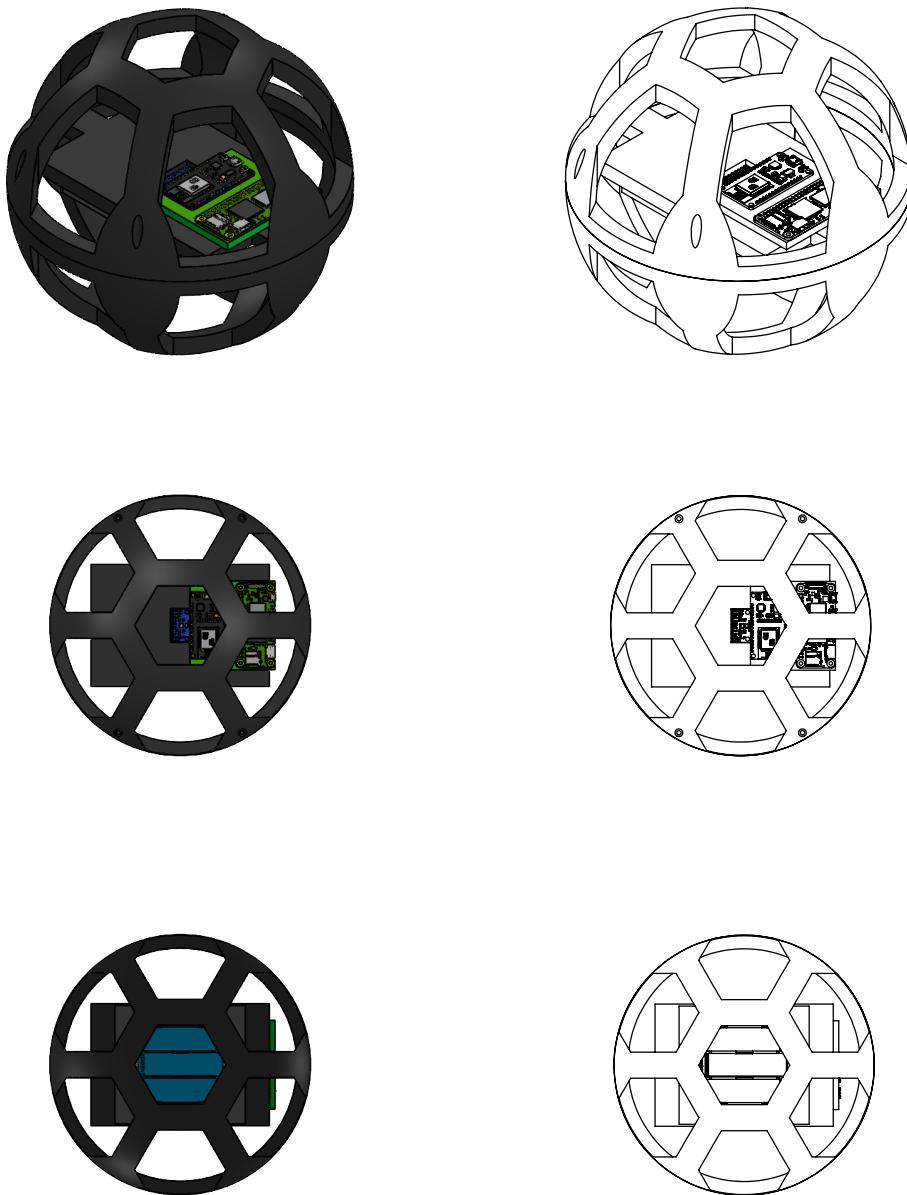
PCB: Se modelo tomando en cuenta el diseño creado usando el software Fritzing con las medidas con las cuales se va a fabricar y de esta manera poder agregarlo al diseño digital final.



Producto SOLIDWORKS Educational. Solo para uso en la enseñanza.

Figura 13: Diseño y dimensiones da la PCB con el ESP32 y el Raspberry Pi Zero 2W.

