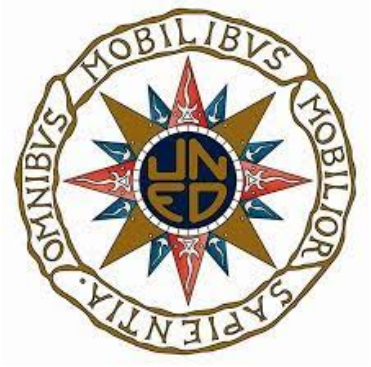

Universidad Nacional de Educación a Distancia

Escuela Técnica Superior de Ingeniería Informática

Máster Universitario en Ingeniería de Sistemas y de Control

Trabajo de Fin de Master

Implementación de algoritmos de estimación de la actitud en un microcontrolador Raspberry Pi Pico

---

Autor:
Enrique FLORES MONTOYA

Directoras:
Raquel DORMIDO
Natividad DURO

16 de julio de 2023

Resumen

El presente trabajo describe detalladamente la implementación de dos algoritmos de estimación de actitud a partir de las lecturas de una Unidad Inercial de Medida (IMU por sus siglas en inglés). En primer lugar, se presentan los formalismos teóricos para la descripción de la actitud de un sistema de referencia con respecto a otro. A continuación, se describe el montaje experimental y se analizan pormenorizadamente los modelos de error y los métodos de calibración de los diferentes sensores. Más adelante, se presentan los fundamentos teóricos de la estimación de la actitud mediante un filtro de Kalman y se detallan los aspectos prácticos de su implementación. Posteriormente, se desarrolla un algoritmo de fusión de datos que combina mediante un filtro complementario la estimación de la actitud del algoritmo TRIAD y la integración de las velocidades angulares.

Los resultados comprenden el desarrollo de un código en C/C++ que consta de varias librerías y que permite la lectura de los sensores inerciales y la implementación los algoritmos de estimación de la actitud. Las instrucciones del código se ejecutan en el microcontrolador Raspberry pi Pico y el acceso a sus variables se realiza mediante la comunicación por el puerto de serie. El código elaborado constituye un primer paso para el desarrollo de un vehículo autónomo no tripulado basado en este microcontrolador.

El último capítulo del documento presenta una primera aproximación a la siguiente etapa del proceso de desarrollo de un vehículo no tripulado. Este capítulo describe la implementación de un modelo no lineal de un quadrotor en MATLAB/Simulink, su linealización en torno a una condición de vuelo a punto fijo y el desarrollo de un algoritmo de control multivariable para el modelo. En primera instancia, la respuesta del controlador en lazo cerrado se evalúa mediante herramientas teóricas. Finalmente, se analiza la respuesta del controlador en lazo abierto a entradas reales generadas medianta el dispositivo experimental y utilizando los algoritmos de estimación de actitud desarrollados.

Agradecimientos

En primer lugar, me gustaría agradecer su labor como tutoras del trabajo de fin de máster a las profesoras Natividad Duro y Raquel Dormido. Gracias por aceptar mi propuesta y por tolerar mis retrasos en las entregas. Sin su colaboración y apoyo, este trabajo, y quizás mis estudios de máster, no habrían podido finalizar. Así mismo, quiero agradecer a la Universidad Nacional de Educación a Distancia el haber proveído los medios y la infraestructura para poder cursar mis estudios. Como *fan* de la educación autónoma y a distancia, ha sido una gran experiencia realizar mis estudios de máster en esta modalidad. Sin lugar a dudas, este tipo de eduación constituye una herramienta poderosa en un mundo cambiante y cada vez más dinámico como el de hoy en día.

Por otro lado, querría destacar el ejemplo de voluntad, constancia y determinación que mi hermana Isabel ha representado siempre para mí. Al mismo tiempo, quiero agradecer a mi hermano Antonio las constantes lecciones de programación y su actitud crítica y analítica frente a los problemas teóricos. Isa y Anto, me gustaría parecerme más a vosotros y menos a mí.

Finalmente, a mis padres. No tengo palabras para describir lo agradecido que estoy hacia vosotros. Han sido muchos los momentos de frustración, desasosiego y abatimiento de los que os he hecho partícipes sin merecerlo. Gracias por seguir creyendo en mi cuando yo mismo dejé de creer. Sin vosotros no lo habría conseguido.

