

Trabajo Fin de Máster

Electrónica, Robótica y Automática

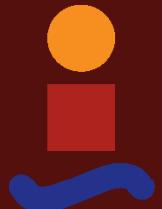
Desarrollo e Integración de un Sistema de Navegación, Estimación y Control para Ornitóptero

Autor: Álvaro García Lora

Tutor: José Ángel Acosta Rodríguez

Dpto. Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2024



Trabajo Fin de Máster
Electrónica, Robótica y Automática

Desarrollo e Integración de un Sistema de Navegación, Estimación y Control para Ornitóptero

Autor:
Álvaro García Lora

Tutor:
José Ángel Acosta Rodríguez
Profesor Titular

Dpto. Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2024

Trabajo Fin de Máster: Desarrollo e Integración de un Sistema de Navegación, Estimación y Control para Ornitóptero

Autor: Álvaro García Lora
Tutor: José Ángel Acosta Rodríguez

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

*A mis padres y hermana
A mis abuelos y familia
A mis amigos y compañeros*

Agradecimientos

A mi familia, por brindarme todo lo que está en sus manos. Todos mis logros son compartidos.

A mi padre y mejor amigo, Carlos, por transmitirme los valores que hoy día me hacen ser quien soy.

A mi madre y salvaguarda, Rocío, por ofrecerme el amor más sincero que se puede tener y facilitar el trabajo de todos con una sonrisa.

A mi hermana y compañera de aventuras, Marina, por su comprensión, actitud y apoyo incondicional. Cabe mencionar su rol de asistente en múltiples experimentos realizados.

A mis abuelas Gracita e Isabel y a mi abuelo Rafael por el cariño y ánimos que recibo.

A mi abuelo Juan, por transmitirme su fuerza y ser fuente extra de motivación. Me honra hacerte sentir orgulloso.

A mis compañeros de formación, en especial a Sergio, por todos estos años de compromiso y esfuerzo juntos.

A todos los profesores que me han impulsado en el camino. A mi señorita M.^a Carmen López por seguir con alegría mi evolución.

A mi tutor José Ángel Acosta, por sus principios, confianza y labor como docente.

A mi responsable de proyectos, Ángel R. Castaño, por su ayuda y el trato recibido durante el curso.

A mi compañera de trabajo Cristina Ruiz, por su participación en las pruebas realizadas y asesoramiento continuo.

A todas aquellas personas, las que siguen y las que se fueron, que dejaron su huella para hacerme mejor persona.

Álvaro García Lora
Ingeniero, Estudiante y Autor

Sevilla, 2024

Resumen

Los avances en el desarrollo de tareas autónomas en exteriores y en la implementación de controladores avanzados en el ámbito de la robótica aérea, requiere un conocimiento completo del estado del vehículo. Esta información es crucial para proporcionar una retroalimentación fiable al sistema.

El proyecto se centra en el diseño e integración de un sistema electrónico para monitorear y registrar datos de vuelo en un UAV de ala batiente tipo orníptero, utilizando un sensor GNSS/INS VN-200 de VectorNav y la placa Raspberry Pi Zero W. El objetivo principal es aportar una solución que permita la comunicación fluida entre el sensor y el sistema de procesamiento, garantizando la correcta recopilación y almacenamiento de telemetría para el análisis de vuelo.

Presentaremos un estudio exhaustivo del sensor GNSS/INS, esencial para comprender su funcionamiento y potencial, especialmente útil para usuarios poco familiarizados con este tipo de tecnología. Analizaremos tanto las herramientas de alto nivel proporcionadas por el fabricante como las librerías de comunicación de bajo nivel. También abordaremos el proceso de selección de una antena GNSS alternativa más ligera que la original. En el estudio, cubriremos la configuración y operación del módulo para obtener datos precisos de navegación y orientación, explorando a fondo sus capacidades. Aseguraremos una integración efectiva del sensor con el hardware del UAV, enfocándonos en la compatibilidad de protocolos e interfaces de comunicación.

Exploraremos cómo utilizar la placa Raspberry Pi para conectarnos de manera inalámbrica y manejar el sensor VN-200 para la grabación de datos de telemetría. Detallaremos el proceso de diseño y ensamblaje de los componentes electrónicos, garantizando su compatibilidad y funcionalidad dentro del chasis utilizado. Presentaremos las pruebas realizadas en condiciones controladas para validar el desempeño del sistema, ajustando los parámetros de configuración para optimizar la precisión y fiabilidad de los datos recopilados.

Finalmente, presentaremos los resultados de vuelo obtenidos y el análisis derivado de la información recopilada. Para ello, será necesario ajustar las superficies de alas y cola para compensar el incremento de peso y asegurar una distribución adecuada de los mismos.

Veremos que este proyecto no solo cumple con los objetivos iniciales de integración y recopilación de datos, sino que también establece una base sólida para futuras mejoras y aplicaciones. Los avances logrados son aplicables a una variedad de vehículos y contextos, destacando el valor intrínseco de la implementación y procedimiento presentado en este documento.

Abstract

Advances in the development of autonomous outdoor tasks and the implementation of advanced controllers in the field of aerial robotics require complete knowledge of the vehicle's status. This information is crucial to provide reliable feedback to the system.

The project focuses on the design and integration of an electronic system for monitoring and recording flight data in an ornithopter, a type of flapping-wing UAV, using a VectorNav VN-200 GNSS/INS sensor and the Raspberry Pi Zero W board. The main objective is to provide a solution that allows seamless communication between the sensor and the processing system, ensuring the correct collection and storage of telemetry for flight analysis.

An exhaustive study of the GNSS/INS sensor will be presented, essential to understand its operation and potential, especially useful for users unfamiliar with this type of technology. We will look at both the high-level tools provided by the manufacturer and the low-level communication libraries. We will also discuss the process of selecting an alternative GNSS antenna that is lighter than the original one. In the study, we will cover the configuration and operation of the module to obtain accurate navigation and orientation data, exploring its capabilities in detail. We will ensure an effective integration of the sensor with the UAV hardware, focusing on the compatibility of communication protocols and interfaces.

We will explore how to use the Raspberry Pi board to wirelessly connect and operate the VN-200 sensor for telemetry data logging. We will detail the design and assembly process of the electronic components, ensuring their compatibility and functionality within the chassis used. We will present the tests carried out under controlled conditions to validate the operation of the system, adjusting the configuration parameters to optimise the accuracy and reliability of the acquired data.

This project not only fulfils the initial objectives of integration and data collection, but also establishes a solid background for future improvements and applications. The advances achieved are applicable to a wide variety of vehicles and contexts, highlighting the inherent value of the procedure and application presented in this document.

Índice

<i>Resumen</i>	V
<i>Abstract</i>	VII
1 Introducción	1
1.1 Contexto y motivación	1
1.2 Objetivos y especificaciones	2
1.3 Antecedentes	2
1.4 Estructura del documento	3
2 Análisis del Sistema de Navegación Inercial VectorNav VN-200	5
2.1 Aproximación a VN-200	5
2.1.1 Características técnicas	6
2.1.2 Sistema de coordenadas	7
2.1.3 VectorNav Control Center	9
2.1.4 Librería Vnproglib	9
2.2 Procedimiento y consideraciones de uso	10
2.3 Comparativa y Selección de Antenas GNSS	12
2.4 Comunicación externa	15
2.5 Requisitos del sistema de navegación	15
3 Descripción y Programa del Computador de a Bordo: Raspberry Pi Zero W	17
3.1 Presentación de Raspberry Pi Zero W	17
3.1.1 Características técnicas	18
3.1.2 Configuración y ajustes iniciales	19
3.1.3 Herramientas software en la estación de tierra	19
3.2 Programa principal	20
4 Integración Hardware	23
4.1 Conexionado y comunicaciones de componentes	23
4.2 Concepción y ensamblaje	25
4.3 Verificación funcional	27
5 Resultados de vuelo	31
5.1 Adaptaciones en superficies aerodinámicas	31
5.2 Pruebas de vuelo y análisis de resultados	32
6 Conclusiones y Trabajo futuro	41
6.1 Objetivos generales y particulares	41
6.2 Perspectivas de desarrollo	42
Apéndice A Registros internos VectorNav	45

A.1	Registro 26: Reference Frame Rotation	45
A.2	Registro 57: GNSS Internal A Antenna Offset	45
A.3	Registro 67: INS Basic Configuration	46
A.4	Registros 75-77: User Output Configuration	46
A.5	Registro 99: GNSS System Configuration	47
A.6	Registro 105: INS Reference Point Offset	48
Apéndice B	Código completo programa principal	49
<i>Índice de Figuras</i>		53
<i>Índice de Tablas</i>		55
<i>Bibliografía</i>		57

1 Introducción

El éxito no es la clave de la felicidad. La felicidad es la clave del éxito. Si amas lo que haces, tendrás éxito.

ALBERT SCHWEITZER

El conocimiento del estado completo de un robot de ala batiente requiere un módulo INS (Sistema de Navegación Inercial) lo suficientemente preciso para operar en exteriores. Esta unidad asegura que las medidas de los sensores se fusionen correctamente mediante técnicas de filtrado avanzado, proporcionando datos fiables de alta calidad necesarios para realimentar estrategias de control autónomo.

Este continúa los avances alcanzados en el desarrollo del proyecto HOMPOT de la Junta de Andalucía, dentro del marco PAIDI 2020 y con código P20_00597 con financiación europea de fondos FEDER.

Tomaremos como punto de partida el *Proyecto Fin de Grado* con autoría propia, titulado "*Mecatrónica, Modelado y Control del Robot Aéreo de Ala Batiente HOMPOT*". En dicha investigación se asentaron las bases de una línea de trabajo en la que se creó un sistema robótico capaz de cumplir las funciones básicas de vuelo de un orníptero, teniendo total control y conocimiento de la plataforma, sobre la cual continuar progresando.

1.1 Contexto y motivación

Los vehículos no tripulados se pueden clasificar en tres categorías principales: vehículos terrestres (UGV, Unmanned Ground Vehicle), embarcaciones de superficie (USV, Unmanned Surface Vessel) y vehículos aéreos (UAV, Unmanned Aerial Vehicle), que operan en tierra, en el agua o en el aire, respectivamente. Estos vehículos tienen el potencial de aumentar la eficiencia, reducir costos y minimizar el riesgo para la vida humana, convirtiéndose en herramientas esenciales en una variedad de aplicaciones. La implementación exitosa de un vehículo no tripulado depende en gran medida del uso de un sistema de navegación de alta calidad para garantizar un control preciso a través de entornos desafiantes y misiones complejas.

Los UAVs son aeronaves que operan sin piloto a bordo, controladas de forma remota o de manera autónoma. Estos vehículos tienen aplicaciones variadas, incluyendo uso militar, vigilancia, fotografía aérea, agricultura y transporte. Los UAVs pueden diferir en tamaño y configuración según su propósito.

En este proyecto, nos centramos en el desarrollo de ornípteros, una categoría de UAV que utiliza alas batientes para imitar el vuelo de las aves. A diferencia de los UAVs tradicionales, que emplean hélices o alas fijas, los ornípteros buscan replicar los principios aerodinámicos de las aves para lograr una mayor eficiencia energética, menor impacto acústico y mejora de la seguridad. Estas características hacen de los ornípteros una opción prometedora para aplicaciones que requieren discreción, como la vigilancia de fauna y la investigación en entornos sensibles [1].

1.2 Objetivos y especificaciones

El objetivo principal es el de integrar un sistema de navegación inercial en una plataforma robótica capaz de realizar las funciones básicas de vuelo de un orníptero, con vistas a disponer de una estimación fiable y completa de su estado (posición, velocidad y actitud).

Para obtener una reconstrucción lo más precisa posible del vuelo del orníptero, se requiere una tasa de muestreo mínima de 40 Hz, aunque lo ideal sería alcanzar los 100 Hz. Esta tasa de muestreo es necesaria para capturar adecuadamente los detalles del movimiento, dado que el sistema operará a una frecuencia máxima de aleteo próxima a los 10 Hz.

Partiendo de un prototipo radiocontrol, el objetivo es añadir un sistema electrónico para la navegación y control propio que posibilite la grabación de los datos de telemetría de vuelo ofrecidos por el INS, el cual debe ser estudiado en detalle y configurado de forma conveniente para satisfacer nuestras necesidades.

Nos enfrentaremos a varios desafíos significativos donde el correcto aprovechamiento de los componentes existentes y la exploración de las diversas posibilidades serán cruciales para el éxito de su implementación práctica. Las restricciones de peso y espacio serán críticas, ya que cualquier exceso podría comprometer la estabilidad y el rendimiento del sistema en vuelo.

1.3 Antecedentes

Como hemos anticipado anteriormente, partiremos de los avances alcanzados y conocimiento adquiridos en nuestro *Proyecto Fin de Grado* [2], donde nos centramos en la modificación del sistema mecatrónico y electrónico interno de un producto existente en el mercado.

La plataforma seleccionada como chasis fue el GoGoBird® 1020 – Eagle del fabricante Hanvon, debido a su resistencia frente a golpes, su bajo peso con relación al tamaño y su envergadura reducida, menor a 1 metro, la cual permite una mayor maniobrabilidad. Cuenta con 3 partes principales desmontables: cuerpo, cola y alas, véase Figura 1.1. Su funcionalidad se limita a realizar movimientos de vuelo básicos con fines recreativos. Las principales características técnicas quedan recogidas en Tabla 1.1.



Figura 1.1 Producto comercial GoGobird® 1020 Eagle - Hanvon.

En primer lugar, después de un exhaustivo estudio de ingeniería inversa sobre la plataforma presentada, se caracterizó por completo cada una de sus partes, así como su funcionamiento en conjunto.

Tras esto, se elaboró un diseño y montaje de hardware reutilizando componentes existentes (placa reguladora de potencia con ESC y motor BLDC para movimiento del mecanismo de alas, servomotores para los flaps de cola, sensor de efecto Hall para medir la frecuencia de aleteo y chasis general), así como añadiendo otros nuevos (sensor barómetro BMP280 para la estimación de altitud, IMU MPU6050 de 6 ejes para la estimación de la orientación y la placa WeMos D1 Mini Wi-Fi con microcontrolador ESP8266 como computador de a bordo). Esto supuso un incremento de peso total de 15 gramos.

Tabla 1.1 Características GoGoBird.

Característica	Valor
Peso completo con baterías	170 g
Comunicación mando control	Radiofrecuencia
Envergadura alas	61.5 cm
Longitud cuerpo - cola	44 cm
Rango control remoto	500 m
Batería	Ion-Litio 2S 520 mAh
Autonomía	15 min

Para el control manual, se usó un mando de XBOX. Generamos a nivel software los programas necesarios para el envío y recepción de comandos de movimiento y datos de telemetría. Incorporamos tanto los controles básicos como nuevas funcionalidades adicionales, tales como: parada sincronizada de alas para planeo, detención de emergencia, cambio entre modos y mayor libertad de movimientos en la cola. Todo el esquema de comunicaciones se basó en el envío de mensajes UDP a través de Wi-Fi.

Gracias a este trabajo resumido, se pudieron realizar pruebas de vuelo satisfactorias, tras las cuales se comenzaron a plantear tareas de control autónomo. Sin embargo, los sensores utilizados en este punto, debido a su bajo costo, ofrecían medidas de calidad limitada. Para obtener datos más fiables, era necesario recurrir a soluciones alternativas, siendo indispensable la inclusión de un INS preciso. Este requerimiento motivó el desarrollo de este Trabajo Fin de Máster, cuyo fin es el cumplimiento de los objetivos descritos.

1.4 Estructura del documento

En los próximos capítulos entraremos en los detalles técnicos abordados y presentaremos la línea de trabajo seguida.

Iniciaremos con un análisis exhaustivo del sensor GNSS/INS utilizado, abarcando las herramientas disponibles, su configuración, las modificaciones realizadas, su funcionamiento, así como las claves para su comunicación externa e integración, que se detallarán en el Capítulo 2.

Continuaremos con el Capítulo 3, donde presentaremos las capacidades hardware del computador de a bordo seleccionado, las herramientas para trabajar con él de forma inalámbrica y el programa desarrollado para recibir y registrar los datos de telemetría.

El Capítulo 4 explorará el diseño e integración del sistema electrónico en el ornitóptero, además de presentar las pruebas de funcionamiento realizadas para asegurar la operatividad del conjunto.

Una vez descritas todas las partes y completado el montaje del sistema, en el Capítulo 5, abordaremos las modificaciones realizadas en las superficies aerodinámicas para compensar el peso añadido y analizaremos los resultados obtenidos durante las pruebas de vuelo.

Finalmente, en el Capítulo 6 se darán reflexiones a modo de conclusión y se discutirán las líneas de trabajo futuro con las que se podría continuar avanzando.

2 Análisis del Sistema de Navegación Inercial VectorNav VN-200

Antes que toda otra cosa, la preparación es la clave para el éxito.

ALEXANDER GRAHAM BELL

El sistema de navegación inercial VectorNav VN-200 será fundamental para la estimación precisa de la pose en vuelo de nuestro orníptero. Se ofrecerá una descripción detallada del dispositivo seleccionado, incluyendo sus características técnicas, capacidades y el proceso de integración en el sistema robótico aéreo. Se abordarán los aspectos clave como la comunicación externa con el dispositivo, uso del software del fabricante y la configuración de registros internos, basándonos en un profundo estudio de manuales y pruebas realizadas en exteriores con el módulo.

Además, se discutirá el conocimiento adquirido sobre el VN-200 y su aplicación práctica, así como la selección de una nueva alternativa de antena GNSS más ligera. Este análisis detallado permitirá comprender cómo el dominio completo del sensor es crucial para lograr disponer de los datos necesarios para futuros trabajos.

2.1 Aproximación a VN-200

De acuerdo con el fabricante VectorNav, el VN-200 es un sistema de navegación inercial INS asistido por GNSS (GNSS/INS, por sus siglas en inglés "Global Navigation Satellite System / Inertial Navigation System") en miniatura, de montaje en superficie y alto rendimiento. Este sistema combina la última tecnología de sensores MEMS (por sus siglas en inglés "Microelectromechanical Systems") de estado sólido, que incluye un conjunto de acelerómetros de 3 ejes, giroscopios de 3 ejes, un magnetómetro de 3 ejes, un sensor de presión barométrica, un receptor GNSS L1 de 72 canales, así como un procesador de 32 bits en un módulo miniatura de montaje en superficie.



Figura 2.1 Vectornav VN-200 Rugged.

La integración de un INS y un GNSS permite combinar sus fortalezas: el INS ofrece alta precisión a corto plazo pero acumula errores con el tiempo, lo que se conoce como deriva (drift), mientras que el GNSS es más estable a largo plazo, aunque más ruidoso a corto. Al combinar ambos, las mediciones del GNSS regulan los errores acumulativos del INS, mientras que el INS proporciona soluciones de navegación a alta frecuencia, llenando los intervalos entre las actualizaciones más lentas del GNSS. De esta forma, VN-200 integra las mediciones del módulo GNSS con los datos de los sensores iniciales a bordo, ejecutando un robusto Filtro Extendido de Kalman (EKF) para ofrecer estimaciones de posición, velocidad y actitud con una precisión superior y un rendimiento dinámico mejorado en comparación con un módulo GNSS independiente o un Sistema de Referencia de Actitud y Rumbo (AHRS, por sus siglas en inglés "Attitude Heading Reference System").

Existen dos versiones disponibles del modelo: VN-200 SMD y VN-200 Rugged. El dispositivo que usaremos en el proyecto, y por tanto, sobre el que nos centraremos en este documento será el VN-200 Rugged, el cual cuenta con un encapsulado de aluminio para aislamiento y protección. La apariencia externa del equipo puede verse en la Figura 2.1.

A lo largo del capítulo se proporcionarán multitud de detalles extraídos de manuales y datasheets que pueden encontrarse en la documentación oficial del fabricante [3].

2.1.1 Características técnicas

Las principales características mecánicas, ambientales y eléctricas del producto quedan recogidas en la Tabla 2.1.

Tabla 2.1 Características generales de VN-200 Rugged.