Estructura completa. El diseño en 3D considera dimensiones adecuadas para alojar componentes electrónicos con cortes y roscas que facilitan su montaje y desmontaje. Se priorizó la estructura ligera y resistente, con base para fijar componentes y cortes para maniobrar sin abrir la esfera completamente.



Producto SOLIDWORKS Educational. Solo para uso en la enseñanza.

Figura 14: Diseño de la esfera completa.

6.2. ESP32 MQTT.

Este es el código que se uso para la lectura del mpu 9250, el cual envía los datos a través de mqtt a la Raspberry.

Listing 1: Esp32

```
1 #include <Wire.h>
2 #include <WiFi.h>
3 #include <PubSubClient.h>
4 #include <MPU9250_asukiaaa.h>
5
6 const char* ssid = "pruebas2";
7 const char* password = "yolo15@A";
8
9 const char* mqtt_server = "10.214.7.60";
10 WiFiClient espClient;
11 PubSubClient client(espClient);
12
13 #ifdef ESP32_HAL_I2C_H
14 #define SDA_PIN 21
15 #define SCL_PIN 22
16 #endif
17
18 MPU9250_asukiaaa mpu;
19
20 void setup() {
21     Serial.begin(115200);
22     while(!Serial);
23
24     Wire.begin(SDA_PIN, SCL_PIN);
25     mpu.setWire(&Wire);
26     mpu.beginAccel();
27     mpu.beginGyro();
28     mpu.beginMag();
29     Serial.println("Sensor MPU9250 inicializado.");
30
31     WiFi.begin(ssid, password);
32     Serial.print("Conectando a WiFi...");
33     while (WiFi.status() != WL_CONNECTED) {
34         delay(500); Serial.print(".");
35     }
36     Serial.println("\nWiFi conectado");
37
38     client.setServer(mqtt_server, 1883);
39     Serial.print("Conectando a MQTT...");
40     while (!client.connected()) {
41         if (client.connect("ESP32Client")) {
42             Serial.println("\nMQTT conectado");
43         } else {
44             Serial.print("falló, rc=");
45             Serial.print(client.state());
46             Serial.println(" intentando de nuevo en 5 segundos");
47             delay(5000);
48     }
```

```

49     }
50 }
51
52 void loop() {
53   if (!client.connected()) {
54     setup();
55   }
56   client.loop();
57
58   uint8_t sensor_update_result = mpu.update();
59
60   if (sensor_update_result == 0) {
61     float ax = mpu.accelX();
62     float ay = mpu.accelY();
63     float az = mpu.accelZ();
64
65     float gx = mpu.gyroX();
66     float gy = mpu.gyroY();
67     float gz = mpu.gyroZ();
68
69     float mx = mpu.magX();
70     float my = mpu.magY();
71     float mz = mpu.magZ();
72
73     String payload = "{";
74     payload += "\"accel\":[\"" + String(ax, 4) + "\",\"" + String(ay, 4) + "\",\"" +
75                 String(az, 4) + "\"],";
76     payload += "\"gyro\":[\"" + String(gx, 4) + "\",\"" + String(gy, 4) + "\",\"" +
77                 String(gz, 4) + "\"],";
78     payload += "\"mag\":[\"" + String(mx) + "\",\"" + String(my) + "\",\"" + String(
79                 mz) + "\"]";
80     payload += "}";
81
82     client.publish("mpu9250/processed", payload.c_str());
83     Serial.println("Publicado:" + payload);
84   } else {
85     Serial.println("Error al leer el sensor MPU9250");
86   }
87
88   delay(100);
89 }

```

6.3. Raspberry MQTT -> ROS2.

Este es el código principal de la Raspberry, es el encargado de leer lo que se manda a través de mqtt a la ip local de la Raspberry y lo sube a ros 2 listo para ser usado por la herramienta de