Índice general

Índice de figuras

Nomenclatura

ψ	Ángulo de guiñada o *yaw*
θ	Ángulo de cabeceo o *pitch*
ϕ	Ángulo de balance o *roll*
$X_I Y_I$	plano formado por las direcciones x_I e y_I
R	Matriz de rotación
$\mathbf{q}$	cuaternio
$\boldsymbol{\omega}$	velocidad angular
$\boldsymbol{\Omega}$	vector de velocidad angular extendido
$S(\boldsymbol{\omega})$	matriz de propagación de cuaternios
$\tilde{S}(\mathbf{q})$	matriz de propagación de cuaternios alternativa
g	constante de aceleración de la gravedad
$\mathbf{a}$	aceleración del sistema
$\mathbf{g}$	vector de gravedad
$\mathbf{m}$	vector de campo magnético
$\mathbf{b}^g$	bias del giróscopo
$\mathbf{b}^a$	bias del acelerómetro
$\mathbf{b}^m$	bias del magnetómetro
$\mathbf{x}$	vector de estado
$\mathbf{y}$	salida del sistema
$\mathbf{u}$	entrada del sistema
$\mathbf{w}$	ruido de la planta
$\mathbf{v}$	ruido de los sensores
A	matriz de estado en tiempo continuo
B	matriz de control en tiempo continuo
C	matriz de observación en tiempo continuo
D	matriz de transmisión directa en tiempo continuo
F	matriz de transición en tiempo discreto
G	matriz de control en tiempo discreto
H	matriz de observación en tiempo discreto
$\overline{z}$	valor de referencia de la variable z
z'	perturbación de z respecto a $\overline{z}$
$\hat{z}$	estimación de la variable z también expresado como $\hat{z}_{k\|k}$
$\hat{z}^-$	estimación a priori también expresado como $\hat{z}_{k\|k-1}$

Operadores

$E()$	esperanza de una variable aleatoria
$V()$	varianza de una variable aleatoria
$\dot{()}$	derivada temporal

Subíndices

k	paso de tiempo

Superíndices

b	ejes cuerpo
I	ejes inerciales
T	matriz transpuesta

Preámbulo

En una reciente nota de prensa (26/05/2023), la compañía Nipona Ispace detallaba las causas del fracaso de la misión HAKUTO-R. La misión tenía como objetivo principal el aterrizaje con éxito de una sonda no tripulada en la superficie de la Luna. Desafortunadamente, la nave no consiguió realizar un aterrizaje controlado en la superficie del satélite. Los datos de telemetría revelaron que la sonda se encontraba en caída libre antes de que se perdiera el contacto con el vehículo. Un error ocurrido durante el descenso fue responsable de la pérdida de la nave.

Según revela el informe publicado tras el análisis de los datos, la causa del fallo en el aterrizador lunar fue una incorrecta estimación de su altitud con respecto a la superficie lunar. Al pasar por encima de un cráter, el sensor de altura midió un aumento repentino de la elevación del vehículo con respecto a la superficie. El sistema de control detectó una discrepancia de aproximadamente 3 km entre la altura medida y la altura estimada. Atribuyendo esta discrepancia a un mal funcionamiento del sensor, el medidor de altitud fue desconectado. A partir de ese momento, las lecturas de altitud fueron ignoradas y la estimación de la posición de la aeronave se basó únicamente en la integración de los sensores inerciales. La deriva en la estimación de la altitud causada por los errores de integración provocó una divergencia entre la altura estimada y la altura real del vehículo, lo que terminó provocando el fracaso de la misión de aterrizaje.

Este reciente ejemplo demuestra cómo los detalles de implementación de algoritmo pueden llegar a ser tan o más importantes como sus fundamentos teóricos. En la práctica, los sensores en cuyos datos se apoya un sistema de control tienen ruido, mediciones imperfectas o directamente fallan. Comprender la importancia de los dificultades prácticas es fundamental en cualquier aplicación tecnológica. La célebre frase del jugador de Béisbol estadounidense Yogi Berra resume a la perfección esta dicotomía entre teoría y practica,

In theory there is no difference between theory and practice. In practice there is

La necesidad de comprender las dificultades prácticas motiva el presente trabajo sobre los algoritmos de fusión de datos para el control de actitud en vehículos autónomos.