Característica	Valor
Tamaño	36 x 33 x 9.5 mm
Peso	16 g
Voltaje de alimentación	3.3 a 17 V
Consumo Energético (sin antena GNSS)	80 mA @ 5 V 500 mW
Interfaz de comunicación externa	Serial RS-232 & TTL
Temperatura de operación	-40° a +85° C

El conector principal es un Harwin M80-5001042 de 10 pines, mientras que el conector de acoplamiento utilizado en los conjuntos de cables proporcionados por VectorNav para su uso con el VN-200 Rugged es un Harwin M80-4861005. El conector utilizado para la antena es del tipo MMCX hembra. Todo ello queda recogido en los esquemáticos del producto en la Figura 2.2. La descripción resumida de cada uno de los pines puede consultarse en la Tabla 2.2. Esta información es bastante relevante de cara al ensamblaje del sistema electrónico propio.

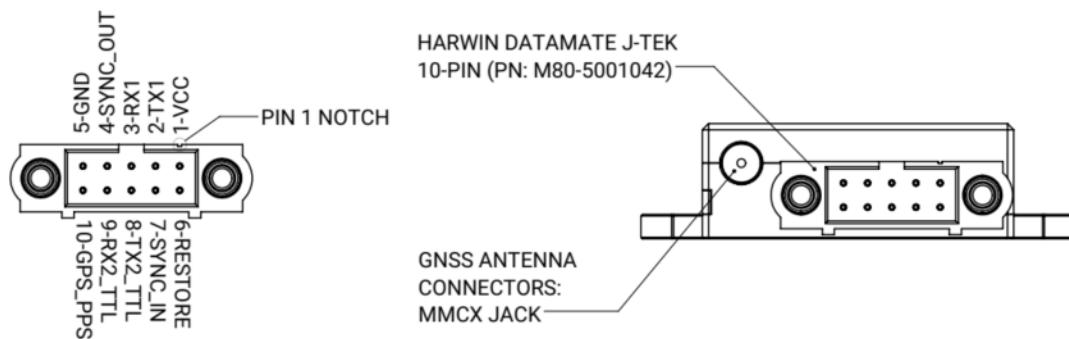


Figura 2.2 Conectores y pinout del Vectornav VN-200 Rugged.

Tabla 2.2 Asignaciones de pines VN-200 Rugged.

Pin	Nombre	Tipo	Descripción
1	VCC	Supply	Alimentación única. Rango: 3.3 a 17 V
2	TX1	Output	Salida de datos a niveles de tensión RS-232 (Serial UART-1)
3	RX1	Input	Entrada de datos a niveles de tensión RS-232 (Serial UART-1)
4	SYNC_OUT	Output	Señal de salida utilizada con fines de sincronización.
5	GND	Supply	Referencia / Ground
6	RESTORE	Input	Restablecer la configuración de fábrica
7	SYNC_IN	Input	Señal de entrada para fines de sincronización
8	TX2_TTL	Output	Salida de datos a nivel de tensión TTL (3 V) (Serial UART-2)
9	RX2_TTL	Input	Entrada de datos a nivel de tensión TTL (3 V) (Serial UART-2)
10	GPS_PPS	Output	Salida TTL (3 V) conectada al pin PPS del receptor GNSS

2.1.2 Sistema de coordenadas

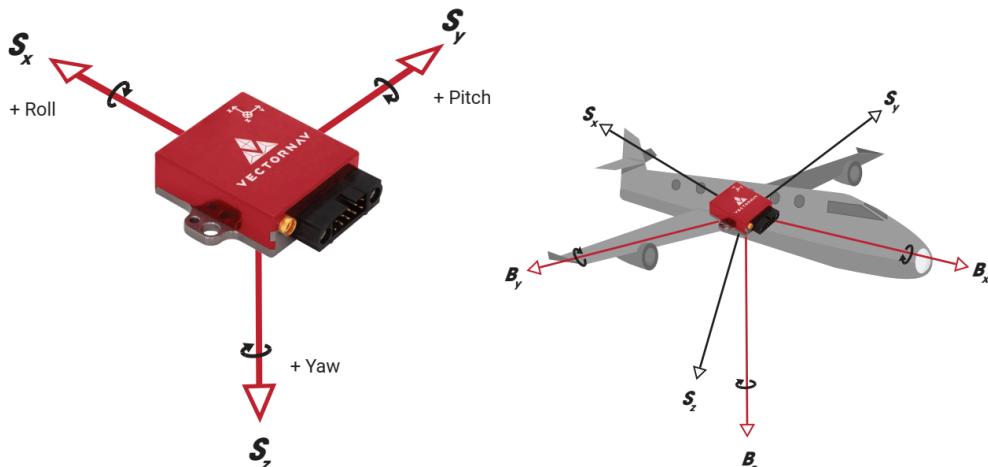
Continuamos definiendo y explicando brevemente los distintos sistemas de referencia con los que trabajaremos. Esta aclaración facilita una correcta interpretación y conversión de los datos.

Sistema de coordenadas del sensor (Sensor Frame)

El VN-200 utiliza un sistema de coordenadas a derechas. El eje X apunta hacia delante, el eje Y apunta hacia la derecha y el eje Z hacia abajo. El ángulo de guiñada (yaw) positivo se define como una rotación positiva hacia la derecha alrededor del eje Z. El ángulo de cabeceo (pitch) positivo se define como una rotación positiva hacia la derecha alrededor de el eje Y. El ángulo de balanceo (roll) positivo se define como una rotación positiva hacia la derecha alrededor del eje X. Los ejes con respecto al módulo VN-200 se muestra en la Figura 2.3 (S_x, S_y, S_z).

Sistema de coordenadas del cuerpo (Body Frame)

Sistema de coordenadas específico que está alineado con un objeto o dispositivo rígidamente conectado al sensor, en nuestro caso, el UAV. Supone una referencia establecida por nosotros mismos o por normativa, y en la mayoría de aplicaciones difiere del marco del sensor. Es útil para corregir desalineaciones mecánicas, errores de montaje o para integrar el sensor en un sistema más amplio. En la Figura 2.3 puede verse un ejemplo genérico de este concepto (B_x, B_y, B_z).

**Figura 2.3** Marcos de referencia sensor y cuerpo.

Coordenadas LLA

Las coordenadas LLA (Latitud, Longitud y Altitud) son utilizadas por el VN-200 para estimar la posición. Estas coordenadas están en un marco geodésico no inercial con origen en la superficie de la Tierra, específicamente el elipsoide WGS84, que es el sistema de referencia utilizado globalmente para coordenadas geográficas.

- Latitud (ϕ): Es el ángulo desde el plano ecuatorial hasta una línea normal a la superficie del elipsoide WGS84 en la ubicación del VN-200. La latitud varía de 0° en el ecuador a $\pm 90^\circ$ en los polos norte y sur, respectivamente.
- Longitud (λ): Es el desplazamiento angular en dirección este-oeste medido positivamente hacia el este desde el Meridiano de Referencia del IERS (International Earth Rotation and Reference Systems Service) hasta la ubicación del VN-200. La longitud varía de 0° en el Meridiano de Greenwich a $\pm 180^\circ$.
- Altitud (h): Es la distancia desde el elipsoide WGS84 hasta la ubicación del VN-200 en una dirección normal al elipsoide. Esta medida toma en cuenta la forma elíptica de la Tierra, proporcionando una altitud precisa respecto al nivel del mar según el modelo WGS84.

Coordenadas NED

Las coordenadas NED (North-East-Down) son un sistema de referencia utilizado por el VN-200 para estimar la velocidad. Nuevamente, este sistema de coordenadas es un marco cartesiano, geodésico y no inercial con el origen ubicado en la superficie de la Tierra según el elipsoide WGS84.

- Eje X (N_x): El eje positivo X apunta hacia el norte y es tangente al elipsoide WGS84 en la superficie de la Tierra. Este eje sigue la dirección del meridiano en la ubicación del VN-200.
- Eje Y (N_y): El eje positivo Y apunta hacia el este y también es tangente al elipsoide WGS84 en la superficie de la Tierra. Este eje sigue la dirección del paralelo en la ubicación del VN-200.
- Eje Z (N_z): El eje positivo Z apunta hacia abajo, penetrando la superficie terrestre. Completa el sistema de coordenadas de acuerdo a la regla de la mano derecha.

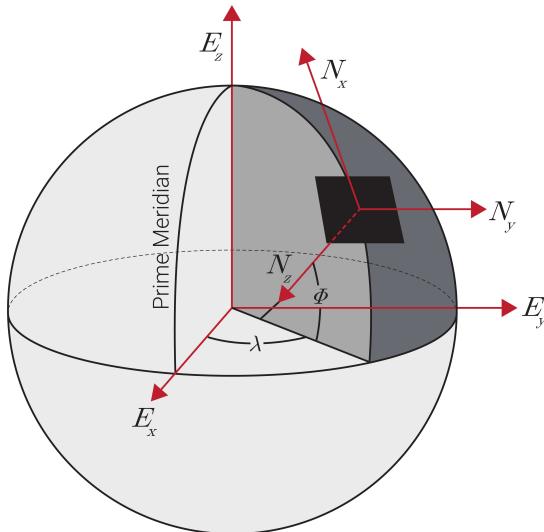


Figura 2.4 Marcos de referencia LLA y NED.

2.1.3 VectorNav Control Center

El fabricante VectorNav ofrece una serie de recursos software para configurar y usar su producto, proporcionando una capa de abstracción de más alto nivel. Existen una serie de herramientas que permiten cambiar valores de registros internos, acceder y procesar datos actuales de los distintos submódulos integrados en el VN-200, así como realizar grabaciones de experimentos y exportar / importar los datos resultantes. El programa principal con el que hemos trabajado ha sido *VectorNav Control Center* [4], disponible para Windows.

A continuación describiremos las funcionalidades básicas que más hemos empleado en el proceso de configuración y pruebas del equipo. Para una guía detallada de uso puede consultarse la documentación oficial del fabricante [3].

La conexión con el PC se hace a través del adaptador USB a Serial disponible. La comunicación del sensor con el software es trivial usando la GUI, seleccionando el puerto COM y la velocidad de transmisión correcta (115200 baud por defecto).

Para configurar los ajustes deseados en el VN-200, se utiliza el comando Write Register. Sin embargo, este comando solo guarda los ajustes en la memoria volátil del sensor, lo que significa que se perderán tras un ciclo de encendido o un reinicio del dispositivo. Para conservar los ajustes del usuario en la memoria no volátil del sensor y asegurar que persistan a través de ciclos de encendido o reinicios, se debe enviar un comando Write Settings después de aplicar los comandos Write Register.

Durante el proceso de puesta en marcha, la ventana *Sensor Status* resulta muy útil. En ella podemos ver detalles interesantes tales como número de satélites recibidos, el modo de INS, el estado de GNSS, actitud, posición LLA y velocidad NED estimadas, así como las incertidumbres de dichas estimaciones.

Realizar grabaciones de nuevos datos e importarlos de experimentos anteriores para su análisis es posible con la funcionalidad *Data Log and Playback*. La extensión del fichero es propia de VectorNav (.vnlog). Si se desea exportar los datos a otros formatos (CSV, Matlab, entre otros) es posible hacerlo con la ventana *Log Explorer*.

Para seleccionar la información de salida del sensor, clasificada en distintas categorías, contamos con la ventana *Outputs*. A su vez, para una visualización de forma gráfica en tiempo real, puede usarse la funcionalidad *Plot*, seleccionando los datos concretos dentro de la ventana.

VN-200 cuenta con numerosa documentación, repartida en extensos manuales accesibles en la sección de soporte y recursos de su web oficial [3]. A su vez, el propio *Control Center* integra dicha documentación, y es accesible a través de la ventana *Help Window*. Es bastante recomendable su uso de cara a conocer detalles de registros internos y datos, ya que ayuda a agilizar el proceso de estudio del producto.

Otras herramientas de interés son la vista 3D del sensor disponible en *3D View* o la evolución temporal de solución de actitud INS en la ventana *Yaw, Pitch, Roll*. Por otro lado, *Street Map* sirve para visualizar la localización del sensor en el mapa con vista satélite (necesita conexión a Internet) y *Sky Plot* muestra un gráfico la disposición de satélites de las distintas constelaciones activadas (GPS, Galileo, GLONASS o BeiDou) y estaciones SBAS (Satellite-Based Augmentation System).

2.1.4 Librería Vnproglb

La librería *vnproglb* (VectorNav Programming Library) [5] es un conjunto integral de recursos diseñado para desarrolladores que trabajan con los productos de VectorNav. Esta librería está organizada en diferentes secciones, cada una enfocada en un entorno de desarrollo y lenguaje de programación específico como C, C++, .NET, Python, MATLAB, LabVIEW y Unity, permitiendo un acceso a material adaptado a las distintas necesidades particulares y facilitando significativamente el proceso de integración en sistemas más complejos.

En el desarrollo del proyecto, hemos puesto especial atención en la sección de la librería correspondiente a C++, orientada al desarrollo en sistemas con un sistema operativo, como se detallará en el Capítulo 3.

2.2 Procedimiento y consideraciones de uso

Comprender el proceso de puesta en marcha y configuración básica del VN-200 es fundamental para garantizar que el sensor funcione nominalmente. A continuación se presentarán una serie de puntos que han sido abordados en este trabajo. Todos ellos proceden de la documentación oficial de VectorNav y su soporte. Estas recomendaciones han sido aplicadas de forma práctica en nuestro proyecto.

Rotación del marco de referencia

El VN-200 puede ser montado en cualquier orientación en la plataforma de instalación final. La rotación del marco de referencia es una matriz de rotación 3x3 que asignará los ejes del sensor a los ejes del orníptero, lo que permitirá al VN-200 dar los datos de salida en el marco de referencia del cuerpo del mismo. Véase Figura 2.5.

El cambio es aplicable a través del registro *Reference Frame Rotation* (Registro 26), véase Sección A.1, siendo la matriz R la correspondiente en nuestra aplicación particular.

$$R = \begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & 1 \\ -1 & 0 & 0 \end{pmatrix}$$

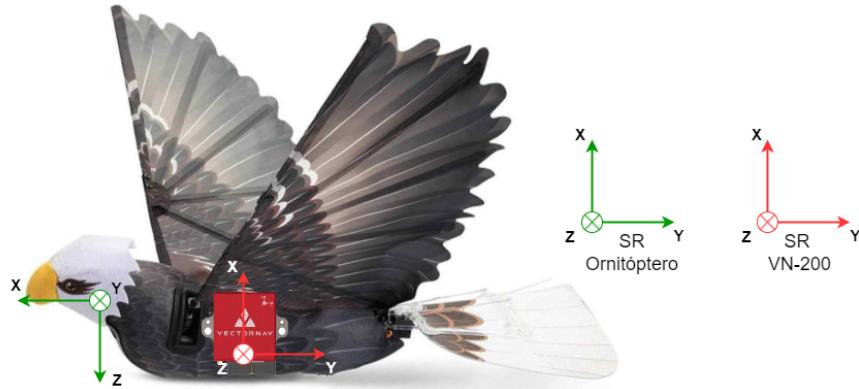


Figura 2.5 Rotación de marcos de referencia.

Offset de antena GNSS y punto de referencia de INS

El filtro INS proporciona la solución de navegación relativa al Punto de Referencia INS, que por defecto está en el centro del sensor inercial. Para obtener datos en un punto diferente, se debe aplicar un Desplazamiento del Punto de Referencia INS, que solo traduce la solución de navegación sin afectar los datos iniciales. Este desplazamiento se configura en el registro *INS Reference Point Offset* (Registro 105), véase Sección A.6.

Por defecto, el VN-200 asume que la antena GNSS está colocada muy próxima (a un máximo de 10 cm) del Punto de Referencia INS. Sin embargo, esto no es un requisito de montaje. La antena GnssA puede montarse más alejada, siempre y cuando esta distancia se mida y se configure correctamente. Para ello debemos modificar el registro *GNSS Internal A Antenna Offset* (Registro 57), véase Sección A.2.

En nuestro caso particular, como veremos en el Capítulo 4, el módulo INS estará colocado lo más cerca posible del centro de gravedad del UAV, por lo que no aplicaremos desplazamiento en el punto de referencia INS. A su vez, la antena estará dispuesta a varios centímetros de proximidad, por lo que no añadiremos offset alguno en su configuración GNSS.

Estado del INS

La salida *InsStatus* es un registro que proporciona información de estado. Nos centraremos en el campo *Mode*, el cual indica el punto de operación actual del filtro INS y se divide en cuatro posibilidades:

- NotTracking (0): El filtro INS no se ha inicializado y sus salidas son inválidas.
- Aligning (1): El filtro INS se ha inicializado y las mediciones de posición y velocidad son válidas, pero las salidas de actitud aún no lo son.
- Tracking (2): El filtro INS está en modo de seguimiento y todas sus salidas (posición, velocidad y actitud) son válidas.
- GnssLost (3): Se ha producido una pérdida prolongada de GNSS, por lo que las salidas de posición y velocidad del INS ya no son válidas, aunque las salidas de actitud permanecen válidas.

Puesta en marcha

Durante el arranque del VN-200, el sensor progresará a través de diferentes eventos marcados por cambios en el InsStatus (vista disponible en VectorNav Control Center).

- Evento A: Inicialización del Sensor. Al encender o reiniciar el sensor, todos los indicadores en el InsStatus estarán bajos y el sensor estará en modo NotTracking (0). En esta fase, el filtro INS no está inicializado, y la posición y velocidad INS se reportan como ceros con alta incertidumbre.
- Evento B: Corrección GNSS Adquirida. Con buena visibilidad del cielo, el sensor adquiere una corrección GNSS en un periodo de tiempo de 30 a 45 segundos. Una vez adquirido, el indicador GnssFix en el InsStatus se activa, el tiempo GPS se actualiza y las salidas GNSS se vuelven válidas. La posición GNSS actual se usa para calcular el vector de referencia magnética local y el ángulo de declinación, ajustando el rumbo inicial.
- Evento C: Posición y Velocidad INS Válidas. El filtro INS se inicializa y las mediciones de posición, velocidad e incertidumbre INS se vuelven válidas. Este evento se indica al cambiar el sensor al modo Aligning (1) en el InsStatus, ocurriendo dentro de un segundo después del Evento B.
- Evento D: Convergencia del Rumbo GNSS. Este evento indica que el proceso de alineación dinámica ha convergido en una solución de rumbo derivada del GNSS. La alineación dinámica comienza cuando la velocidad supera los 5 m/s por al menos un segundo. En este proceso se correlacionan las aceleraciones horizontales medidas por el acelerómetro y el GNSS para derivar una estimación de rumbo precisa. El rumbo transiciona suavemente de la solución basada en el magnetómetro a la derivada del GNSS.
- Evento E: Actitud Válida. Al alcanzar este evento, el sensor transiciona al modo Tracking (2) en el InsStatus, indicando que el rumbo INS concuerda con el derivado del GNSS y hay consistencia confirmada entre las mediciones INS y GNSS, asegurando que la actitud está dentro de las especificaciones.

Resulta de vital importancia alcanzar el Evento E, y consecuentemente, trabajar en modo Tracking (2) para asegurar unos datos fiables de la solución INS. Para ello, volvemos a recalcar que es necesario someter al sensor a una velocidad que supere los 5 m/s durante un segundo. Tras eso se requiere una aceleración horizontal de cualquier tipo durante el proceso de alineación dinámica. Nuestra experiencia en pruebas corrobora que basta con dar una vuelta en coche durante un par de minutos para lograr que la transición del modo Aligning (1) al Tracking (2) se haga efectiva.

Este procedimiento debe repetirse cada vez que dejemos de alimentar al conjunto VN-200 y antena. En caso de pérdida del modo Tracking (2), no es necesario volver a repetir el procedimiento completo, ni someter al sensor a las condiciones de velocidad descritas. Es suficiente con dar cierto movimiento manualmente durante unos segundos para recuperarlo.

Configuración adicional

Activación del barómetro en la solución INS. Por defecto el filtro EKF no usa las mediciones de presión. Para activarlo es necesario cambiar INS Basic Configuration (Registro 67) a la opción GNSSInsWithPressure. Véase Sección A.3.

2.3 Comparativa y Selección de Antenas GNSS

El sistema VectorNav VN-200 utiliza originalmente una antena GNSS de la marca Tallysman, específicamente el modelo TW2712 [6]. Sin embargo, las dimensiones y el peso de esta antena exceden los requisitos de nuestra aplicación, lo que nos llevó a buscar una alternativa más ligera con características funcionales similares. Mantener una cobertura de satélites comparable a la antena original es un desafío fundamental en la optimización de este proyecto. Si bien consideramos otras marcas, la experiencia con un producto Tallysman nos llevó a priorizar esta opción debido a su similitud con la antena original.