Listing 2: Raspberry

```
1 import rclpy
2 from rclpy.node import Node
3 from sensor_msgs.msg import Imu
4 import paho.mqtt.client as mqtt
5 import json
6
7 G_A_METROS_S2 = 9.80665
8
9 class MQTTPuente(Node):
10     def __init__(self):
11         super().__init__('mqtt_a_ros_puente')
12
13         self.publisher_ = self.create_publisher(Imu, '/imu/data_raw', 10)
14
15         self.mqtt_client = mqtt.Client()
16         self.mqtt_client.on_connect = self.on_connect
17         self.mqtt_client.on_message = self.on_message
18         self.mqtt_client.connect("localhost", 1883, 60)
19         self.mqtt_client.loop_start()
20
21     def on_connect(self, client, userdata, flags, rc):
22         self.get_logger().info("Puente conectado al broker MQTT")
23         client.subscribe("mpu9250/raw")
24
25     def on_message(self, client, userdata, msg):
26         try:
27             data = json.loads(msg.payload.decode())
28
29             acc_g = data["accel"]
30             gyro_rads = data["gyro"]
31
32             imu_msg_raw = Imu()
33             imu_msg_raw.header.stamp = self.get_clock().now().to_msg()
34             imu_msg_raw.header.frame_id = "imu_link"
35
36             imu_msg_raw.orientation_covariance[0] = -1.0 # -1 significa "
37             # desconocida"
38
39             imu_msg_raw.angular_velocity.x = float(gyro_rads[0])
40             imu_msg_raw.angular_velocity.y = float(gyro_rads[1])
41             imu_msg_raw.angular_velocity.z = float(gyro_rads[2])
42             imu_msg_raw.angular_velocity_covariance[0] = 0.0
43
44             imu_msg_raw.linear_acceleration.x = float(acc_g[0]) *
```

```

45         G_A_METROS_S2
46     imu_msg_raw.linear_acceleration.z = float(acc_g[2]) *
47         G_A_METROS_S2
48     imu_msg_raw.linear_acceleration_covariance[0] = 0.0
49
50     self.publisher_.publish(imu_msg_raw)
51     self.get_logger().info('Publicando datos crudos en /imu/
52                             data_raw')
53
54 def main(args=None):
55     rclpy.init(args=args)
56     mqtt_puente_node = MQTT_Puente()
57     rclpy.spin(mqtt_puente_node)
58     mqtt_puente_node.destroy_node()
59     rclpy.shutdown()
60
61 if __name__ == '__main__':
62     main()