Capítulo 1

Introducción

1.1. Avances en UAVs and MAVs

Los vehículos autónomos o Unmanned Aerial Vehicles (UAVs) han experimentado un intenso desarrollo en las últimas décadas. Su avance se ha visto impulsado por las mejoras en las baterías y los motores sin escobillas, así como por el desarrollo de nuevos materiales compuestos y el abaratamiento y la miniaturización de la electrónica [1, 2, 3]. Los UAVs se emplean en diferentes aplicaciones, tanto civiles como militares: dragaminas (identificación de minas marinas), cobertura inalámbrica, mantenimiento de carreteras, planificación urbanística, prevención de desastres naturales y usos agrícolas. Se trata de un sector creciente cuyo valor de mercado se esperaba que alcanzase los 127 billones de dólares en 2020 [3]. Existe una amplia variedad de configuraciones de UAV que van desde las aeronaves de ala fija hasta los vehículos de ala batida pasando por dispositivos de ala rotatoria y de tipo globo. Los vehículos de ala fija se utilizan para cubrir grandes áreas de terreno dada su mayor autonomía y alcance. En contraste, los vehículos de ala rotatoria basados en el uso de hélices son más adecuados para interiores y zonas de baja accesibilidad gracias a su mayor maniobrabilidad.

Independientemente de su configuración alar, los UAVs comparten una serie de elementos comunes que se resumen en el diagrama de bloques de la figura 1.1. Todo sistema de vuelo autónomo consta de un vehículo no pilotado, una estación de control en tierra que puede ser autónoma u operada manualmente y un sistema de telemetría y telecomandos. Este último mantiene una comunicación entre la estación en suelo y el vehículo. Dentro de los elementos que conforman el vehículo, podemos distinguir los siguientes subsistemas,

1. **Planta de potencia y sistema de gestión de energía.** La mayoría de los vehículos de tamaño mediano y pequeño son alimentados por baterías. Los vehículos alimentados por baterías se caracterizan por una gran fiabilidad, baja firma térmica y nulas emisiones de contaminantes. Los recientes avances en baterías de Ión Litio-Polímero

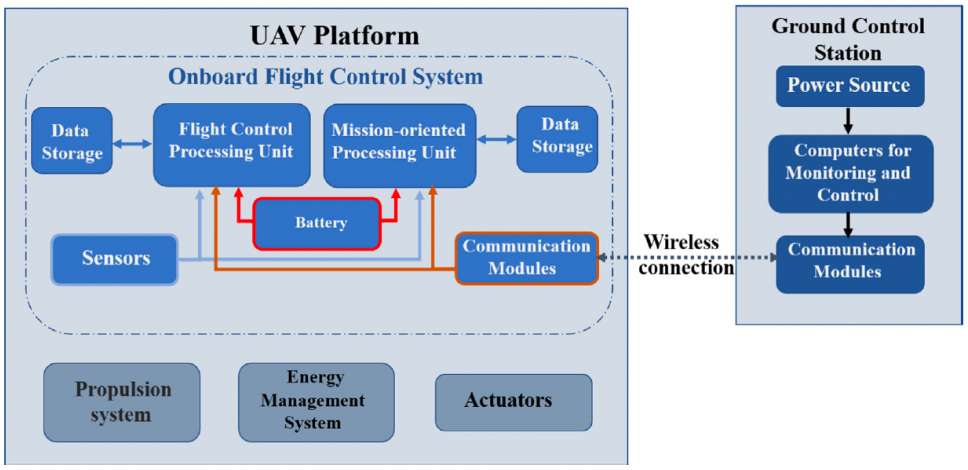

Figura 1.1: Diagrama de bloques de un UAV. (Fuente: [1]).

(LiPo) han permitido lograr autonomías de hasta 90 minutos. Alternativamente, los UAVs eléctricos pueden ser alimentados por pilas de combustible. Las pilas de combustible producen energía eléctrica a partir de hidrógeno y se caracterizan por una mayor densidad energética que las baterías. No obstante, presentan una menor potencia específica [1]. Otra de sus ventajas es que la recarga del depósito de combustible es mucho más rápida que la de las baterías. Finalmente, los vehículos de mayor tamaño emplean motores térmicos como fuente de potencia. Un análisis más detallado sobre las fuentes de potencia y las estrategias de gestión de energía puede encontrarse en [1]. De ahora en adelante, nos centraremos en el análisis vehículos eléctricos alimentados por baterías.