Descripción antena original

La antena TW2712 soporta múltiples constelaciones, incluidas GPS L1, GLONASS G1, Galileo E1 y BeiDou B1. Con un peso de 110 gramos, más los componentes adicionales como el latiguillo de acople, conector y placa metálica para mejorar prestaciones, esta antena resulta inviable para nuestra aplicación en términos de espacio y peso. El LNA (Low Noise Amplifier) de la TW2712 ofrece un bajo nivel de ruido (≤ 1 dB) y una ganancia de 26 dB, características clave que buscamos replicar en la nueva antena.



Figura 2.6 Antena TW2712 original del kit VN-200 Rugged.

Requisitos

Para nuestra aplicación, es esencial que la nueva antena cumpla con los siguientes requisitos:

- **Cobertura de Satélites:** La cobertura debe ser similar a la original, incluyendo al menos las constelaciones GPS y Galileo.
- **Banda Única:** La antena debe ser de banda única (Single Band). Según la documentación de VectorNav, se desaconseja el uso de antenas multifrecuencia con el VN-200 Rugged, ya que otras frecuencias pueden introducir ruido en la banda L1 y degradar el rendimiento GNSS.
- **Peso Ligero:** El peso no debe superar los 25 gramos.
- **Tamaño Reducido y Forma Plana:** Estas características son necesarias para facilitar la integración en el sistema.
- **Ganancia de Señal:** La ganancia de señal debe ser similar a la de la antena original (28 dB).

Alternativas consideradas

Durante el proceso de selección de una nueva antena, se revisaron múltiples datasheets y se mantuvieron conversaciones con distribuidores y soporte de Tallysman. Las siguientes alternativas se barajaron durante este análisis:

Tabla 2.3 Comparativa de Alternativas de Antenas GNSS.

Modelo	Detalles
HC771	Constelaciones: GPS/QZSS-L1, GLONASS-G1, Galileo-E1, BeiDou-B1 Banda: Single Peso: 24 g LNA Gain: 28 dB Observaciones: Tipo helicoidal, forma no conveniente
SSL889XF	Constelaciones: GPS L1/L2, GLONASS G1/G2/G3, Galileo E1/E5b, BeiDou B1/B2b Banda: Dual Peso: 45 g LNA Gain: 28 dB Observaciones: No Single Band y demasiado pesada
TW1430	Constelaciones: GPS L1, GLONASS G1 Banda: Single Peso: 18 g LNA Gain: 32 dB Observaciones: Ligera, no soporta constelación Galileo (a priori)
HC871SXF	Constelaciones: GPS L1/L2, GLONASS G1/G2, Galileo E1, BeiDou B1 Banda: Dual Peso: 23 g LNA Gain: 28 dB Observaciones: Peso aceptable, no adecuado en dimensiones

Selección final

Después de analizar las opciones disponibles, se decidió optar por la antena TW1430 [7]. A pesar de que originalmente no soporta las constelaciones Galileo y BeiDou según las especificaciones del fabricante, su peso de 18 gramos y una ganancia de 32 dB la hacen una opción viable para nuestra aplicación. La antena se adquirió con un conector tipo MMCX Straight Male (plug) y un cable de 300 mm de longitud.

**Figura 2.7** Antena TW1430 seleccionada.

Prestaciones

Se realizaron pruebas con la nueva antena, obteniendo prestaciones comparables a las logradas con la antena original más grande. Sorprendentemente, la antena TW1430 logró captar satélites de la constelación Galileo, lo que no estaba previsto en las especificaciones del fabricante. Este comportamiento se debe al solapamiento de las bandas de Galileo E1 con GPS L1, lo cual se observa en el esquema de bandas de navegación de la Figura 2.8.

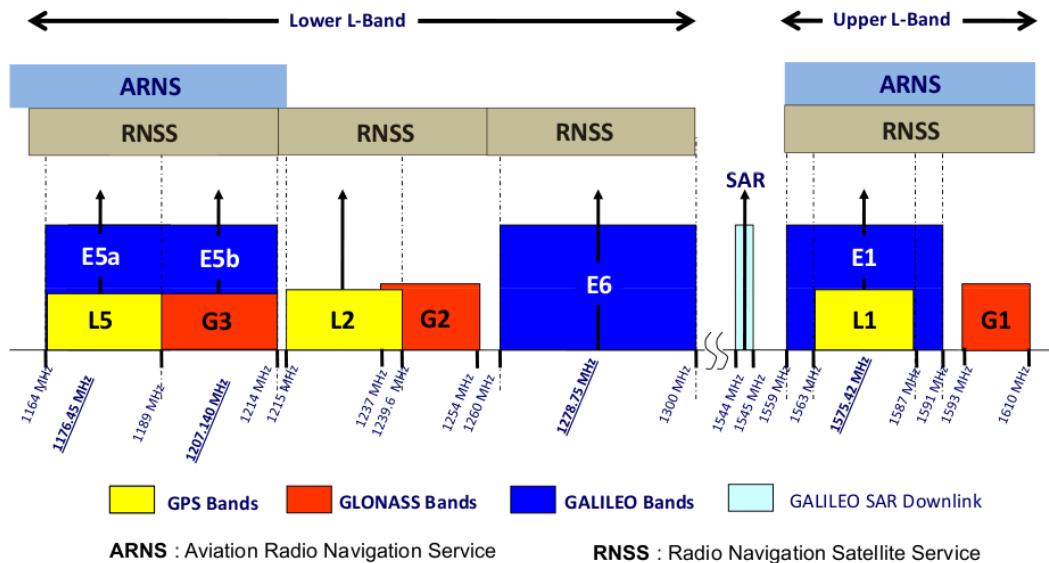


Figura 2.8 Bandas de frecuencia de navegación GPS, GLONASS y Galileo..

Configuración GNSS

En la configuración del sistema, se activaron las constelaciones GPS, Galileo y GLONASS para maximizar el número de satélites disponibles.

El campo *SBAS Mode* se configuró en valor 2 (*Integrity*). Además, se ajustaron parámetros como el número máximo de satélites utilizados por el EKF para la solución INS, la potencia mínima y el ángulo de inclinación necesarios para el uso. Se estableció que el número máximo de satélites considerados para la estimación sería 16, utilizando aquellos con la señal más potente. La elevación mínima de los satélites se ha dejado a 5 grados por encima del horizonte. La potencia mínima requerida para tener en cuenta un satélite fue de 10 dB.

Todos estos detalles son configurables en el registro *GNSS System Configuration* (Registro 99), como puede verse en la Figura 2.9. Más información en Sección A.5.

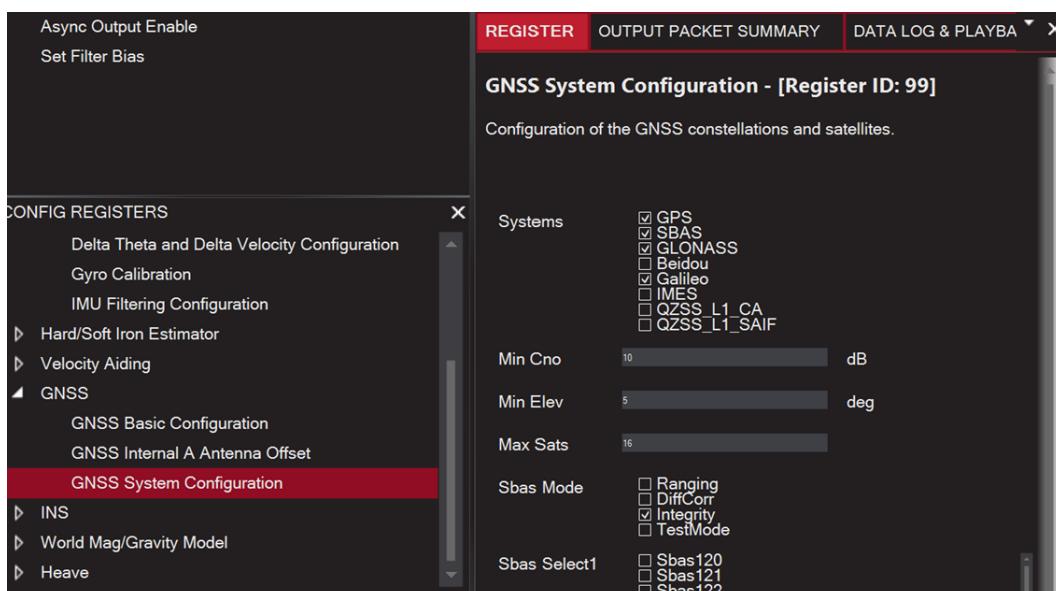


Figura 2.9 Configuración interna de ajustes GNSS en software Control Center.

2.4 Comunicación externa

El VectorNav VN-200 ofrece diversas opciones de comunicación para interactuar con el dispositivo y extraer datos en tiempo real. En esta sección, se presenta un resumen de las interfaces y protocolos disponibles, proporcionando una visión general que ayudará a contextualizar el enfoque elegido.

Interfaces de Comunicación

El VN-200 cuenta con dos interfaces de comunicación separadas en dos puertos seriales y un bus SPI (Serial Peripheral Interface). En nuestro caso, nos centraremos en el puerto serie ya que la versión Rugged del sensor no soporta con la interfaz SPI. La versión Rugged incluye un conversor de nivel TTL a RS-232 integrado, por lo que en el conector de 10 pines ofrece un puerto serie con niveles de voltaje RS-232 (Serial 1), mientras que el otro puerto serie (Serial 2) permanece con niveles lógicos TTL de 3V. Véase Tabla 2.2. En nuestra aplicación trabajaremos con lógica TTL en cuanto a la integración con el computador de a bordo.

Protocolos de Comunicación

El VN-200 emplea un protocolo de comunicación basado en comandos para las interfaces seriales. Este protocolo se presenta en dos formas principales: ASCII y binario, cada uno diseñado para diferentes propósitos. Auxiliarmente, también incluye una línea de comandos opcional para configuración avanzada y diagnósticos, diseñada para uso manual a través de un terminal serial.

Protocolo Serial ASCII

En la interfaz serial, se utiliza un protocolo ASCII completo que soporta todos los comandos y la consulta de registros. Este protocolo es muy similar al ampliamente utilizado NMEA 0183, que es compatible con la mayoría de los receptores GNSS, y se compone de parámetros delimitados por comas en texto legible.

Protocolo Serial Binario

La interfaz serial también soporta la transmisión continua de mediciones de sensores a velocidades fijas mediante paquetes binarios de salida configurables por el usuario. Estos paquetes binarios ofrecen una forma eficiente de transmitir mediciones de sensores a alta velocidad desde el dispositivo, minimizando tanto el ancho de banda requerido y la sobrecarga necesaria para procesar las mediciones en el host.

El VN-200 permite configurar hasta 3 mensajes binarios por el usuario. Cada mensaje puede adaptarse para incluir cualquier tipo de medición disponible de los subsistemas IMU, NavState, NavFilter o GNSS. Se pueden obtener todas las mediciones en tiempo real, tanto las directamente capturadas por los sensores como las estimadas a partir de los filtros internos. El dispositivo está diseñado para enviar cada mensaje de forma asíncrona a una tasa fija, determinada por un divisor de la tasa de muestreo interna de la IMU (IMU Rate). La configuración de estos 3 mensajes de salida se realiza mediante los registros *User Output Configuration* (Registros 75-77), uno para cada mensaje. Véase Sección A.4.

Este protocolo será el usado en adelante por las ventajas comentadas.

2.5 Requisitos del sistema de navegación

Según los objetivos del proyecto, el sensor VN-200 se utilizará para proporcionar, a frecuencia de 89 Hz, los datos de la solución INS necesarios durante el vuelo. El listado de los datos requeridos para nuestra aplicación se detalla a continuación:

- Time Startup: El tiempo del sistema desde el arranque medido en nano segundos.
- YawPitchRoll: Los ángulos de guiñada, cabeceo y balanceo estimados, medidos en grados. La actitud se da como una secuencia de ángulos de Euler que describe el marco del cuerpo con respecto al marco local North East Down (NED).
- Position: La posición estimada como latitud, longitud (grados) y altitud en metros.
- Velocity: La velocidad estimada en el marco NED, expresada en m/s.

El dispositivo puede configurarse para emitir asíncronamente cada mensaje binario, formado por todos los datos de la lista, a una frecuencia fija, basada en un divisor de la frecuencia de muestreo interna de la IMU (800 Hz).

Debido a las limitaciones en el ancho de banda de la comunicación, la frecuencia máxima a la que podemos sacar la información es de 89 Hz (9 divisiones) lo cual entra dentro de nuestras especificaciones.

En caso de necesitar aumentar la frecuencia habría que bajar la cantidad de datos seleccionados o dividirlo en varios de los 3 mensajes de salida binarios independientes configurables por el usuario.

En la Figura 2.10 pueden verse los detalles descritos, proporcionados por la herramienta *Binary Output Configuration Window* dentro de *Control Center*.

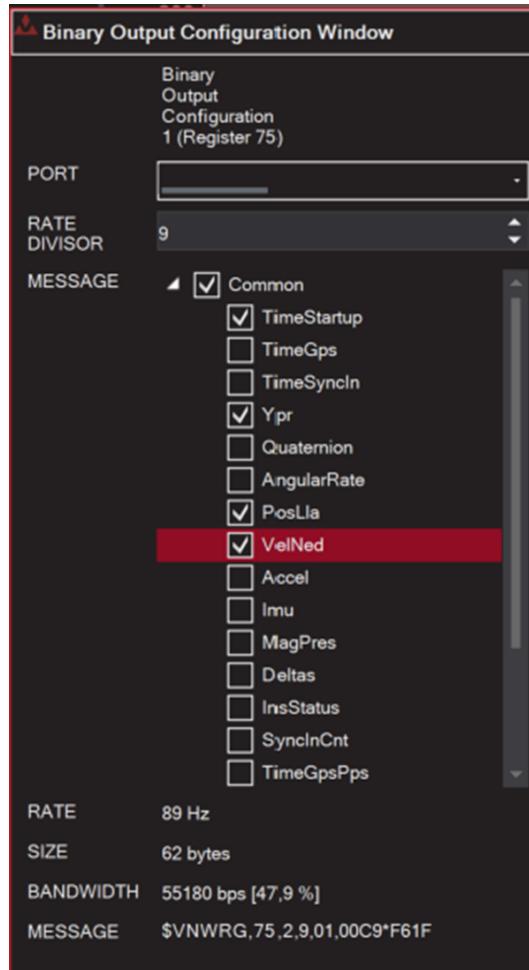


Figura 2.10 Limitación en el ancho de banda y configuración del mensaje binario.

3 Descripción y Programa del Computador de a Bordo: Raspberry Pi Zero W

Esfuérzese por la perfección en todo lo que hace. Toma lo mejor que existe y hazlo mejor. Cuando no exista, diseñelo.

SIR HENRY ROYCE

El computador de a bordo seleccionado para el proyecto ha sido la placa Raspberry Pi Zero W. Nos centraremos en las características de comunicación, recursos y peso que ofrece. Describiremos el proceso de configuración de la microcomputadora Raspberry Pi y los programas instalados en el PC usado como estación de tierra. Además, se explicará en detalle el programa creado para la comunicación con VectorNav VN-200 para la grabación de datos.

3.1 Presentación de Raspberry Pi Zero W

La Raspberry Pi Zero W [8] es un microcomputador de bajo coste y alta eficiencia desarrollado por la Fundación Raspberry Pi. Integra conectividad inalámbrica completa con soporte para Wi-Fi y Bluetooth, lo que la convierte en una opción versátil para una amplia gama de aplicaciones. A pesar de su tamaño compacto, el Raspberry Pi Zero W es capaz de ejecutar un sistema operativo completo, como Raspberry Pi OS. Suele emplearse en proyectos el ámbito de Internet de las Cosas (IoT) y en sistemas embebidos.

Destacando sus características físicas, la Raspberry Pi Zero W se distingue por su bajo peso, dimensiones reducidas y delgadez, lo que facilita su integración en proyectos donde el espacio es un factor crítico. Con un peso de solo 10 gramos y dimensiones de 65 x 30 x 1.5 mm, es una de las opciones más compactas del mercado actual. A pesar de su tamaño diminuto, no es un microcontrolador sino un microprocesador con un rendimiento considerable. Este diseño permite ejecutar un sistema operativo completo, soportar múltiples aplicaciones simultáneamente y manejar la placa de forma remota mediante protocolos como SSH y VNC. Por todo lo descrito se hace conveniente su uso en nuestro sistema.

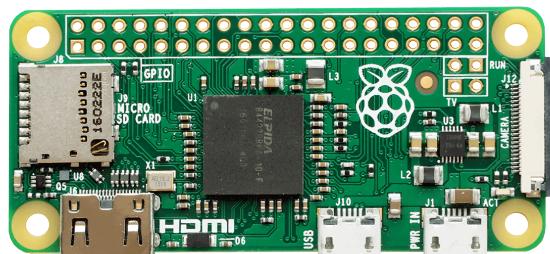


Figura 3.1 Raspberry Pi Zero W.

3.1.1 Características técnicas

Presentamos las principales características generales y en cuanto a prestaciones que nos ofrece la placa. Esta información queda recogida en Tabla 3.1 y Tabla 3.2.

Tabla 3.1 Prestaciones de la Raspberry Pi Zero W.

Característica	Detalles
Conectividad inalámbrica	802.11 b/g/n wireless LAN (WiFi)
Bluetooth	Bluetooth 4.1 & BLE
Procesador	BCM2835 1GHz, single-core CPU
Memoria RAM	512MB
Almacenamiento	SD-card 32 GB
Puerto de video	Mini HDMI
Conector para cámara	CSI camera connector

Tabla 3.2 Características estructurales y eléctricas de la Raspberry Pi Zero W.

Característica	Valor
Conector de pines	40-pin header
Peso	10 gramos
Tamaño	65 x 30 x 1.5 mm
Alimentación	Micro USB power (5V DC)
Consumo energético	240 - 300 mA @ 5V 1.5 W (WiFi ON)

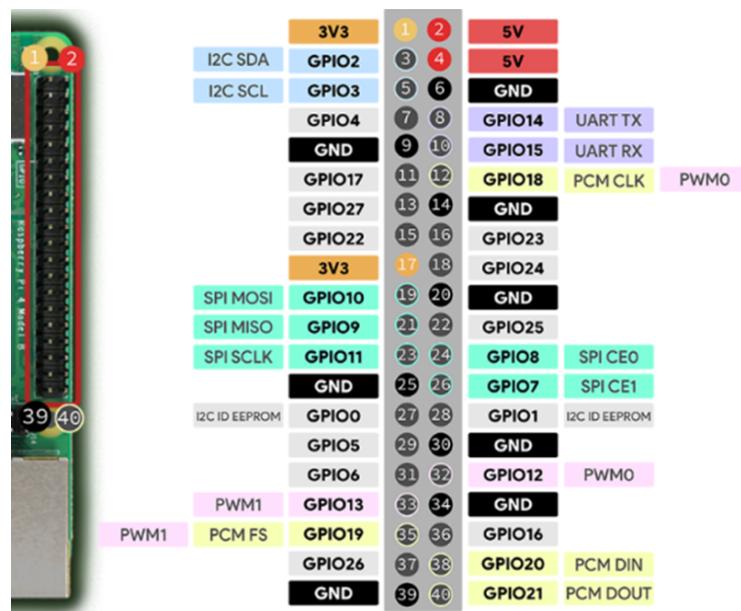


Figura 3.2 Asignaciones de pines GPIO Raspberry Pi Zero W.

El GPIO (General Purpose Input/Output) de la Raspberry Pi Zero W ofrece un conjunto versátil de posibilidades para la interacción con hardware externo.

Tiene un encabezado de 40 pines, de los cuales 26 son pines GPIO configurables que permiten tanto entradas como salidas digitales. Aunque los pines GPIO son digitales por naturaleza, es posible medir voltajes analógicos utilizando convertidores ADC, ya que la Raspberry Pi no tiene entradas analógicas integradas.

Además, los pines GPIO pueden generar señales PWM (Pulse Width Modulation), lo cual es útil para aplicaciones como el control de velocidad de motores y servos.

En términos de comunicación, el GPIO de la Raspberry Pi Zero W soporta varios protocolos, incluyendo I2C (Inter-Integrated Circuit) para conectar múltiples dispositivos con solo dos pines de comunicación, SPI (Serial Peripheral Interface) para conexiones rápidas con dispositivos como memorias y sensores, y UART (Universal Asynchronous Receiver/Transmitter) para la comunicación serial con otros microcontroladores o dispositivos.