```

6.4. Gráficas de los resultados

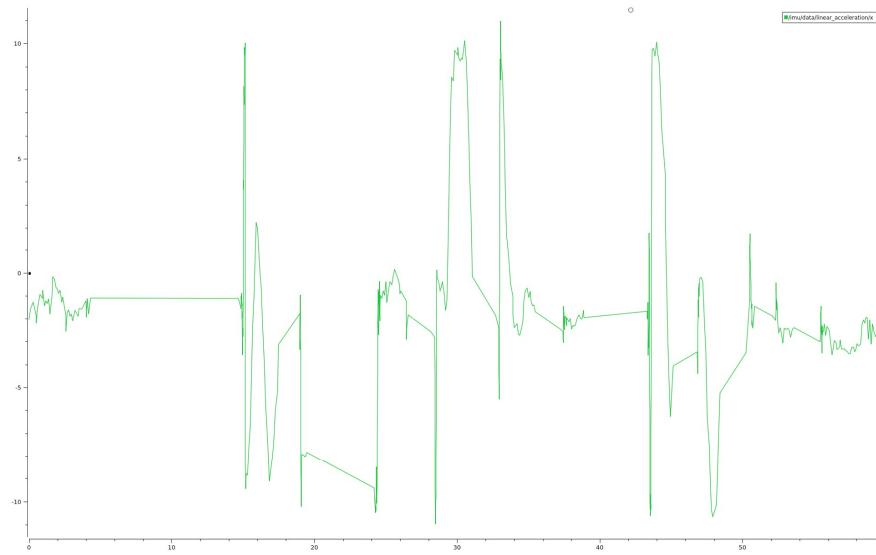


Figura 15: Gráfica de la Aceleración Lineal en X

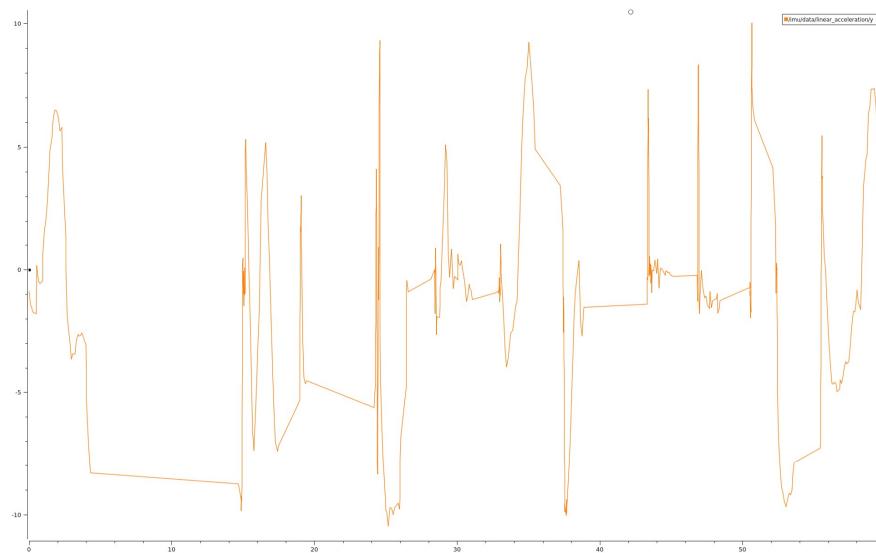


Figura 16: Gráfica de la Aceleración Lineal en Y

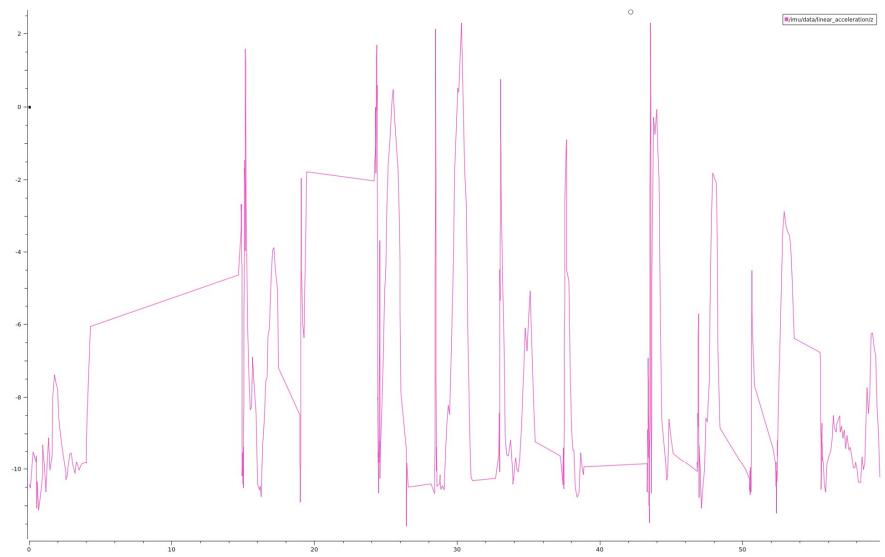


Figura 17: Gráfica de la Aceleración Lineal en Z

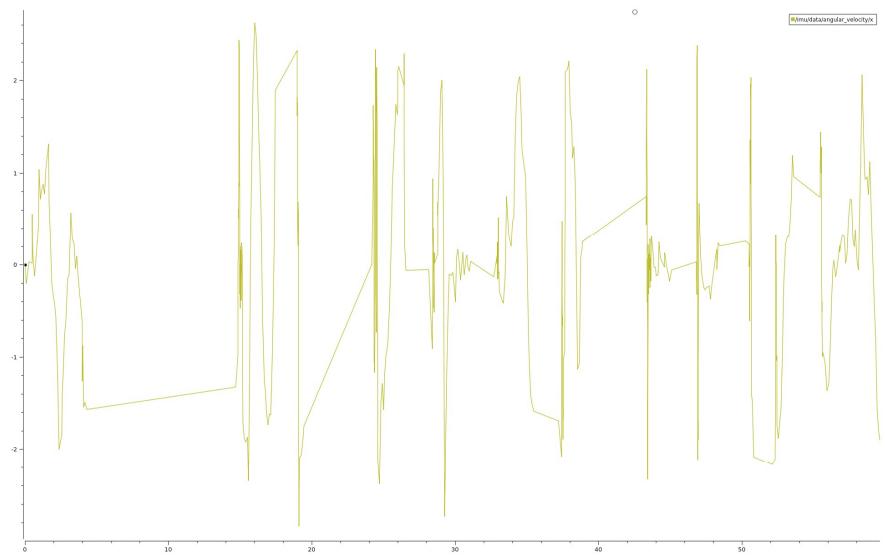


Figura 18: Gráfica de la Velocidad Angular en X

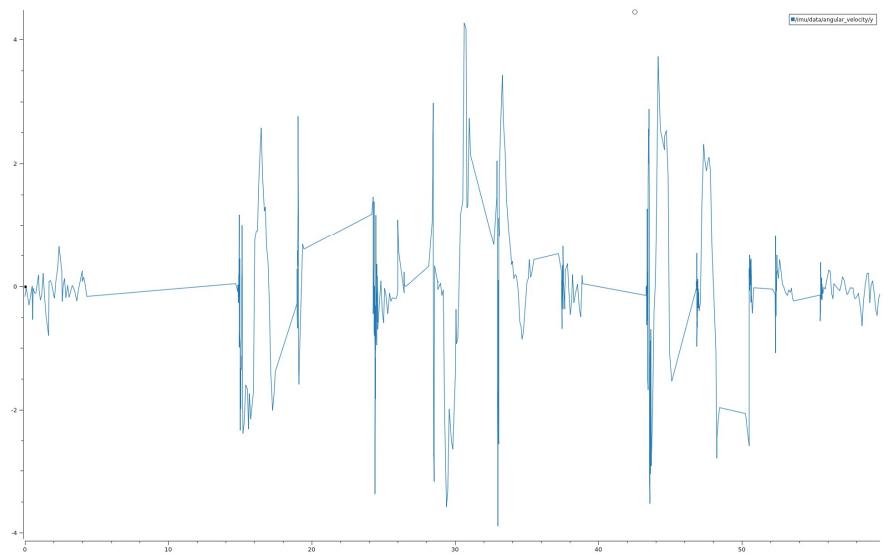