Nos centraremos en la comunicación vía puerto serie con VN-200 por lo que emplearemos la UART de la placa. En la Figura 3.3 puede verse cuáles son los pines asociados (TX y RX), 8 y 10 (físicos) o 14 y 15 (GPIO) respectivamente.

3.1.2 Configuración y ajustes iniciales

El proceso de configuración inicial de una placa con procesador, como la Raspberry Pi Zero W, implica una serie de pasos básicos que permiten preparar el dispositivo para su uso. Si bien es cierto que pueden usarse periféricos externos (teclado, ratón y pantalla) durante la configuración inicial, también es posible prescindir de ellos, permitiendo que la placa esté lista para su uso remoto desde el primer momento.

El primer paso es instalar un sistema operativo compatible en la tarjeta SD, utilizando una imagen del sistema. Este procedimiento puede realizarse de forma sencilla con el programa Raspberry Pi Imager [9]. En nuestro caso hemos optado por la instalación Raspberry Pi OS - 32 bits (previamente llamado Raspbian). Este sistema operativo está diseñado específicamente para la línea de microcomputadoras Raspberry Pi. Se trata de una distribución de Linux basada en Debian, una de las distribuciones más populares y estables de Linux. Está optimizado para nuestro hardware específico, aprovechando al máximo los recursos de la placas para ofrecer un entorno más eficiente.

Para facilitar un acceso remoto desde el primer momento, se puede realizar una preconfiguración del Wi-Fi y del usuario directamente en la tarjeta SD antes de insertarla por primera vez. Es recomendable habilitar previamente SSH y VNC, lo que permite no solo acceder a la terminal de la Raspberry Pi de manera remota, sino también controlar su entorno de escritorio gráfico.

Además, se habilitó la interfaz UART para permitir la comunicación en serie con otros dispositivos, lo cual resulta fundamental en nuestro caso. Esta configuración se realizó a través de la interfaz gráfica que se despliega al ejecutar el comando '`sudo raspi-config`' en una terminal, utilizando una conexión remota SSH con la placa.

3.1.3 Herramientas software en la estación de tierra

Una vez preparado el computador de a bordo, necesitaremos contar con un conjunto de herramientas y programas software que nos permitan la conexión al mismo y uso de forma remota, facilitándonos el proceso de desarrollo y pruebas. Todas ellas estarán instaladas en el equipo que llamaremos estación de tierra o Ground Control.

A continuación se ofrece la lista de alternativas con las que hemos trabajado:

- [TabbyTerminal](#) [10]: Aplicación de terminal multiplataforma altamente personalizable, diseñada para ofrecer una experiencia flexible y potente en la gestión de shells locales, conexiones serie, SSH, y Telnet. A diferencia de muchas otras aplicaciones de terminal, Tabby permite a los usuarios personalizar prácticamente todos los aspectos de su interfaz y funcionamiento, desde la apariencia hasta el comportamiento de los atajos de teclado y la disposición de las pestanas. Además, soporta la conexión simultánea a múltiples servidores, lo que lo hace ideal para gestionar varios entornos de trabajo de forma simultánea y centralizada. También tiene disponibles servicios de SFTP (SSH File Transfer Protocol), por lo que pueden intercambiarse ficheros entre máquina local y servidor.
- [Bitvise SSH Client](#) [11]: Otra alternativa software conocida para el acceso a servidores remotos.

- RealVNC Viewer [12]: Aplicación que permite a los usuarios conectarse y controlar ordenadores remotos a través de la red utilizando el protocolo VNC (Virtual Network Computing). Su principal ventaja es la capacidad de mostrar la interfaz gráfica de usuario (GUI) del servidor remoto, facilitando la administración y el uso de aplicaciones gráficas, aunque esta funcionalidad es opcional si se prefiere trabajar con la terminal directamente.
- Virtual Studio Code [13]: Editor de código fuente multiplataforma altamente versátil y personalizable desarrollado por Microsoft. Ofrece características avanzadas como resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización, así como soporte integrado para la depuración y control de versiones con GIT. Su principal ventaja radica en la capacidad de utilizar múltiples extensiones, permitiendo trabajar con diversos lenguajes de programación simultáneamente.

En resumen, utilizamos Tabby para establecer conexiones SSH con la Raspberry Pi, facilitando la transferencia de archivos, como la carga y modificación de programas, y la descarga de datos de telemetría. Para la programación del código principal, empleamos Visual Studio Code, mientras que la representación gráfica de los resultados se realiza utilizando MATLAB.

3.2 Programa principal

En esta sección daremos una explicación detallada del programa principal desarrollado en C++ para la grabación de datos de telemetría recibidos desde el sensor VN-200 hacia una placa Raspberry Pi Zero W a través de la interfaz serial usando los pines físicos de la UART de la placa. El programa se encarga de registrar los datos proporcionados por el sensor VN-200, generando un fichero en formato .csv. Para la implementación del mismo, se ha utilizado la biblioteca *vnproglis*, introducida en la Sección 2.1.4. Se ha tomado como punto de partida el ejemplo disponible para C++ "getting_started.cpp", que cuenta con material útil sobre los diversos tipos de comunicación vistos en la Sección 2.4. La explicación detallada acerca de los datos que se manejan en este programa puede consultarse en la Sección 2.5.

Pasamos a comentar las principales funcionalidades del código desarrollado.

1. Configuración y conexión del sensor:

- El sensor VN-200 se conecta a través del puerto serie */dev/ttyS0* con una velocidad de transmisión de 115200 baudios.
- Se configura el registro de salida binaria 1 para obtener los datos específicos: marca temporal, actitud, posición y velocidad. Como información adicional registramos también el estado del INS (del cual extraemos los 2 bits menos significativos correspondientes al modo actual del filtro INS (véase Sección 2.2), usado para conocer la situación del dispositivo de cara al proceso de arranque).

2. Manejo de fichero CSV:

- El programa graba los datos recibidos en un archivo CSV (data.csv), utilizando la clase CsvFileHandler, que gestiona la creación, apertura, escritura y cierre del archivo de manera segura.

3. Recepción y procesamiento de paquetes:

- El código emplea un manejador de paquetes asíncronos para procesar los datos en tiempo real.
- Solo se procesan los paquetes binarios que sean compatibles al formato configurado, de los cuales se extraen y registran datos específicos.

4. Monitoreo de Modo INS:

- El programa monitorea las transiciones del modo INS (de 1 a 2 y viceversa), mostrando mensajes en la consola cuando ocurre un cambio. Esto nos ayuda a saber cuando el sistema completo se encuentra listo para hacer experimentos.

5. Desconexión:

- Al finalizar el programa, se asegura de desconectar el sensor y desregistrar el manejador de paquetes, cerrando correctamente todas las conexiones y archivos. El programa finaliza al interrumpirlo por terminal al enviar la señal SIGINT (combinación teclas Ctrl + C).

```

1 Timestamp,Yaw,Pitch,Roll,Lat,Lon,Alt,VelN,VelE,VelD,InsMode
2 4225681257000,-81.171,-0.387,-1.915,37.4218,-5.9986964,58.722,-0.319,0.229,-0.044,2
3 4225692507000,-81.118,-0.024,-2.011,37.4217999,-5.9986963,58.722,-0.331,0.248,-0.034,2
4 4225703757000,-81.155,0.279,-2.046,37.421801,-5.9986961,58.729,-0.298,0.245,-0.062,2
5 4225715007000,-81.227,0.341,-2.021,37.421801,-5.9986961,58.73,-0.3,0.222,-0.049,2
6 4225726257000,-81.258,0.214,-2.091,37.4218009,-5.998696,58.73,-0.295,0.204,-0.041,2
7 4225737507000,-81.282,-0.038,-2.165,37.4218009,-5.998696,58.731,-0.298,0.2,-0.053,2
8 4225748757000,-81.229,-0.34,-2.248,37.4218009,-5.998696,58.728,-0.302,0.206,-0.06,2
9 4225760007000,-81.125,-0.508,-2.373,37.4218008,-5.998696,58.728,-0.309,0.212,-0.054,2
10 4225771257000,-81.062,-0.601,-2.577,37.4218008,-5.9986959,58.729,-0.309,0.218,-0.054,2
11 4225782507000,-81.044,-0.557,-2.765,37.4218008,-5.9986959,58.73,-0.313,0.228,-0.049,2
12 4225793757000,-81.095,-0.418,-2.968,37.4218008,-5.9986959,58.73,-0.306,0.229,-0.047,2
13 4225805007000,-81.184,-0.301,-3.045,37.4218007,-5.9986958,58.731,-0.302,0.231,-0.048,2
14 4225816257000,-81.272,-0.161,-3.086,37.4218007,-5.9986958,58.731,-0.301,0.23,-0.047,2
15 4225827507000,-81.319,-0.069,-3.123,37.4218007,-5.9986958,58.732,-0.303,0.229,-0.049,2
16 4225838757000,-81.345,-0.038,-3.184,37.4218006,-5.9986957,58.732,-0.303,0.222,-0.051,2

```

Figura 3.3 Ejemplo de fichero .csv generado por programa principal.

El código completo puede consultarse en el Apéndice B.

La sección del código resaltada en amarillo contiene las directivas de preprocesador y las declaraciones de espacios de nombres que son necesarias para configurar el entorno de programación. Incluye las librerías estándar de C++ como `<iostream>`, `<fstream>`, `<iomanip>`, y `<string>`, que se utilizan para la entrada/salida de datos, manejo de archivos, manipulación de formato de datos y gestión de cadenas de caracteres, respectivamente. Además, se incorpora el fichero de cabeceras "vn/sensors.h", específico para facilitar el uso con los sensores de VectorNav. Esto permite que el programa se comunique con el sensor VN-200 de forma eficiente. Los espacios de nombres std y varios propios de VectorNav (vn::math, vn::sensors, vn::protocol::uart) se utilizan para simplificar el código y evitar la necesidad de prefijar funciones con sus espacios de nombres completos.

En la sección en verde, se define la clase CsvFileHandler, que es responsable de la creación y manejo de archivos CSV donde se almacenarán los datos de telemetría. Esta clase tiene un constructor que abre el archivo CSV y escribe la cabecera con los nombres de los campos de datos (como Timestamp, Yaw, Pitch, etc.). Si el archivo no puede ser abierto, se muestra un mensaje de error y el programa termina. El destructor de la clase cierra el archivo cuando la instancia de la clase es destruida, asegurando una buena gestión de recursos. También incluye el método writeData, que se encarga de escribir los datos de telemetría en el archivo CSV, con un formato adecuado y precisión específica para cada tipo de dato.

La sección en rojo abarca la función principal del programa. Aquí se definen los parámetros básicos para la conexión con el sensor VN-200, como el puerto serial (/dev/ttyS0) y la velocidad de transmisión en baudios (115200). Luego, se instancia un objeto VnSensor y se establece la conexión con el sensor. El programa configura el registro de salida binaria del sensor para que incluya los grupos de datos requeridos. A continuación, se inicializa una instancia de CsvFileHandler para gestionar el archivo CSV y se registra la función BinaryAsyncMessageReceived como manejador para los paquetes de datos recibidos. El programa entra en un bucle infinito para mantenerse ejecutando, esperando y procesando los datos del sensor de forma asíncrona. Finalmente, hay una limpieza del programa que elimina el registro del manejador de paquetes y desconecta el sensor.

La sección final en azul define la implementación de la función BinaryAsyncMessageReceived. Esta función maneja los paquetes de mensajes binarios recibidos del sensor de forma asíncrona, extrayendo los distintos datos que posteriormente se escriben en una línea en el archivo CSV a través del objeto CsvFileHandler. Además, la función monitorea cambios en el modo del INS y genera mensajes en la consola que permite al usuario obtener información en tiempo real sobre el estado del sistema mientras el programa está ejecutándose.

4 Integración Hardware

Fabricar es algo más que juntar piezas. Es concebir ideas, probar principios y perfeccionar la ingeniería, así como el montaje final.

JAMES DYSON

A bordaremos en detalle el proceso de integración y validación de los componentes del sistema, centrándonos en la incorporación de los nuevos elementos y su ensamblaje, administrando los espacios disponibles dentro del chasis seleccionado.

Finalmente, se presentarán las pruebas de funcionamiento realizadas antes del vuelo, incluyendo la estimación de tiempos de batería y recorridos con el sistema montado en un coche. Estos ensayos son fundamentales para asegurar el correcto funcionamiento del conjunto, permitiendo el análisis de los resultados y el postprocesado de datos antes de proceder al vuelo real.

4.1 Conexión y comunicaciones de componentes

El sistema completo se divide en dos partes: la original de fábrica de la plataforma GoGoBird y la nueva electrónica que hemos añadido. Aunque nos enfocaremos en la sección desarrollada por nosotros, es esencial adaptar nuestra tecnología para integrarse de manera eficiente con la electrónica y el chasis originales. Para ilustrar cómo luce el producto tras el desmontaje, previo a realización de modificaciones, se puede consultar la Figura 4.1.

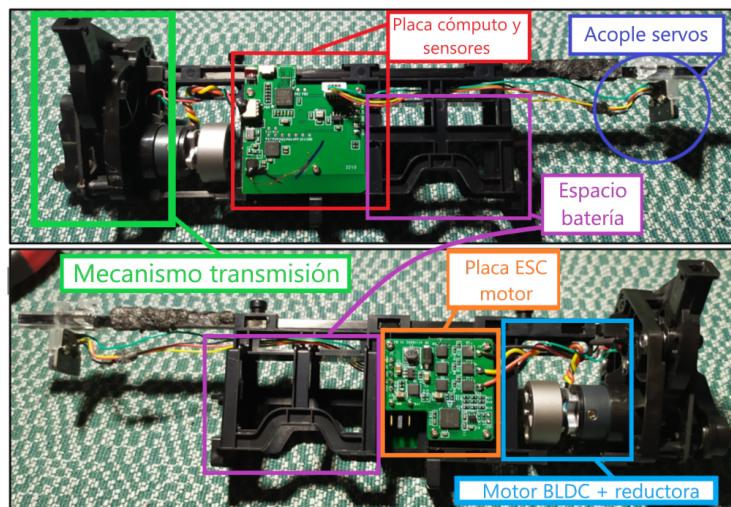


Figura 4.1 Sistema electrónico interno de GoGobird.

En resumen, nuestro sistema de adquisición de datos se compone del módulo INS/GNSS VN-200, la antena TW2710 y la placa Raspberry Pi Zero W.

La comunicación se establece a través de puerto serie, y todo el conjunto es alimentado por la misma fuente que nutre la electrónica original: una batería de ion-litio de 2 celdas y 520 mAh. Mientras que la Raspberry Pi requiere una entrada de 5V, el módulo de VectorNav admite un rango más amplio de voltaje.

Hemos decidido alimentar el conjunto VN-200 y antena a un voltaje más alto para mejorar el rendimiento de la antena, lo cual es conveniente para la aplicación. Aprovechamos tanto la salida directa de 8.4V (cuando la batería está completamente cargada) para alimentar el módulo VN-200 y la antena, como la salida de 5V proporcionada por los reguladores de la placa de potencia para la Raspberry Pi. Para alimentar la placa de forma segura sin usar el conector micro USB, y así evitar peso añadido, se ha considerado soldar en los puntos de test de la PCB bajo del puerto micro USB, ya que estas zonas están protegidas y ofrecen seguridad frente a posibles problemas eléctricos. Alimentar a través del GPIO carece de protección, lo que podría dañar la placa en caso de fallos. Esta configuración permite separar las alimentaciones, manteniendo una referencia común, lo cual puede verse en detalle en la Figura 4.2.

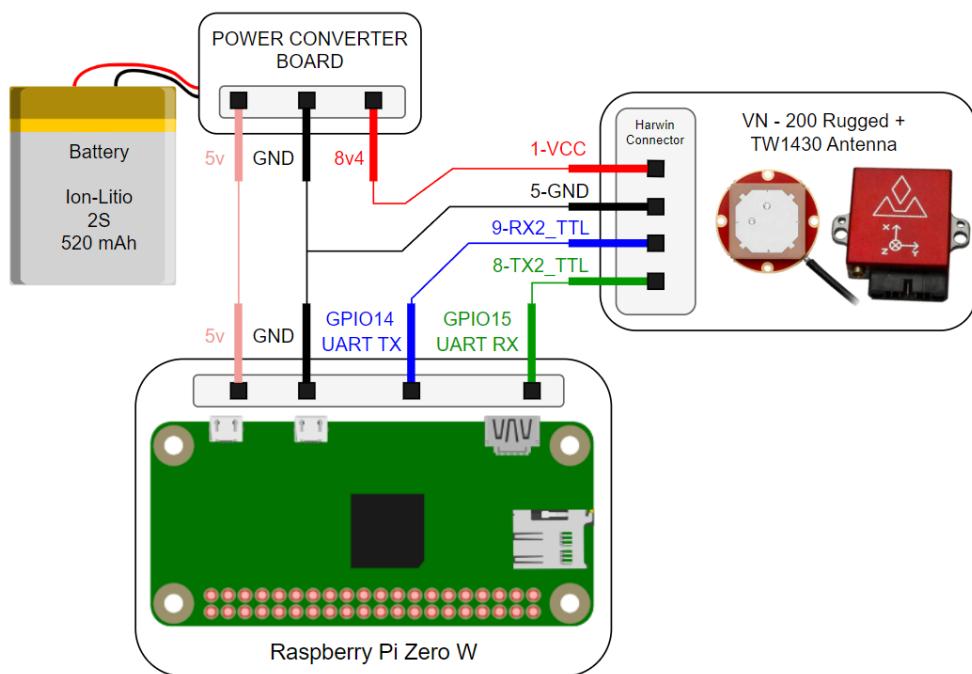


Figura 4.2 Esquema del conexionado eléctrico.

Por otro lado, como se anticipó en el Capítulo 3, el equipo necesario para realizar la grabación de telemetría durante los experimentos se compone del UAV, la estación de control en tierra y un router, tal como se muestra en la Figura 4.3.

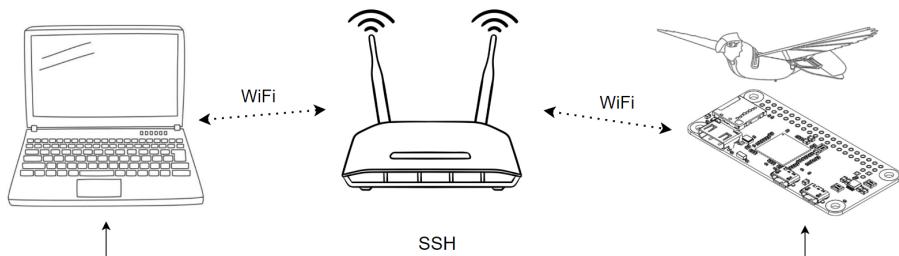


Figura 4.3 Esquema conceptual de equipo y conectividad SHH vía WiFi.

4.2 Concepción y ensamblaje

La integración del nuevo sistema electrónico sobre el diseño original, respetando las restricciones de espacio y asegurando una compensación de masas para no comprometer la estabilidad del orníptero, ha representado un verdadero desafío. Tras un profundo estudio de alternativas, optamos por la configuración que se muestra en el modelo 3D del UAV en la Figura 4.4.

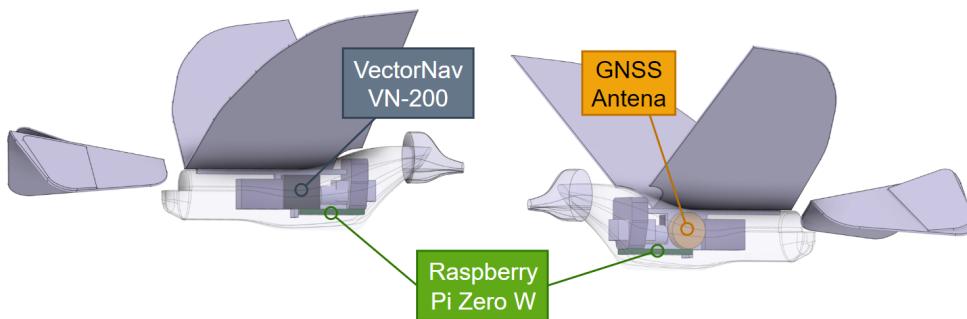


Figura 4.4 Distribución de componentes en modelo 3D.

Para facilitar el acceso a los pines del VN-200 y asegurar una sujeción de garantías, se fabricó un conector completamente personalizado, incluyendo solo los pines necesarios y eliminando los demás para reducir el peso. El acabado de este conector se puede apreciar en la Figura 4.5.

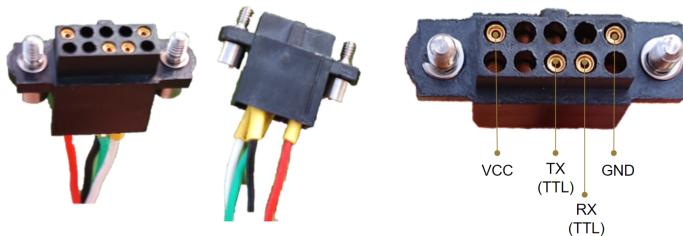


Figura 4.5 Conector Harwin fabricado. Conexión segura con VN-200.

El resto de conexiones se realizaron mediante soldadura directa o utilizando conectores tipo Dupont. Para fijar los componentes al chasis, se moldeó la estructura de corcho con calor, creando una forma precisa que permite un ajuste a presión. Adicionalmente, se añadió velcro para asegurar la sujeción. Por último, se utilizaron bandas elásticas ligeras para mantener cerradas las distintas partes. Los detalles comentados se aprecian en las Figura 4.6 y Figura 4.7.

El resultado final ha supuesto un incremento de 50 gramos en el peso total del orníptero, que asciende a los 220 gramos. En la Tabla 4.1 se muestra el desglose de los pesos.

Tabla 4.1 Aportaciones de componentes al peso total.

Componente	Peso
VN-200 Rugged	16 g
Antena GNSS	18 g
Placa Raspberry Pi	10 g
Conector Harwin	2 g
Cableado Resto de conectores	2 g
Velcro y gomas elásticas	2 g

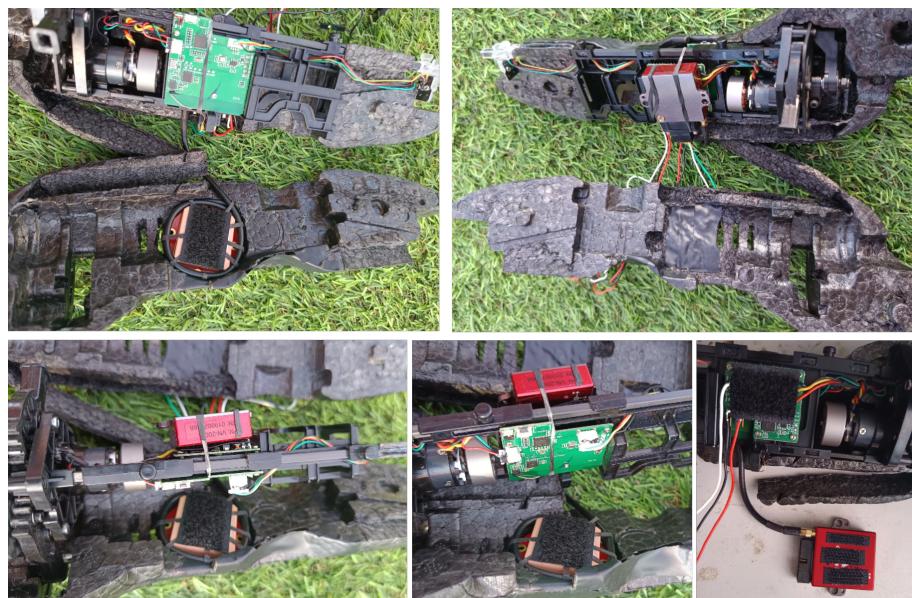


Figura 4.6 Proceso de integración física.



Figura 4.7 Detalles de montaje y acabado final.

4.3 Verificación funcional

El objetivo de esta sección es presentar las pruebas realizadas antes del vuelo, llevadas a cabo en tierra para asegurar que el sistema funcionara conforme a lo esperado. Estas pruebas preliminares fueron esenciales para validar el correcto funcionamiento del sistema y detectar posibles errores.

Calibración VN-200

Tras la disposición del sensor en el cuerpo del robot aéreo lo primero es realizar el ajuste apropiado de la matriz de rotación para que las medidas del sensor queden referidas en el body frame, tal y como se detalla en Sección 2.2. En la Figura 4.8 se muestra una prueba de este hecho.

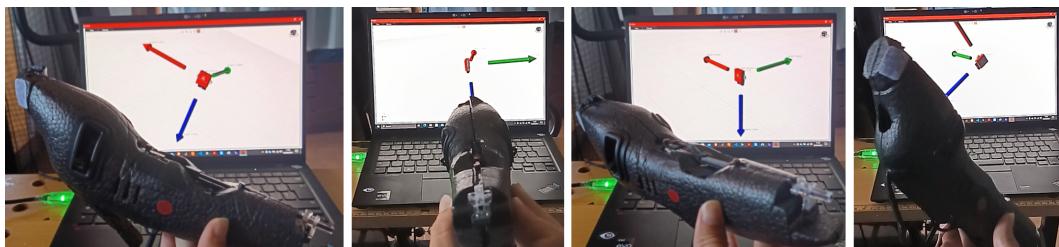


Figura 4.8 Calibración de actitud en el VN-200.

Estimación de tiempo de duración de la batería

Para estimar la duración de la batería que alimenta todo el sistema bajo condiciones de funcionamiento real, se ha simulado el sistema en operación completa: actuadores en funcionamiento, sensor INS activo, computador de a bordo registrando telemetría, envío de órdenes por RF para el movimiento, electrónica original en operación, y antena captando señales de satélites.

Los resultados obtenidos indican una duración de 26 minutos sin las alas acopladas al mecanismo de transmisión, y 7 minutos con ellas. Esto demuestra que el esfuerzo de mover aire con las alas incrementa significativamente el consumo del motor, afectando considerablemente la autonomía.

Prevemos que, durante nuestros experimentos, la batería tendrá una duración superior a 7 minutos, dado que no se utilizarán los motores de manera continua entre pruebas. Este tiempo es suficiente y satisface nuestras necesidades, por lo que no es necesario incorporar una batería de mayor capacidad que la original, opción que se consideró inicialmente.

Validación de distancias en comunicación

A continuación, se llevaron a cabo experimentos en exterior para valuar la distancia de comunicación entre la Raspberry Pi y el PC, asegurando la integridad de la conexión SSH y la correcta grabación de los datos de telemetría, así como la verificación de la comunicación por radiofrecuencia para el control teleoperado del UAV.

El experimento se llevó a cabo en un campo abierto, permitiendo probar la efectividad y robustez de la comunicación en un entorno libre de interferencias y obstáculos. Los resultados obtenidos fueron satisfactorios:

- Distancia de comunicación WiFi: La conexión WiFi entre la Raspberry Pi y el PC superó los 300 metros. Esto asegura una cobertura adecuada para la transmisión de datos y el manejo del sistema operativo del computador embarcado.
- Comunicación RF: La conexión RF con el mando de control se mantuvo estable y efectiva a lo largo de toda la distancia cubierta. Esta prueba también sirvió para confirmar que la antena RF y las placas internas primarias no sufrieron daños durante el montaje.

La telemetría se registró de manera consistente y sin fallos durante todo el experimento, validando que la configuración es adecuada para operaciones de campo.

Recorridos en coche

En este experimento, todo el sistema fue instalado en un vehículo y se llevaron a cabo diversos recorridos, repitiendo cada uno varias veces para verificar que las trayectorias capturadas por el sensor se grabaran correctamente. Con estas pruebas se quiere verificar que la secuencia de arranque y los modos del INS funcionen correctamente, tal como se detalla en la Sección 2.2.

Se han llevado a cabo 4 rutas distintas:

- Ruta 1: Recorrido abierto (≈ 2 km) por caminos rurales.
- Ruta 2: Recorrido cerrado (≈ 1.5 km) urbano con distintas cuestas para registrar variaciones significativas de altura (≈ 35 m).
- Ruta 3: Recorrido cerrado (≈ 2.7 km) urbano - rural.
- Ruta 4: Recorrido en carretera recta (≈ 900 m).

Tras postprocesar los datos obtenidos y analizar los resultados vemos como las rutas grabadas en cada iteración son consistentes. Todas las trayectorias se han referenciado al punto inicial mediante una transformación de coordenadas LLA (latitud, longitud, altitud) a XYZ, lo que permite obtener una representación espacial de la ruta seguida. Esta transformación se ha realizado utilizando el modelo WGS84 de la Tierra, calculando las coordenadas X e Y en un plano y tomando como referencia el punto de partida. La altitud Z se ajusta para ser relativa al valor inicial, lo que proporciona una descripción tridimensional del recorrido.

Además, se ha hecho una comparativa entre los recorridos registrados por el sensor y los planificados con las herramientas disponibles en Google Earth, lo cual puede verse en las distintas figuras que se muestran.

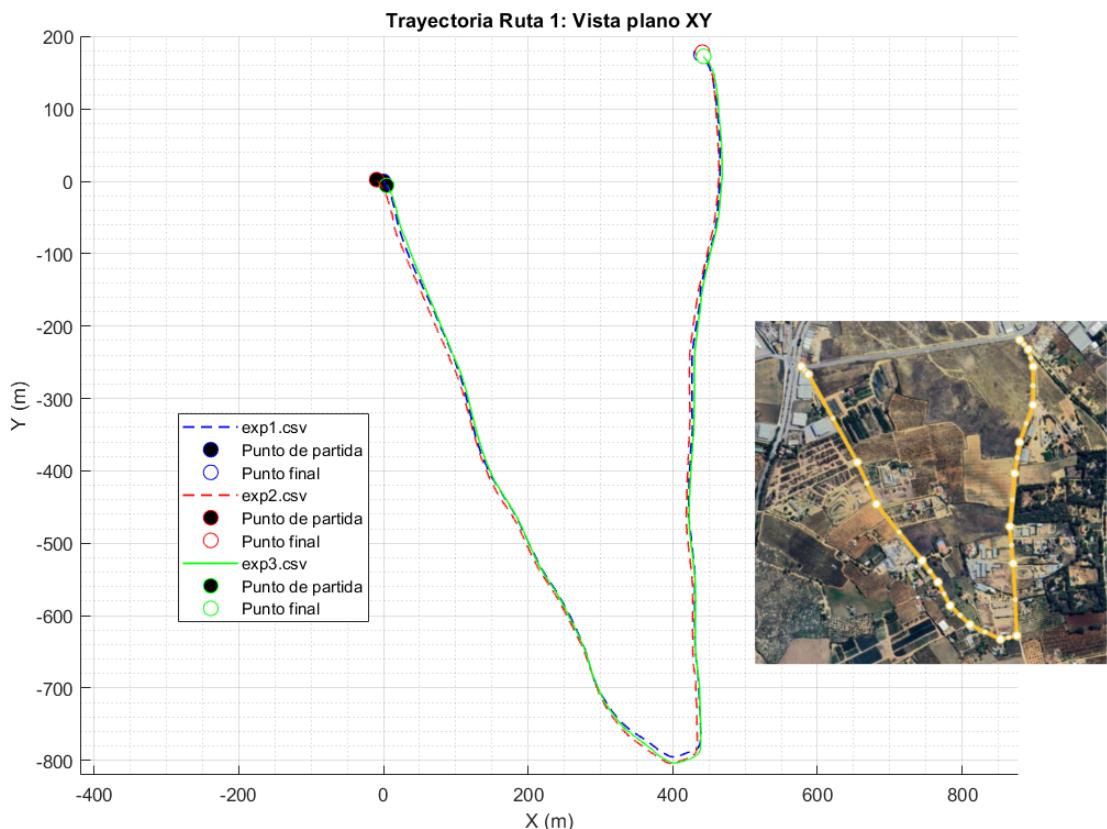


Figura 4.9 Recorrido en coche: Ruta 1.

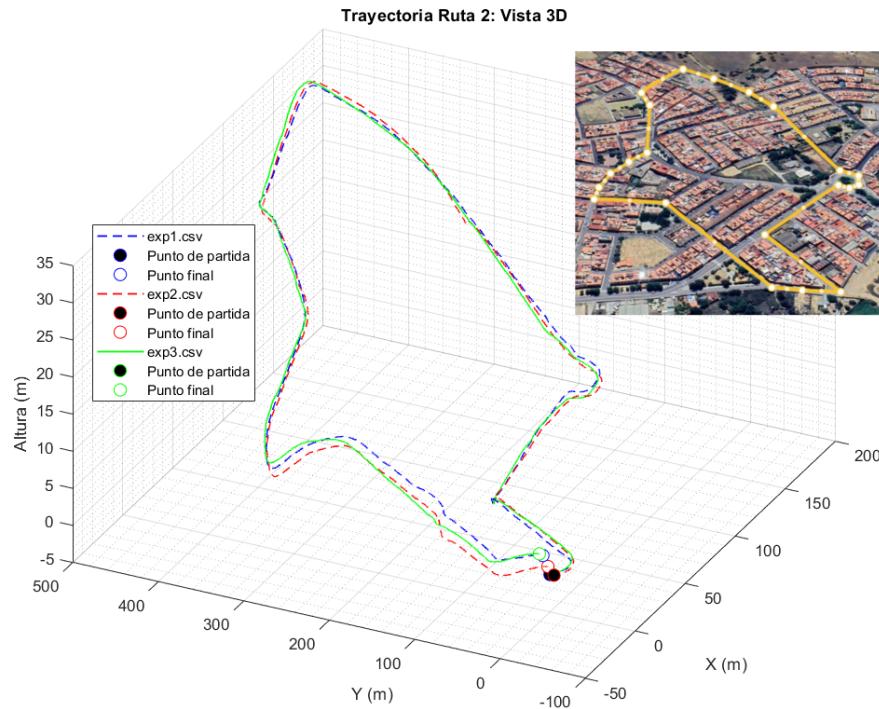


Figura 4.10 Recorrido en coche: Ruta 2.

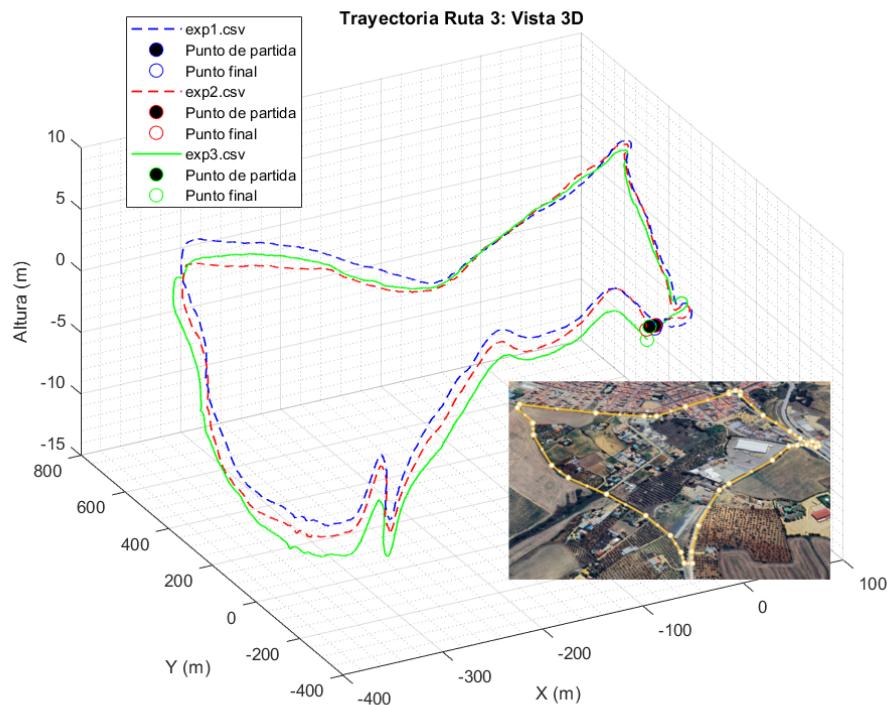


Figura 4.11 Recorrido en coche: Ruta 3.

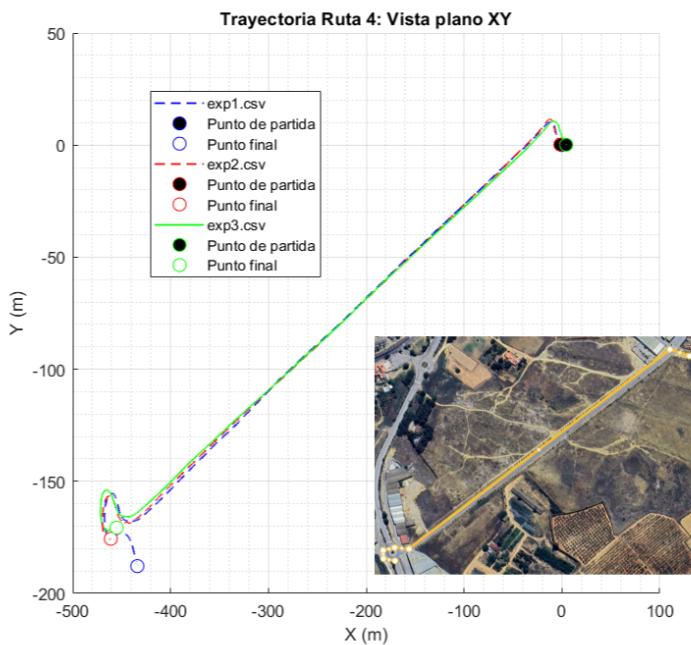


Figura 4.12 Recorrido en coche: Ruta 4.

Tabla 4.2 Resumen de pruebas. Verificación funcional del sistema.

Test	Descripción	Propósito	Resultado
Calibración	Ajuste de matriz de rotación según disposición final de VN-200.	Medidas referidas al body frame.	Solución en actitud correcta.
Autonomía	Someter al sistema a condiciones similares a las de vuelo.	Estimación de duración de la batería.	7 minutos. Tiempo suficiente para múltiples pruebas.
Alcance WiFi	Medición de la cobertura de comunicación WiFi entre el PC en tierra y placa Raspberry Pi.	Comprobar la integridad de conexión SSH y la correcta grabación de los datos.	Superior a 300 m. Distancia aceptable.
Alcance RF	Medición de la cobertura de comunicación RF para control teleoperado.	Comprobar el correcto funcionamiento del sistema primario RF tras ensamblaje.	Superior a 300 m. Rango admisible.
Recorrido rural	Grabación de datos en caminos rurales sin edificios.	Verificar funcionamiento GNSS en exteriores sin interferencias y con buena visibilidad de satélites.	Figura 4.9. Trayectorias de repeticiones coherentes.
Recorrido urbano	Grabación de datos en entornos urbanos con edificios.	Comprobar GNSS en exterior con posibles interferencias y condiciones más exigentes.	Figura 4.10. Trayectorias de repeticiones coherentes.
Recorrido cuestas	Grabación de datos en rutas con cuestas.	Validar la precisión en la estimación de la altitud.	Figura 4.10. Desviaciones entre iteraciones de 2 metros.
Recorrido rural-urbano	Grabación de datos en rutas mixtas rural-urbanas.	Revisar que no existan cambios significativos en la precisión de las medidas.	Figura 4.11. No se aprecian variaciones significativas.
Recorrido recto	Grabación de datos en carretera recta.	Examinar la aparición de posibles desviaciones en las medidas.	Figura 4.12. No encontramos anomalías.

5 Resultados de vuelo

El único lugar donde el éxito viene antes que el trabajo es en el diccionario.

VIDAL SASSOON

A barcamos las modificaciones implementadas en alas y cola del pájaro original, necesarias para su mejora en vuelo, así como los resultados finales logrados de pruebas en exterior.

5.1 Adaptaciones en superficies aerodinámicas

Antes de realizar el vuelo del sistema completo con toda la electrónica integrada, se llevaron a cabo simulaciones utilizando lastres para evaluar la capacidad de la plataforma original para soportar el incremento de peso asociado a los nuevos componentes, asegurando que se mantuviera la capacidad de vuelo. Los pesos simulados fueron colocados en el ornitóptero siguiendo la disposición planificada, y se realizaron pruebas de vuelo para determinar si la plataforma podía despegar y mantener el vuelo con el peso adicional. Los resultados iniciales mostraron que, aunque lograba despegar, descendía rápidamente, indicando que la superficie de las alas no era suficiente para generar la sustentación y el empuje necesarios.