Figura 19: Gráfica de la Velocidad Angular en Y

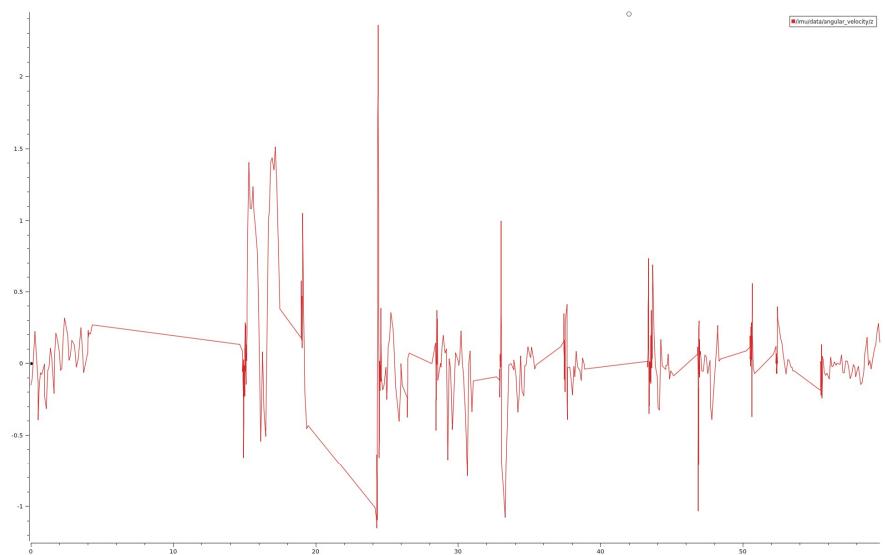


Figura 20: Gráfica de la Velocidad Angular en Z

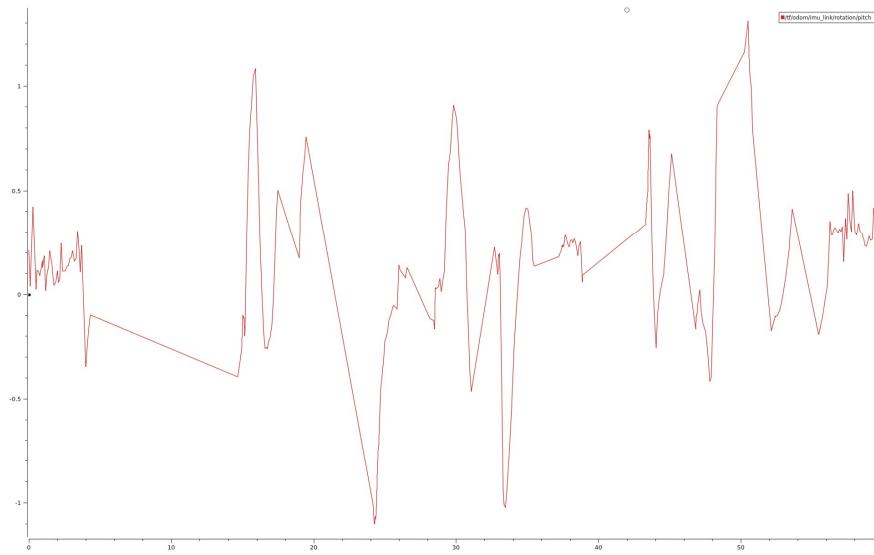


Figura 21: Gráfica de la Rotación Pitch