En respuesta a esta observación, se decidió aumentar la superficie de las alas para incrementar carga aerodinámica y, en consecuencia, aumentar las fuerzas de sustentación y empuje necesarias para mantener el vuelo con el nuevo peso añadido. Las pruebas realizadas con alas de mayor tamaño demostraron una mejora, pero también revelaron problemas de maniobrabilidad, posiblemente debidos a una falta de capacidad de control de la cola original al haber aumentado la superficie sustentadora. Como consecuencia el sistema presentaba poca capacidad para efectuar giros y cambios de rumbo.



Figura 5.1 Comparativa entre alas originales y modificadas.

Para solucionar este problema, se optó por aumentar también la superficie de los flaps de cola. Tras varias iteraciones y ajustes en el diseño, se seleccionaron unas alas y cola modificadas que permitieron volar adecuadamente con los pesos simulados. La comparación entre las alas y la cola originales (en rojo) y las modificadas (en amarillo) se muestran en Figura 5.1 y Figura 5.2 respectivamente.

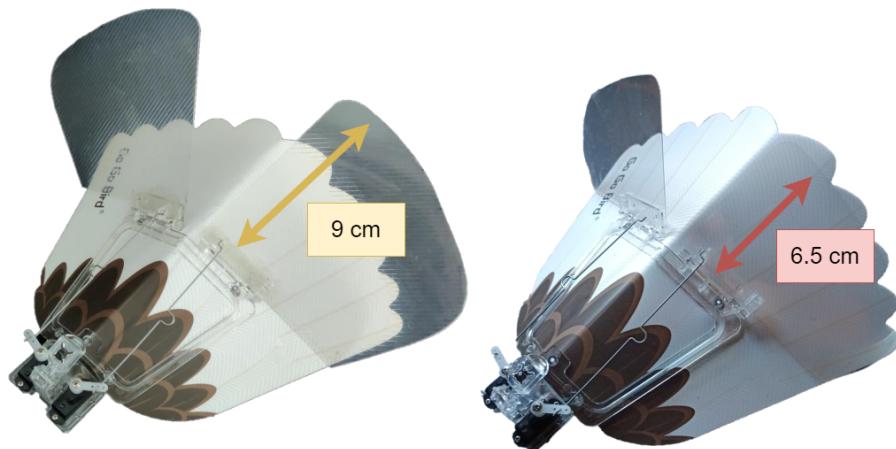


Figura 5.2 Comparativa entre flaps de cola originales y modificados.

5.2 Pruebas de vuelo y análisis de resultados

Tras haber detallado los preparativos y nuevas configuraciones del sistema, continuamos con las pruebas hechas para recopilar datos de vuelo del ornitóptero. En esta sección se presentan los resultados obtenidos, tras la integración de la electrónica completa, centrándonos en la configuración con alas y cola modificadas previamente descrita.

Antes de analizar los resultados, es importante considerar ciertos aspectos clave del tipo de prueba realizada. Los experimentos se llevaron a cabo en un entorno exterior, donde el sistema fue lanzado desde un punto de partida lo más similar posible en cada iteración, aunque con variaciones menores. Una vez en el aire, el ornitóptero voló de manera libre sin intervención en su trayectoria a través del mando de radiofrecuencia. Este solo se utilizó para ajustar la frecuencia de aleteo a su valor máximo, aproximadamente 10 Hz, y para detener el vuelo cuando fuera necesario. Por lo tanto, todos los tests de vuelo se realizaron bajo las mismas condiciones de frecuencia de aleteo. De esta forma se pretende estudiar los posibles desequilibrios en el montaje.

Se realizaron numerosas pruebas, de las cuales se seleccionaron tres representativas para un análisis detallado, en base a la similitud de las trayectorias seguidas, para proporcionar un análisis coherente y fiable de los resultados obtenidos. Todas ellas corresponden al montaje con las alas y cola modificadas de acuerdo a la Sección 5.1. Las comparativas se recogen en las gráficas asociadas a Figura 5.4, Figura 5.5, Figura 5.6, Figura 5.7, Figura 5.9 y Figura 5.8.

En todos los experimentos, se utilizó como punto de referencia inicial una altura de lanzamiento de 1.7 metros, determinada tras observar que en todas las iteraciones el sistema se encontraba a esa altura antes de ser lanzado (previamente a cada prueba se colocaba al UAV a nivel del suelo físico durante un tiempo prudente).

Para garantizar la veracidad de los resultados, se contrastaron las coordenadas iniciales y finales con grabaciones en vídeo de los experimentos, así como con herramientas como Google Maps y Google Earth, lo que permitió corroborar la precisión de las distancias y las trayectorias observadas en relación con el punto de lanzamiento.

Durante las repeticiones, se observó que el ornitóptero tendía a describir una curva hacia la izquierda en su sistema de referencia (body frame), manifestando un giro negativo en yaw y, por tanto, pendientes negativas en el rumbo. Este comportamiento se originó por un ángulo de roll negativo, en torno a los -20°, que difiere del valor nulo deseado, como se muestra en la Figura 5.7. Debido al acoplamiento de movimientos en este sistema, el ángulo de roll induce el giro en yaw. Este fenómeno se atribuye a un desequilibrio en la fabricación de la nueva ala. La hipótesis de una mala distribución de los pesos de los componentes añadidos fue descartada, ya que pruebas previas con la cola modificada y las alas originales lograron un vuelo razonablemente recto, como se puede apreciar en la Figura 5.3. Este hallazgo sugiere que las nuevas alas requieren un diseño y preparación más precisos. No obstante, los resultados obtenidos son valiosos para el análisis y estudio del rendimiento en vuelo de nuestro sistema.

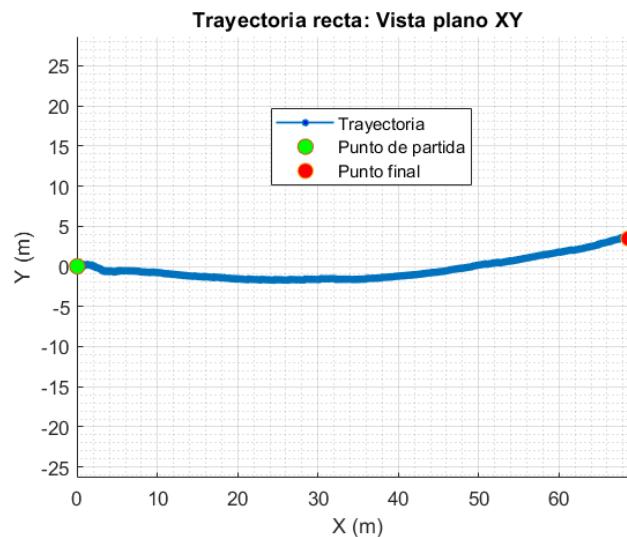


Figura 5.3 Vuelo sin desviaciones: Cola modificada y alas originales.

Tras el lanzamiento, el sistema adquiere una cierta altura que pierde pasados unos segundos, pasando a volar a ras de suelo mientras realiza el giro hasta que finalmente cae en el suelo, momento en el que finaliza la iteración concreta de cada test. Esto puede verse en las trayectorias en el plano vertical XZ, vista aérea en plano XY y tridimensional.

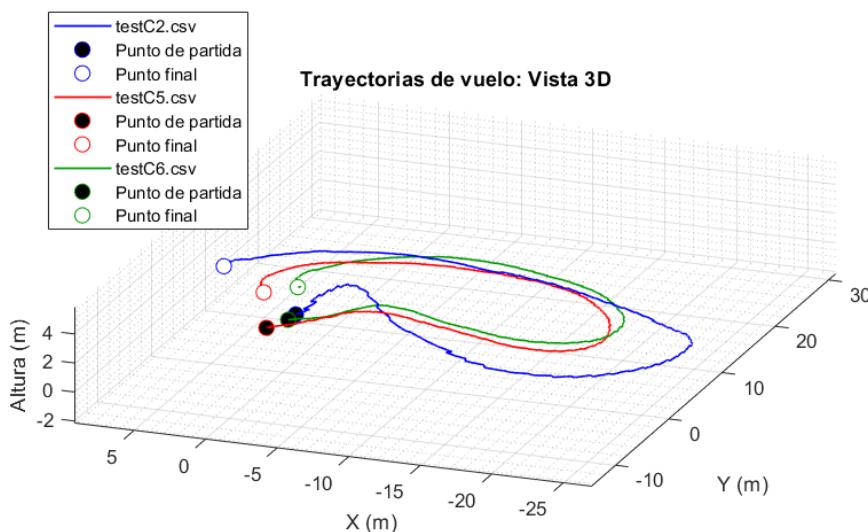


Figura 5.4 Trayectorias de vuelo 3D.

Concretamente, en la Figura 5.6, se observa que los puntos finales se encuentran a distintas alturas respecto al valor de referencia correspondiente al nivel del terreno en el punto de lanzamiento (inferiores a 1 m). Esto se debe, en parte, al pequeño desnivel de la superficie, aunque en gran medida podemos decir que es causa de la precisión en altura que tiene el sensor VN-200, tras la fusión de medidas de GNSS y barómetro.

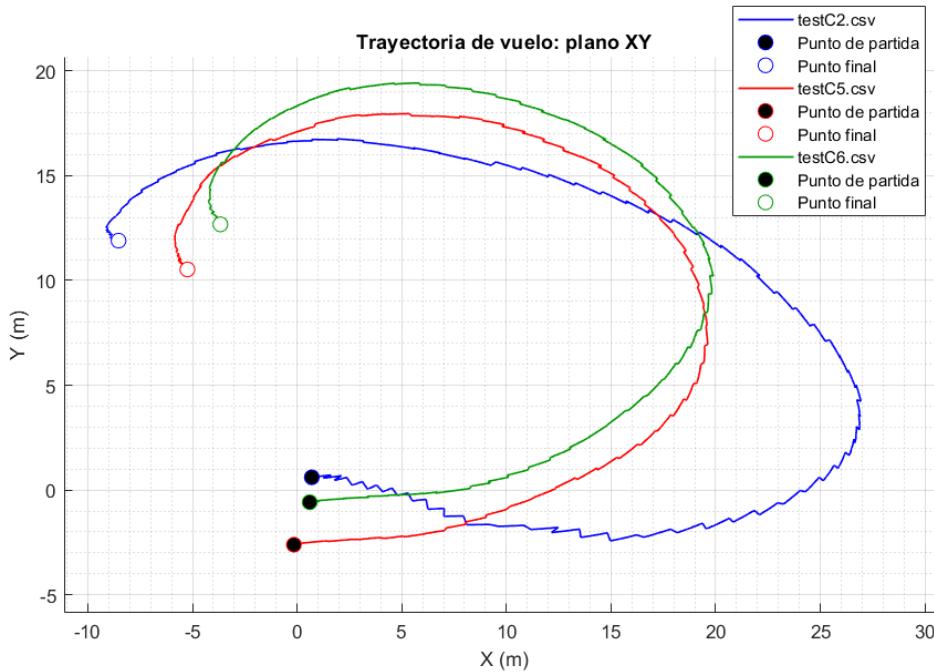


Figura 5.5 Trayectorias de vuelo. Vista plano XY.

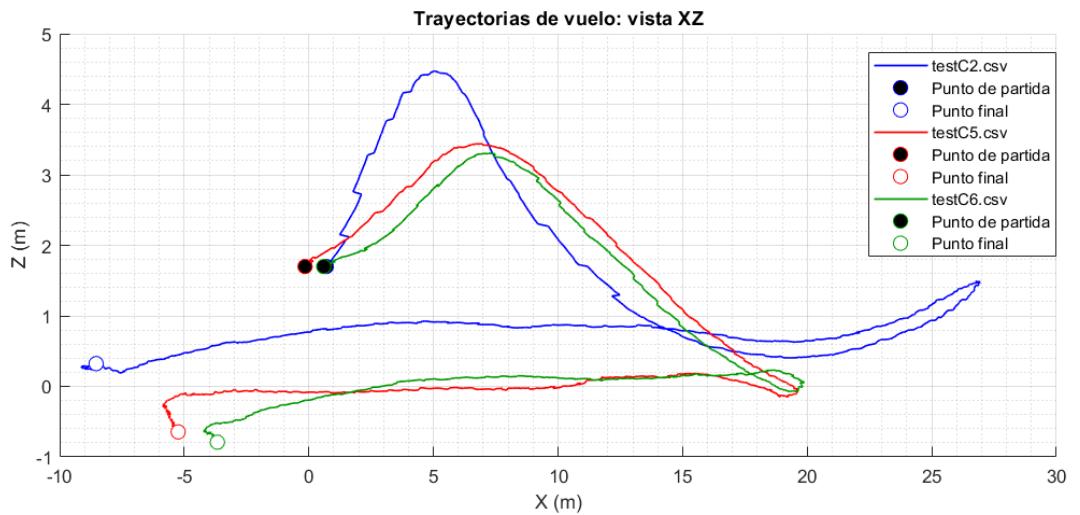


Figura 5.6 Trayectorias de vuelo. Vista plano XZ.

Durante el vuelo, el ángulo de cabeceo se mantiene bastante constante, en torno a los 20°, pasado el transitorio inicial correspondiente al despegue. Estos datos pueden consultarse nuevamente en la Figura 5.7. Los valores máximos se dan en el lanzamiento, que se intenta efectuar a un ángulo ideal de 45° para lograr que se alce el vuelo con éxito. También vemos como tras una subida rápida inicial en altura, el ángulo pitch se hace negativo por unos instantes hasta que logra recuperarse. Tras aterrizar, los ángulos pitch y roll se hacen nulos mientras que el valor final de yaw dependerá de la disposición del UAV tras la maniobra.

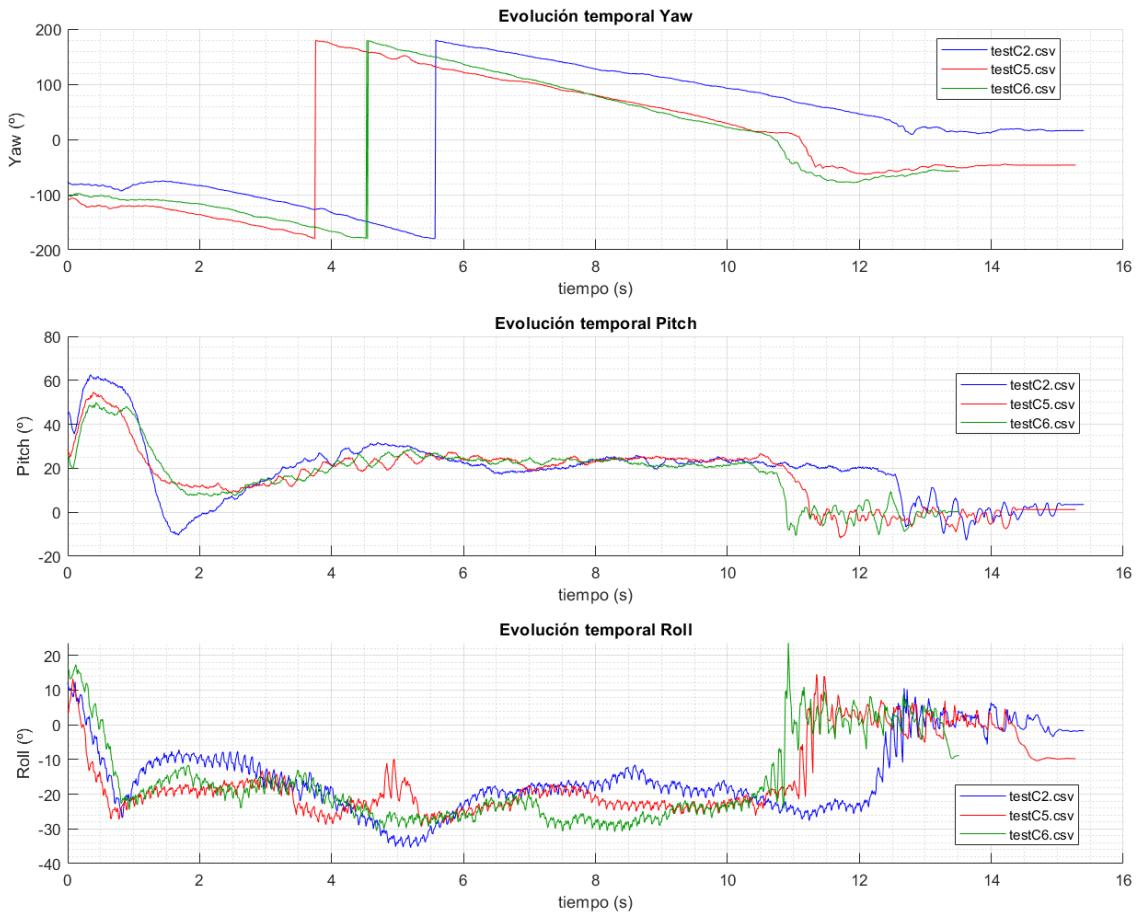


Figura 5.7 Evolución temporal de la actitud del ornitóptero en vuelo.

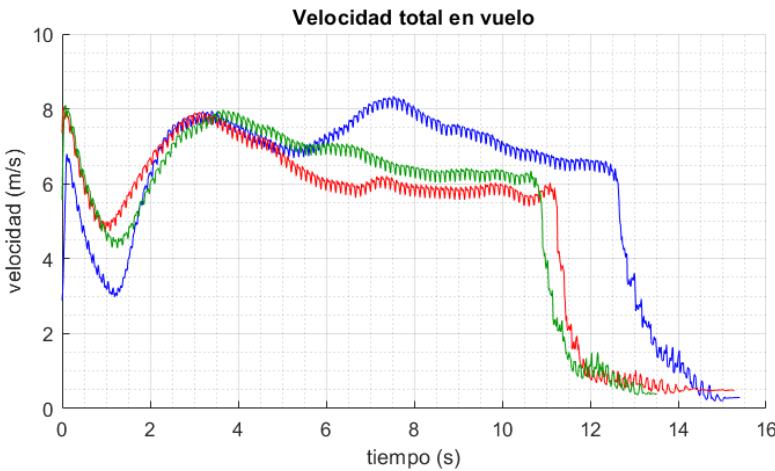


Figura 5.8 Evolución temporal de la velocidad total del ornitóptero en vuelo).

En cuanto a las velocidades alcanzadas durante los experimentos, la Figura 5.9 muestra su descomposición en el marco de referencia NED (North, East, Down). Observamos que la componente de velocidad en la dirección "down" presenta una evolución significativa durante la fase inicial de lanzamiento, cuando la altitud varía rápidamente. Sin embargo, durante el resto del vuelo, esta velocidad disminuye a valores casi nulos, lo que indica que el sistema logra estabilizarse en altura.

Por otro lado, las componentes de velocidad en las direcciones "north" y "east" muestran fluctuaciones notables, en torno a los ± 5 m/s. Estas variaciones reflejan el comportamiento dinámico del orníptero en el giro comentado en las trayectorias. La combinación de estas componentes da lugar a una velocidad total nominal en torno a 6-7 m/s, tal como se observa en la Figura 5.8

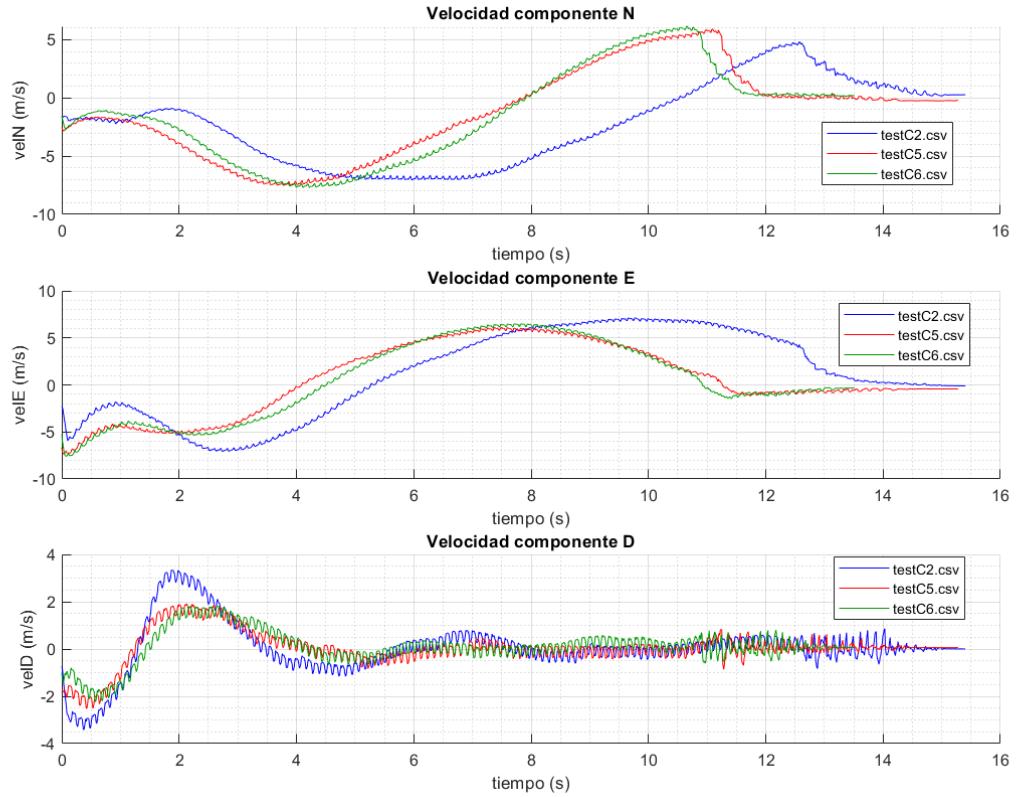


Figura 5.9 Evolución temporal de las componentes de velocidad (NED).