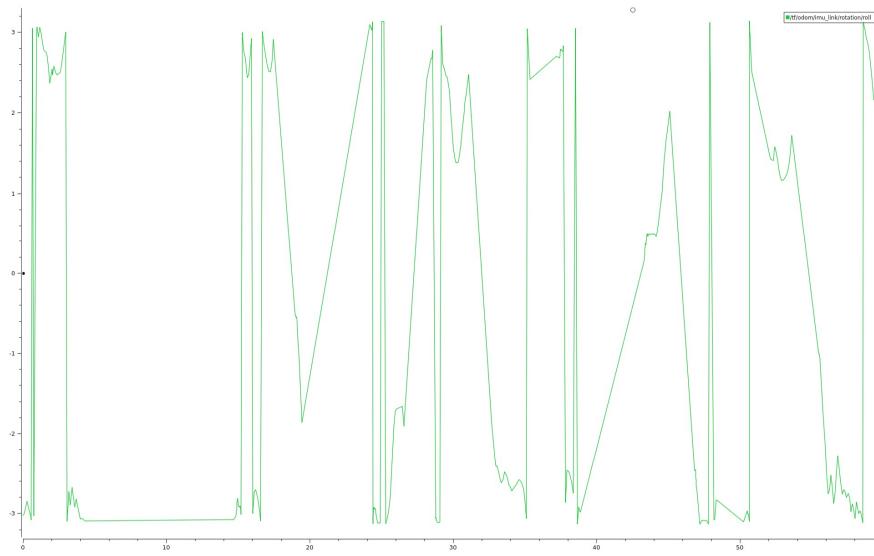


Figura 22: Gráfica de la Rotación Roll

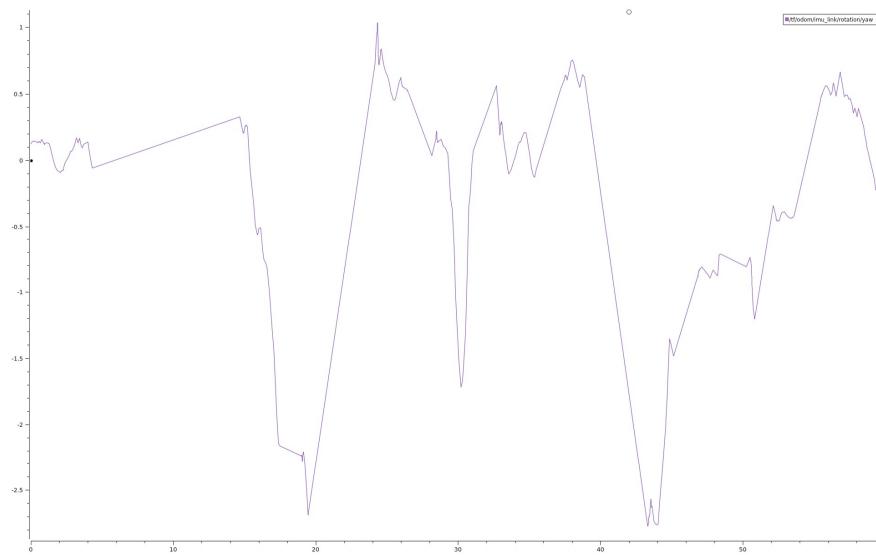


Figura 23: Gráfica de la Rotación Yaw

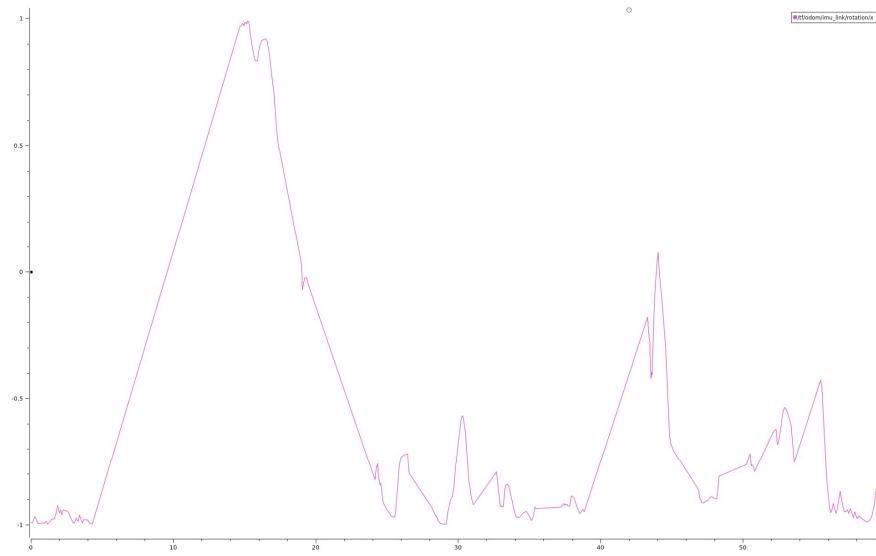


Figura 24: Gráfica de la Rotación X