El sensor VN-200 proporciona las velocidades en coordenadas NED. Sin embargo, para un análisis más intuitivo, estas velocidades se transforman a los ejes del cuerpo (UVW). Esta transformación permite comprender mejor el movimiento relativo del objeto en su propio marco de referencia. La transformación de velocidades de NED a UVW se realiza mediante la matriz de transformación del sistema de ejes horizonte local al sistema de ejes cuerpo, L_{bh} , que depende de los ángulos de orientación: *roll* (ϕ), *pitch* (θ), y *yaw* (ψ).

La relación entre las velocidades en coordenadas NED (v_N, v_E, v_D) y las velocidades en los ejes del cuerpo (u, v, w) está dada por la ecuación:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = L_{bh} \begin{bmatrix} v_N \\ v_E \\ v_D \end{bmatrix}$$

donde la matriz de transformación L_{bh} es:

$$L_{bh} = \begin{bmatrix} \cos(\theta)\cos(\psi) & \cos(\theta)\sin(\psi) & -\sin(\theta) \\ \sin(\phi)\sin(\theta)\cos(\psi) - \cos(\phi)\sin(\psi) & \sin(\phi)\sin(\theta)\sin(\psi) + \cos(\phi)\cos(\psi) & \sin(\phi)\cos(\theta) \\ \cos(\phi)\sin(\theta)\cos(\psi) + \sin(\phi)\sin(\psi) & \cos(\phi)\sin(\theta)\sin(\psi) - \sin(\phi)\cos(\psi) & \cos(\phi)\cos(\theta) \end{bmatrix}$$

Tras aplicar la transformación comentada a las velocidades de la Figura 5.9, obtenemos la descomposición de velocidades en ejes cuerpo, véase Figura 5.10.

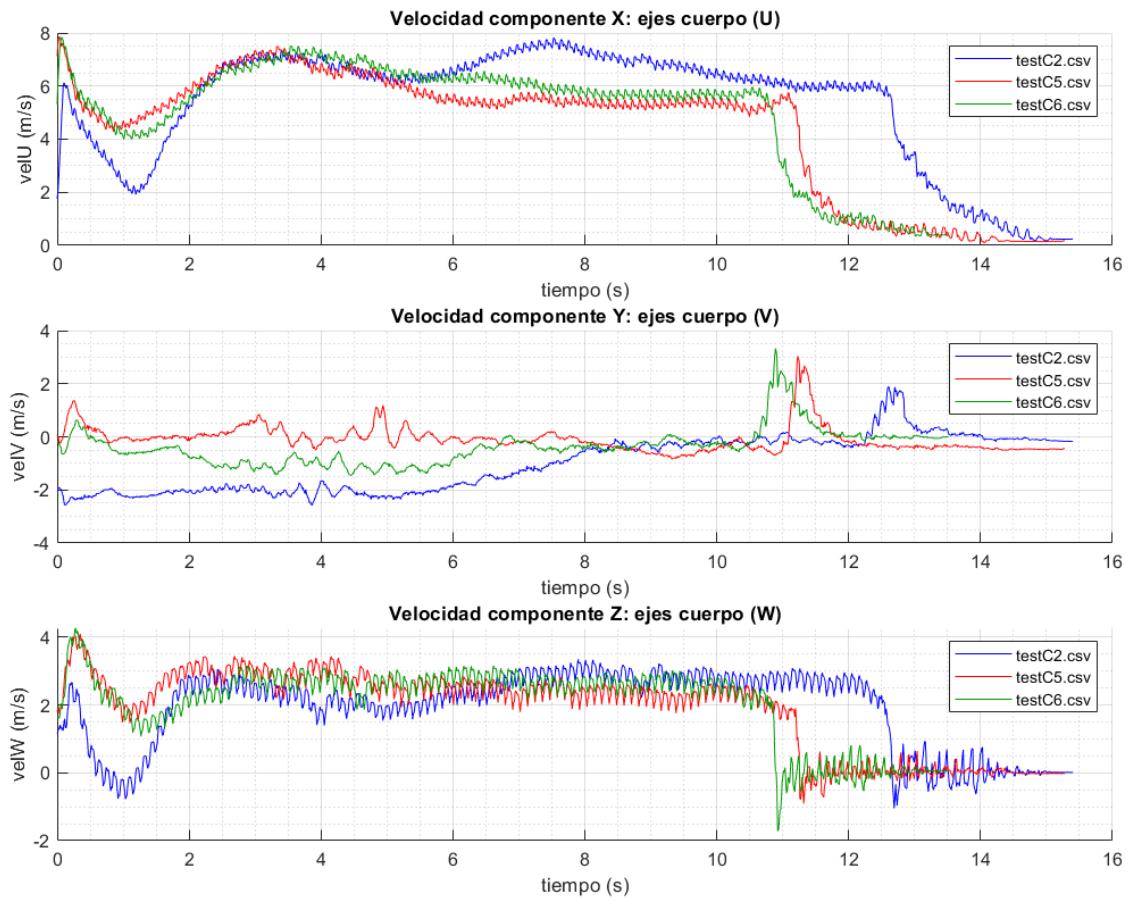


Figura 5.10 Evolución temporal de las componentes de velocidad en ejes cuerpo (UVW).

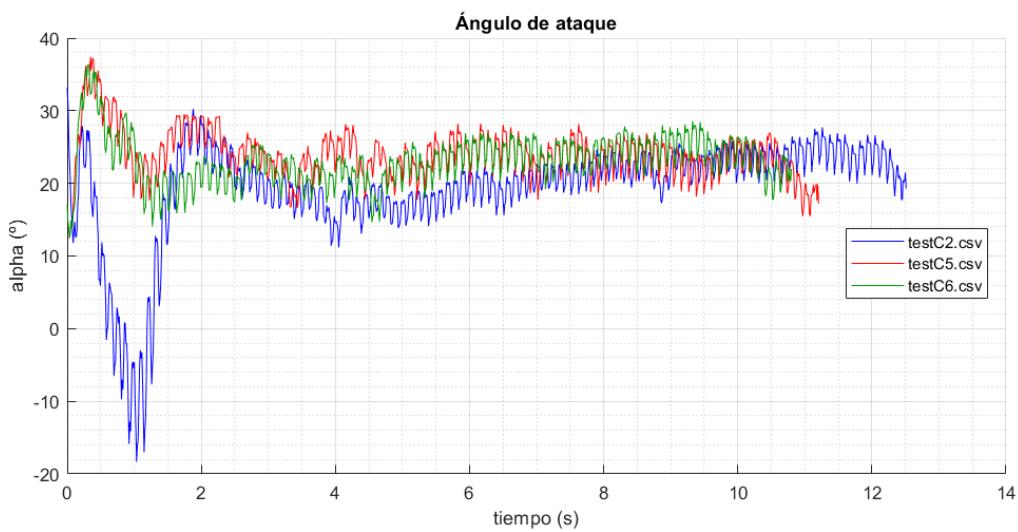


Figura 5.11 Evolución temporal del ángulo de ataque.

Se observa que la componente predominante de la velocidad es la velocidad longitudinal u , la cual permanece siempre positiva, con valores alrededor de 6-7 m/s. En cuanto a la componente de velocidad v , puede verse que no es nula, lo que se debe a la presencia de giro durante el vuelo. La componente de velocidad w también es siempre positiva, lo cual es crucial para mantener un ángulo de ataque positivo. Un ángulo de ataque negativo implicaría que la fuerza aerodinámica, en lugar de proporcionar sustentación, generaría una fuerza que haría descender al ornitóptero.

En la Figura 5.11 puede verse el ángulo de ataque presente durante las pruebas, el cual se estabiliza en torno a los 25°. La expresión para calcular el ángulo de ataque, α , se define como:

$$\alpha = \arctan\left(\frac{w}{u}\right)$$

donde w es la componente de la velocidad en el eje z del sistema de ejes cuerpo (body frame) y u es la componente de la velocidad en el eje x del mismo sistema, como ya se ha descrito con anterioridad.

Para completar el análisis de velocidades, la Figura 5.12 muestra las trayectorias del UAV con colores que representan la velocidad total en cada punto del recorrido. Esta visualización espacial permite identificar intuitivamente las áreas con velocidades superiores o inferiores, ofreciendo una perspectiva clara de cómo varía la velocidad a lo largo de la trayectoria. A diferencia de las representaciones temporales anteriores, esta gráfica facilita la identificación de patrones de velocidad en distintas secciones del vuelo (lanzamiento inicial, avance, giro y aterrizaje).

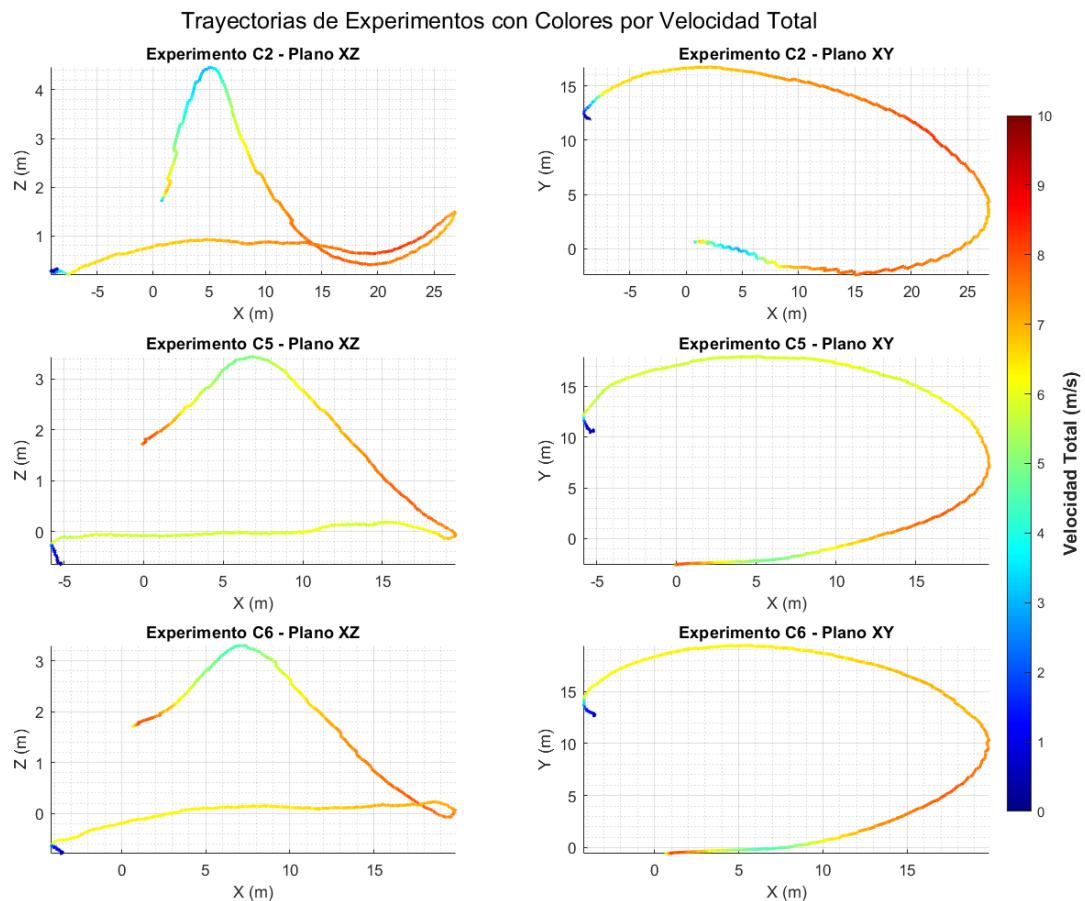


Figura 5.12 Velocidad total en trayectorias de vuelo.

Por último, en la Tabla 5.1 quedan recogidos los principales resultados obtenidos de las pruebas mostradas.

Tabla 5.1 Resultados y características de pruebas de vuelo.

Característica	Valor
Tiempo de vuelo	10 a 12 s (suficiente para análisis de datos)
Distancia recorrida	88 m (testC2), 68 m (testC5), 65 m (testC6)
Altura inicial	1.7 m sobre altura de referencia (superficie del terreno)
Velocidad total	6 a 7 m/s (vuelo libre, régimen estacionario)
Ángulo cabeceo (θ)	20 ° (vuelo libre, régimen estacionario)
Ángulo alabeo (ϕ)	-20 a -30 ° debido a giro (vuelo libre, régimen estacionario)
Ángulo de ataque (α)	20 a 25 ° (vuelo libre, régimen estacionario)

6 Conclusiones y Trabajo futuro

Un objetivo sin un plan es solo un deseo.

ANTOINE DE SAINT-EXUPÉRY

Pese a los avances significativos logrados hasta ahora, queda mucho trabajo por hacer en este campo. Este capítulo se centrará en explorar posibilidades de mejora para el futuro del trabajo, identificando los desafíos y oportunidades que se avecinan.

Se analizarán los aspectos clave que deben abordarse para seguir avanzando en este proyecto.

6.1 Objetivos generales y particulares

El desarrollo del proyecto ha supuesto una base de conocimiento teórico y experimental para el correcto uso de la tecnología VectorNav, la cual puede ser altamente versátil en el sentido de posibles extensiones de su aplicación más allá del orníntóptero. Podremos embarcar el VN-200 en otros sistemas autónomos, como vehículos terrestres (UGV) y embarcaciones no tripulados (USV).

Entre los logros clave, se encuentra la sustitución de la antena original por una más ligera, reduciendo peso y mejorando la eficiencia energética del sistema. Además, se ha adquirido un conocimiento tanto de las herramientas de alto nivel del fabricante como de la librería para comunicación en bajo nivel con placas hardware, lo que ha permitido un uso efectivo del sensor en diferentes plataformas.

La correcta distribución de los pesos en el chasis de partida tras el diseño del sistema electrónico, acompañado por el desarrollo del programa de adquisición de datos que cumple con las especificaciones funcionales de partida requeridas para nuestra aplicación. La selección adecuada del computador de a bordo y la cohesión de todos los componentes han garantizado el éxito del sistema.

Finalmente, a la luz de los resultados obtenidos en los vuelos realizados con el robot aéreo, podemos concluir que han sido altamente satisfactorios y son un testimonio del esfuerzo y dedicación invertidos en este proyecto.

A pesar de los desafíos que surgieron, hemos logrado que el orníntóptero vuele de manera estable con la nueva configuración y la electrónica integrada. Si bien aún existe margen para optimizar el equilibrio de las superficies alares, nuestro objetivo principal, que era el diseño e implementación funcional del nuevo sistema electrónico embebido para la recopilación de datos, se ha alcanzado con éxito, cumpliendo con las especificaciones iniciales.

Este logro representa un avance significativo, proporcionando un entendimiento más profundo del comportamiento en vuelo del sistema y estableciendo una base sólida para futuros desarrollos y nuevos proyectos que requieran un módulo INS/GNSS.

6.2 Perspectivas de desarrollo

Control de actuadores

Aunque el control de los actuadores sale del alcance principal de este TFM, ya se están realizando avances para que la Raspberry Pi Zero W asuma esta tarea, eliminando la necesidad del canal de radiofrecuencia (RF) actual. Esto permitiría una reducción de peso y una comunicación exclusivamente mediante Wi-Fi con la estación de control en tierra. En el *Proyecto Fin de Grado* [2], se desarrolló un sistema de control manual para los actuadores, que podría servir como base para este nuevo desarrollo, aprovechando tanto el software existente como el conocimiento del hardware involucrado.

Ya se han realizado pruebas iniciales utilizando la Raspberry Pi Zero W para controlar los actuadores del orníptero. La placa dispone de salidas PWM a través del GPIO, con dos generadores de PWM y dos pines asociados a cada uno. En estas pruebas, un generador PWM se utilizó para controlar el motor, mientras que el otro se destinó al control de los dos servomotores de cola. Esto es posible gracias a la capacidad de variar el ciclo de trabajo (duty cycle) de cada pin, siempre que la frecuencia de la PWM generada sea la misma para ambos pines de un mismo generador. Véase los pines de PWM0 (GPIO18 y GPIO12) y PWM1 (GPIO13 y GPIO19) en la Figura 3.3.

Para implementar este control, se ha utilizado la librería *pigpio* en Python, que permite generar señales PWM con la precisión necesaria para el correcto funcionamiento de los actuadores. Durante las pruebas, se logró sincronizar el motor mediante una señal PWM de 50 Hz y un duty cycle del 5%, regulando así la velocidad del motor mediante ajustes precisos del duty cycle (en el rango comprendido entre el 5 y 12%). Inicialmente, se empleó la librería RPi.GPIO, la cual generaba PWM, pero tras analizar las señales con un osciloscopio, se observó que la precisión del duty cycle era insuficiente para un control fiable, oscilando en torno al 5%, no lográndose la sincronización inicial con la secuencia sonora asociada en el motor brushless. La librería pigpio, por el contrario, ha demostrado ser suficientemente precisa, asegurando un rendimiento óptimo del sistema de control de actuadores.

Adición de sensores

Para mejorar el control y la monitorización del orníptero, podríamos añadir sensores adicionales que permitan una gestión más precisa y un ajuste en tiempo real durante el vuelo. Por ejemplo, podríamos integrar un ligero sensor de efecto Hall para medir la frecuencia real de aleteo de las alas, con un pequeño imán en el mecanismo de movimiento. Este sensor proporcionaría datos que permitirían ajustar la frecuencia de aleteo y, en consecuencia, controlar la velocidad y altura del UAV.

También podemos considerar la incorporación de un barómetro y una IMU adicionales como complementos al sensor principal VectorNav. La idea es tener estos sensores de respaldo disponibles para garantizar la continuidad del funcionamiento en caso de fallo del sensor principal. Esta estrategia de redundancia no solo incrementa la seguridad operativa, sino que también mejora la capacidad del orníptero para adaptarse a condiciones imprevistas, asegurando un funcionamiento más robusto en vuelo. Además, en situaciones de falla de uno de los sensores, los datos proporcionados por los sensores secundarios permitirían realizar ajustes en tiempo real para minimizar el impacto en la operación del UAV.

Tareas autónomas

Se pueden explorar diversas técnicas de control realimentado para el manejo autónomo del vuelo, tales como el seguimiento de trayectorias, el control de altura, el posado suave y el planeo para un ahorro energético. Estos temas fueron inicialmente introducidos durante el desarrollo del Trabajo Fin de Grado, alcanzándose ciertos avances preliminares. Sin embargo, contábamos con una gran limitación debido a la falta de estimaciones precisas de actitud, velocidad y posición del orníptero.

Con los recientes avances conseguidos en este Trabajo Fin de Máster, ahora disponemos de esa información necesaria para avanzar en el desarrollo y perfeccionamiento de estas tareas.

Inclusión de mini-cámara

La integración de una mini cámara en el ornitóptero podría aportar significativas ventajas y ampliar las aplicaciones del sistema. Con una cámara, podrían desarrollarse tareas basadas en visión por computador para evitar obstáculos durante el vuelo, mejorando la seguridad y reduciendo el riesgo de colisiones. Además, ofrecería al operador una visión en tiempo real del entorno, lo que facilitaría la supervisión y el monitoreo de la misión.

Esta mejora también abre la puerta a aplicaciones en áreas como espionaje y vigilancia, donde la discreción y agilidad del robot de ala batiente serían cruciales para misiones encubiertas. Además, la posibilidad de capturar imágenes y videos aéreos desde una perspectiva única podría ser valiosa en sectores como la fotografía y el cine.

Considerando que la placa Raspberry Pi Zero W cuenta con el hardware necesario para acoplar una cámara, sería posible transmitir imágenes en tiempo real al ordenador de la estación base. Esto no solo mejoraría las capacidades de monitoreo, sino que también ofrecería nuevas posibilidades para el desarrollo de futuras aplicaciones, aumentando la versatilidad y el valor del sistema. Una opción compatible de bajo coste a considerar sería el modelo Raspberry Pi Camera Module 3 (con 12 MP y autoenfoque).

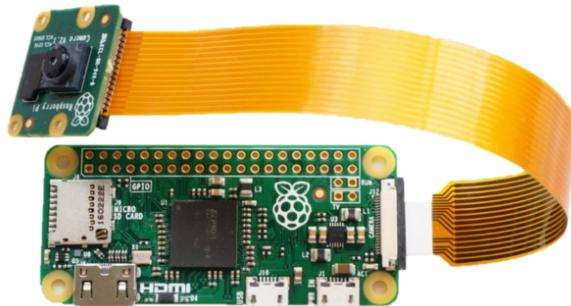


Figura 6.1 Raspberry Pi Zero con módulo de cámara acoplado.

Apéndice A

Registros internos VectorNav

A.1 Registro 26: Reference Frame Rotation

Register ID	26
Size (Bytes)	36
Description	Allows for measurements to be rotated into a different reference frame. Defines the transformation from sensor-frame to body-frame.
Example Read Response	\$VNRRG,26,1,0,0,0,1,0,0,0,1*6A
Example Write Command	\$VNWRG,26,1,0,0,0,-1,0,0,0,-1*6F

OFFSET	NAME	FORMAT	UNIT	DEFAULT	DESCRIPTION
0	RFR00	float	-	1	Direction cosine matrix, Row 0, Column 0
4	RFR01	float	-	0	Direction cosine matrix, Row 0, Column 1
8	RFR02	float	-	0	Direction cosine matrix, Row 0, Column 2
12	RFR10	float	-	0	Direction cosine matrix, Row 1, Column 0
16	RFR11	float	-	1	Direction cosine matrix, Row 1, Column 1
20	RFR12	float	-	0	Direction cosine matrix, Row 1, Column 2
24	RFR20	float	-	0	Direction cosine matrix, Row 2, Column 0
28	RFR21	float	-	0	Direction cosine matrix, Row 2, Column 1
32	RFR22	float	-	1	Direction cosine matrix, Row 2, Column 2

A.2 Registro 57: GNSS Internal A Antenna Offset

Register ID	57
Size (Bytes)	12
Description	Configures the position offset of GNSS Internal A (GnssA) antenna from the INS Reference Point in the body reference frame.
Example Read Response	\$VNRRG,57,0,0,0*6D
Example Write Command	\$VNWRG,57,1,1,1*69

OFFSET	NAME	FORMAT	UNIT	DEFAULT	DESCRIPTION
0	PositionX	float	m	0	Relative position of GnssA antenna, body-frame x-axis.
4	PositionY	float	m	0	Relative position of GnssA antenna, body-frame y-axis.
8	PositionZ	float	m	0	Relative position of GnssA antenna, body-frame z-axis.

A.3 Registro 67: INS Basic Configuration

Register ID	67
Size (Bytes)	4
Description	Configures basic parameters of the INS.
Example Read Response	\$VNRRG,67,2,1,0,0*71
Example Write Command	\$VNWRG,67,2,1,0,0*XX

OFFSET	NAME	FORMAT	UNIT	DEFAULT	DESCRIPTION
0	Scenario	uint8	ENUM	2	Active INS scenario.
1	AhrsAiding	uint8	ENUM	1	Selects whether the INS should use an AHRS-based attitude solution during periods when there is insufficient GNSS-based attitude observability.
2	Resv1	uint8	-	0	Reserved. Must be set to default value.
3	Resv2	uint8	-	0	Reserved. Must be set to default value.

Enumeration: Scenario

NAME	VALUE	DESCRIPTION
Ahrs	0	AHRS (Attitude Only - no GNSS/INS)
GnssInsWithPressure	1	GNSS/INS - with barometric pressure sensor.
GnssInsNoPressure	2	GNSS/INS - without barometric pressure sensor.

A.4 Registros 75-77: User Output Configuration

Binary Output Register 1-3				
Register ID :	75-77	Access :	Read / Write	
Comment :	These registers allow the user to construct a custom output message that contains a collection of desired estimated states and sensor measurements.			
Size (Bytes):	6-22			
Example Response:	\$VNWRG,75,2,4,1,8*XX			
Offset	Name	Format	Unit	Description
0	AsyncMode	uint16	-	Selects whether the output message should be sent out on the serial port(s) at a fixed rate. 0 = None. User message is not automatically sent out either serial port. 1 = Message is sent out serial port 1 at a fixed rate. 2 = Message is sent out serial port 2 at a fixed rate. 3 = Message is sent out both serial ports at a fixed rate.
2	RateDivisor	uint16	-	Sets the fixed rate at which the message is sent out the selected serial port(s). The number given is a divisor of the ImuRate which is nominally 800Hz. For example to have the sensor output at 50Hz you would set the Divisor equal to 16.
4+N	OutputGroup(N)	uint8	-	Selects which output groups are active in the message. The number of OutputFields in this message should equal the number of active bits in the OutputGroup .
4+N+2*M	OutputField(1)	uint16	-	Selects which output data fields are active within the selected output groups.

A.5 Registro 99: GNSS System Configuration

Register ID	99
Size (Bytes)	12
Description	Configuration of the GNSS constellations and satellites.
Example Read Response	\$VNRRG,99,0013,10,5,16,03,EBF9,003F,0000*4C
Example Write Command	\$VNWRG,99,0013,10,10,16,02,EBF9,003F,0000*XX

OFFSET	NAME	FORMAT	UNIT	DEFAULT	DESCRIPTION
0	Systems	uint16	BITS	0013	Hex value bitfield of enabled GNSS systems.
2	MinCno	uint8	dB	10	Minimum satellite CN0 to use for GNSS navigation solution. (Cannot be set greater than 50 dB)
3	MinElev	uint8	deg	5	Minimum satellite elevation angle to use for GNSS navigation solution. (Cannot be set greater than 90°)
4	MaxSats	uint8	-	16	Maximum number of satellites to use in GNSS navigation solution. (Cannot be set less than 4 or greater than 32)
5	SbasMode	uint8	BITS	03	Hex value bitfield configuring SBAS usage.
6	SbasSelect1	uint16	BITS	EBF9	Hex value bitfield masking individual SBAS satellites for use.
8	SbasSelect2	uint16	BITS	003F	Hex value bitfield masking individual SBAS satellites for use.
10	SbasSelect3	uint16	BITS	0000	Hex value bitfield masking individual SBAS satellites for use.

Bit Field: Systems

NAME	OFFSET	DESCRIPTION
GPS	0	GPS
SBAS	1	SBAS
GLONASS	2	GLONASS
Beidou	3	Beidou
Galileo	4	Galileo
IMES	5	IMES
QZSS_L1_CA	6	QZSS-L1 C/A
QZSS_L1_SAIF	7	QZSS-L1 SAIF

Bit Field: SbasMode

NAME	OFFSET	DESCRIPTION
Ranging	0	Use ranging in navigation solution.
DiffCorr	1	Apply SBAS differential corrections.
Integrity	2	Use SBAS integrity information.
TestMode	3	Utilize signals from SBAS satellites broadcasting in test mode.

A.6 Registro 105: INS Reference Point Offset

Register ID.....	105
Size (Bytes).....	24
Description.....	Configures the position offset and measurement uncertainty of the INS Reference Point relative to the IMU sensor installation location in the body-frame.
Example Read Response	\$VNRRG,105,+00.000,+00.000,+00.000,00.0000,00.0000, 00.0000*5C
Example Write Command	\$VNWRG,105,1,1,1,1,1,1*42

OFFSET	NAME	FORMAT	UNIT	DEFAULT	DESCRIPTION
0	RefOffsetX	float	m	0	Position of INS Reference Point relative to IMU location, body-frame x-axis.
4	RefOffsetY	float	m	0	Position of INS Reference Point relative to IMU location, body-frame y-axis.
8	RefOffsetZ	float	m	0	Position of INS Reference Point relative to IMU location, body-frame z-axis.
12	RefUncertX	float	m	0.0	Uncertainty in the x-axis offset measurement.
16	RefUncertY	float	m	0.0	Uncertainty in the y-axis offset measurement.
20	RefUncertZ	float	m	0.0	Uncertainty in the z-axis offset measurement.

Apéndice B

Código completo programa principal

```
/*
*
* Main program for communication between VN-200 and Raspberry Pi Zero W.
* Telemetry data recording.
*
*
* AUTHOR: Álvaro García Lora
* DATE: August 2024
*
*
* DESCRIPTION:
* This program connects to a VectorNav VN-200 sensor via serial port,
* configures it to output telemetry data in a binary format, and records
* the data to a CSV file. The program is designed to run continuously,
* logging telemetry information such as yaw, pitch, roll, latitude,
* longitude, altitude, and velocities, while monitoring changes in the
* INS mode.
*
* USAGE:
* Compile and run this program on a Raspberry Pi Zero W connected to a VN-200
* sensor. Ensure the correct serial port and baud rate are configured in the
* code.
*/

```

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <string>

// Header file for VectorNav sensors access
#include "vn/sensors.h"

using namespace std;
using namespace vn::math;
using namespace vn::sensors;
using namespace vn::protocol::uart;
```

```
// ----- Class to handle the CSV file operations -----  
  
class CsvFileHandler {  
public:  
  
    // Constructor: Opens the CSV file and writes the header  
  
    CsvFileHandler(const string& filename) : m_filename(filename) {  
        m_outputFile.open(filename);  
        if (!m_outputFile.is_open()) {  
            cerr << "Error opening CSV file." << endl;  
            exit(1);  
        }  
        m_outputFile << "Timestamp,Yaw,Pitch,Roll,Lat,Lon,Alt,VelN,VelE,VelD,  
        InsMode" << endl;  
    }  
  
    // Destructor: Closes the CSV file  
  
    ~CsvFileHandler() {  
        m_outputFile.close();  
    }  
  
    // Method to write data to the CSV file  
  
    void writeData(uint64_t timestamp, const vec3f& yawPitchRoll, const vec3d&  
        position, const vec3f& nedVel, uint16_t insMode) {  
        m_outputFile << timestamp << ","  
        << fixed << setprecision(3) << yawPitchRoll.x << "," // Yaw  
        << fixed << setprecision(3) << yawPitchRoll.y << "," // Pitch  
        << fixed << setprecision(3) << yawPitchRoll.z << "," // Roll  
        << fixed << setprecision(7) << position.x << "," // Latitude  
        << fixed << setprecision(7) << position.y << "," // Longitude  
        << fixed << setprecision(3) << position.z << "," // Altitude  
        << fixed << setprecision(3) << nedVel.x << "," // North Velocity  
        << fixed << setprecision(3) << nedVel.y << "," // East Velocity  
        << fixed << setprecision(3) << nedVel.z << "," // Down Velocity  
        << insMode << endl; // INS Mode  
    }  
  
private:  
    ofstream m_outputFile; // Output file stream  
    string m_filename; // CSV file name  
};
```

```
// Declaration of the function to handle incoming async messages  
  
void BinaryAsyncMessageReceived(void* userData, Packet& p, size_t index);
```

```
// ----- Main function -----  
  
int main(int argc, char *argv[])  
{  
    const string SensorPort = "/dev/ttyS0"; // Sensor's serial port  
    const uint32_t SensorBaudrate = 115200; // Baud rate  
  
    VnSensor vs; // Sensor object  
    vs.connect(SensorPort, SensorBaudrate); // Connect to the sensor  
  
    // Configure binary output register to include required data groups  
  
    BinaryOutputRegister bor(  
        ASYNCMODE_PORT2,  
        9, // Output frequency (800 / 9 = 89 Hz)  
        COMMONGROUP_TIMESTAMP | COMMONGROUP_YAWPITCHROLL | COMMONGROUP_POSITION |  
            COMMONGROUP_VELOCITY | COMMONGROUP_INSSTATUS,  
        TIMEGROUP_NONE,  
        IMUGROUP_NONE,  
        GPSGROUP_NONE,  
        ATTITUDEGROUP_NONE,  
        INSGROUP_NONE,  
        GPSGROUP_NONE);  
  
    vs.writeBinaryOutput1(bor); // Apply binary output settings  
  
    // Initialize CSV file handler  
    CsvFileHandler csvHandler("data.csv");  
  
    // Register handler for async packet reception  
    vs.registerAsyncPacketReceivedHandler(&csvHandler,  
        BinaryAsyncMessageReceived);  
  
    // Infinite loop to keep the program running  
    while(1){  
        // Wait for incoming data (handled asynchronously)  
    }  
  
    // Cleanup: unregister handler and disconnect the sensor  
    vs.unregisterAsyncPacketReceivedHandler();  
    vs.disconnect();  
  
    return 0;  
}
```

```
// ----- Function to handle received async messages (Binary only) -----  
  
void BinaryAsyncMessageReceived(void* userData, Packet& p, size_t index)  
{  
    static uint16_t prevInsMode = 0xFFFF; // Initial value outside expected  
    range  
    CsvFileHandler* csvHandler = static_cast<CsvFileHandler*>(userData);  
  
    if (p.type() == Packet::TYPE_BINARY)  
    {  
        // Check if the binary packet matches the expected format  
        if (!p.isCompatible(  
            COMMONGROUP_TIMESTAMP | COMMONGROUP_YAWPITCHROLL | COMMONGROUP_POSITION  
            | COMMONGROUP_VELOCITY | COMMONGROUP_INSSTATUS,  
            TIMEGROUP_NONE,  
            IMUGROUP_NONE,  
            GPSGROUP_NONE,  
            ATTITUDEGROUP_NONE,  
            INSGROUP_NONE,  
            GPSGROUP_NONE))  
            // If not compatible, exit the function  
            return;  
  
        // Extract data from the binary packet  
        uint64_t time = p.extractUint64(); // Startup time  
        vec3f yawPitchRoll = p.extractVec3f(); // Yaw, Pitch, Roll  
        vec3d position = p.extractVec3d(); // Position (Lat, Lon, Alt)  
        vec3f nedVel = p.extractVec3f(); // NED velocities (North, East, Down)  
        uint16_t InsStatus = p.extractUint16(); // INS Status  
        uint16_t mask = 0x03; // Mask for the 2 least significant bits  
        uint16_t InsMode = InsStatus & mask; // INS Mode  
  
        // Write the extracted data to the CSV file  
        csvHandler->writeData(time, yawPitchRoll, position, nedVel, InsMode);  
  
        // Check and report changes in INS mode  
        if (prevInsMode != InsMode) {  
            if (prevInsMode == 1 && InsMode == 2) {  
                cout << "MODE 2 activated" << endl;  
            } else if (prevInsMode == 2 && InsMode == 1) {  
                cout << "MODE 2 lost" << endl;  
            }  
            // Update previous INS mode  
            prevInsMode = InsMode;  
        }  
    }  
}
```

Índice de Figuras

1.1	Producto comercial GoGobird® 1020 Eagle - Hanvon	2
2.1	Vectornav VN-200 Rugged	5
2.2	Conectores y pinout del Vectornav VN-200 Rugged	6
2.3	Marcos de referencia sensor y cuerpo	7
2.4	Marcos de referencia LLA y NED	8
2.5	Rotación de marcos de referencia	10
2.6	Antena TW2712 original del kit VN-200 Rugged	12
2.7	Antena TW1430 seleccionada	13
2.8	Bandas de frecuencia de navegación GPS, GLONASS y Galileo.	14
2.9	Configuración interna de ajustes GNSS en software Control Center	14
2.10	Limitación en el ancho de banda y configuración del mensaje binario	16
3.1	Raspberry Pi Zero W	17
3.2	Asignaciones de pines GPIO Raspberry Pi Zero W	18
3.3	Ejemplo de fichero .csv generado por programa principal	21
4.1	Sistema electrónico interno de GoGobird	23
4.2	Esquema del conexionado eléctrico	24
4.3	Esquema conceptual de equipo y conectividad SHH vía WiFi	24
4.4	Distribución de componentes en modelo 3D	25
4.5	Conector Harwin fabricado. Conexión segura con VN-200	25
4.6	Proceso de integración física	26
4.7	Detalles de montaje y acabado final	26
4.8	Calibración de actitud en el VN-200	27
4.9	Recorrido en coche: Ruta 1	28
4.10	Recorrido en coche: Ruta 2	29
4.11	Recorrido en coche: Ruta 3	29
4.12	Recorrido en coche: Ruta 4	30
5.1	Comparativa entre alas originales y modificadas	31
5.2	Comparativa entre flaps de cola originales y modificados	32
5.3	Vuelo sin desviaciones: Cola modificada y alas originales	33
5.4	Trayectorias de vuelo 3D	33
5.5	Trayectorias de vuelo. Vista plano XY	34
5.6	Trayectorias de vuelo. Vista plano XZ	34
5.7	Evolución temporal de la actitud del ornitóptero en vuelo	35
5.8	Evolución temporal de la velocidad total del ornitóptero en vuelo)	35
5.9	Evolución temporal de las componentes de velocidad (NED)	36
5.10	Evolución temporal de las componentes de velocidad en ejes cuerpo (UVW)	37
5.11	Evolución temporal del ángulo de ataque	37

5.12	Velocidad total en trayectorias de vuelo	38
6.1	Raspberry Pi Zero con módulo de cámara acoplado	43

Índice de Tablas

1.1	Características GoGoBird	3
2.1	Características generales de VN-200 Rugged	6
2.2	Asignaciones de pines VN-200 Rugged	7
2.3	Comparativa de Alternativas de Antenas GNSS	13
3.1	Prestaciones de la Raspberry Pi Zero W	18
3.2	Características estructurales y eléctricas de la Raspberry Pi Zero W	18
4.1	Aportaciones de componentes al peso total	25
4.2	Resumen de pruebas. Verificación funcional del sistema	30
5.1	Resultados y características de pruebas de vuelo	39

Bibliografía

- [1] J. A. Moreno, C. Ruiz, A. C. Satue, J. Á. Acosta, and A. Ollero, “Design, development and testing of a hybrid fixed-flapping wing uav,” *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 329–338, 2022. [Online]. Available: <https://api.semanticscholar.org/CorpusID:251112643>
- [2] Álvaro G.Lora. (2023) Mecatronica, modelado y control del robot aereo de ala batiente hompot. Proyecto Fin de Grado. [Online]. Available: <https://github.com/aglora/HOMPOT>
- [3] VectorNav. (2024) Vectornav vn-200 documentation. Accedido: 2024-08-10. [Online]. Available: <https://www.vectornav.com/resources/by-product/vn-200>
- [4] ——, “Vectornav control center,” <https://www.vectornav.com/resources/software>, 2024, accedido: 2024-08-08.
- [5] ——. (2024) Vectornav programming library. Accedido: 2024-08-10. [Online]. Available: <https://www.vectornav.com/resources/programming-libraries>
- [6] Tallysman. (2024) Tw2712 antenna. Accedido: 2024-08-11. [Online]. Available: <https://sites.calian.com/app/uploads/sites/8/2024/06/Calian%C2%AE-TW2712-Datasheet-Rev.-202407.pdf>
- [7] ——. (2024) Tw1430 antenna. Accedido: 2024-08-11. [Online]. Available: <https://www.tallysman.com/app/uploads/2018/12/Tallysman%C2%AE-TW1430-Datasheet.pdf>
- [8] R. P. Foundation. (2024) Raspberry pi zero w. Accedido: 2024-08-12. [Online]. Available: <https://www.raspberrypi.com/products/raspberry-pi-zero-w/>
- [9] ——. (2024) Raspberry pi imager. Accedido: 2024-08-10. [Online]. Available: <https://www.raspberrypi.com/software/>
- [10] Eugeny. (2024) Tabbyterminal. Accedido: 2024-07-16. [Online]. Available: <https://github.com/Eugenyc/tabby>
- [11] B. Limited. (2024) Bitvise ssh client. Accedido: 2024-06-01. [Online]. Available: <https://bitvise.com/>
- [12] RealVNC. (2024) Realvnc viewer. Accedido: 2024-05-19. [Online]. Available: <https://www.realvnc.com/en/connect/download/viewer/>
- [13] Microsoft. (2024) Virtual studio code. Accedido: 2024-05-19. [Online]. Available: <https://code.visualstudio.com/>