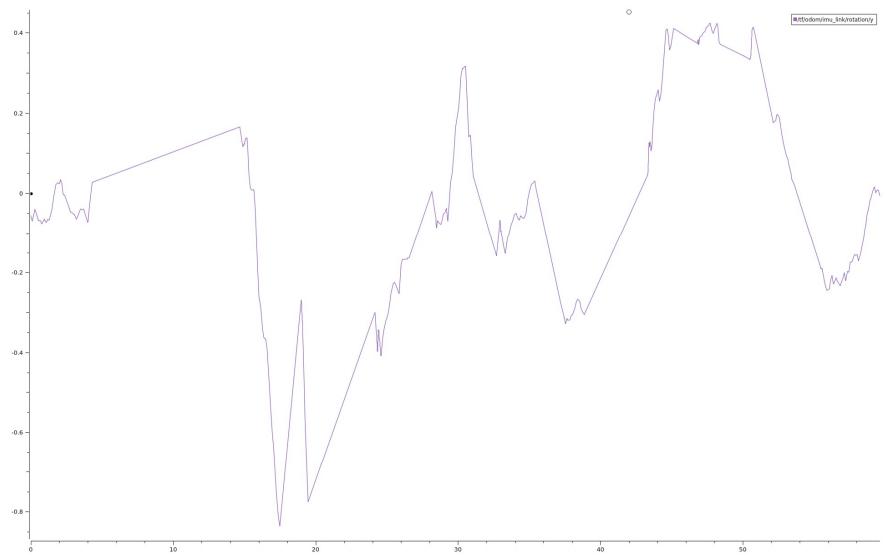


Figura 25: Gráfica de la Rotación Y

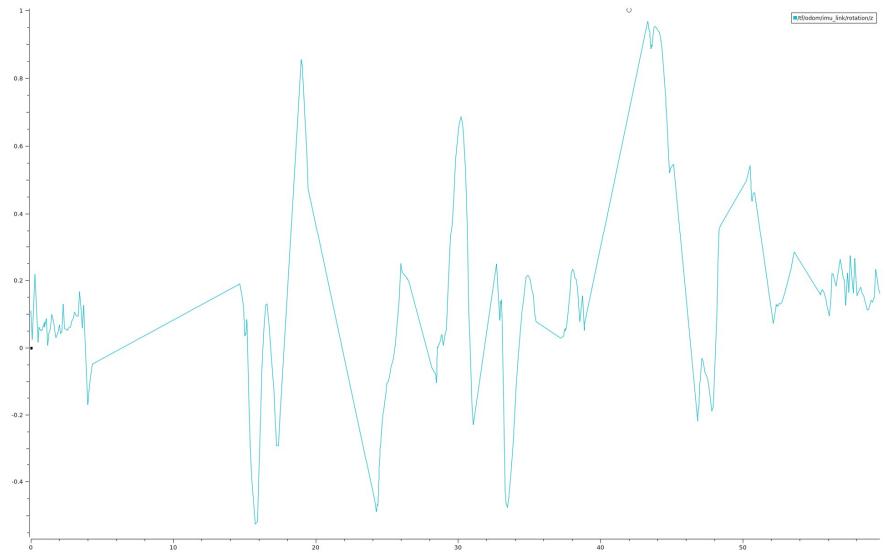


Figura 26: Gráfica de la Rotación Z

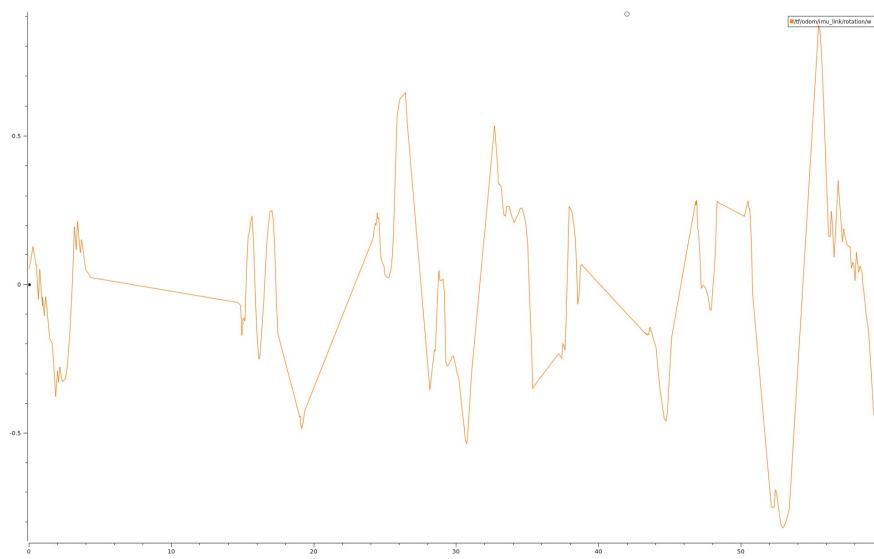


Figura 27: Gráfica de la Rotación W