2. **Sistema de propulsión y actuadores.** En los UAVs, el sistema de propulsión transforma la energía eléctrica proporcionada por las baterías en energía mecánica que permite al UAV desplazarse y desempeñar su misión. Generalmente, los UAVs utilizan un sistema de propulsión a hélice para mantenerse en vuelo. En las aeronaves de ala rotatoria, el sistema de propulsión proporciona la fuerza de sustentación que permite al vehículo elevarse del suelo. Por el contrario, en las aeronaves de ala fija, la fuerza de sustentación se genera en las alas y el sistema de propulsión se limita a compensar la resistencia aerodinámica permitiendo al vehículo mantener una velocidad de avance. La transformación de la energía eléctrica en energía mecánica se realiza mediante motores sin escobillas de corriente continua. Las aeronaves de ala fija requieren además de un sistema de actuadores que permita accionar las superficies de control aerodinámico como los alerones y el timón de cola. En los UAVs eléctricos de pequeño tamaño esto puede lograrse mediante el uso de servomotores.

3. **Sistema de control a bordo.** El sistema de control de vuelo a bordo consta de, como mínimo, una unidad de procesado y control de vuelo, un módulo de comunicación

que permite al vehículo recibir órdenes de la estación en tierra y una serie de sensores que proporcionan al ordenador de a bordo información sobre el entorno y el estado del UAV. Además, el sistema de vuelo puede incluir una unidad de procesado de misión, un sistema de almacenamiento de datos y sensores adicionales. Estos últimos suelen estar relacionados con la misión del UAV y son independientes del control de vuelo *i. e.*, cámaras termográficas, medidores de la composición del aire, etc.

Dependiendo del objetivo de su misión, los UAVs modernos pueden ser generalmente operados tanto en modo manual como de manera completamente autónoma. Independientemente de como se realice la planificación de la trayectoria del vehículo a alto nivel, el control y la estabilización en vuelo requieren estimar su posición y su actitud para poder enviar las señales de control a los actuadores. Para ello, estos sistemas integran diferentes sensores tales como medidores de altitud barométricos, IMUs (Inertial Measurement Unit), GPSs y cámaras de baja resolución. No obstante, en general, se necesita procesar los datos de los sensores para estimar el estado del sistema. En particular, para la estimación de la actitud, la mayoría de los vehículos utilizan filtros de Kalman [4, 5, 6].

1.2. El filtro de Kalman

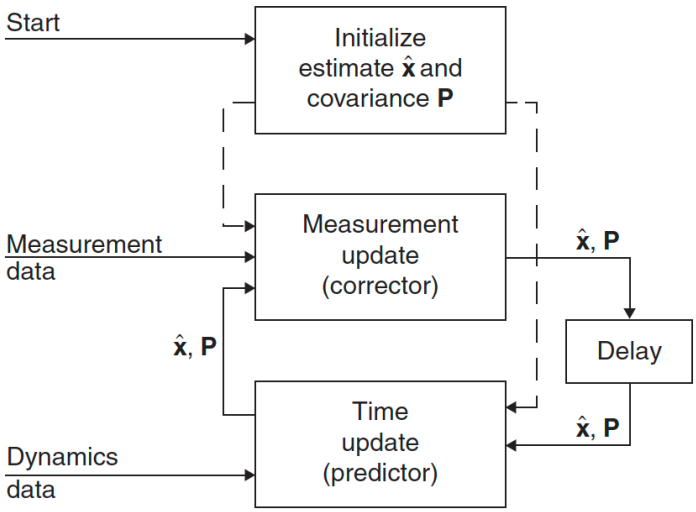

Figura 1.2: Diagrama de flujo de un filtro de Kalman. (Fuente: [7]).

El filtro de Kalman [8] fue desarrollado por Rudolf E. Kalman en 1960. En [9], Kalman describe una solución recursiva para el problema del filtrado lineal de datos discretos. Un filtro de Kalman es un estimador en tiempo real que se utiliza para trazar el estado de un

sistema dinámico con perturbaciones aleatorias a partir de mediciones ruidosas. El filtro es un procedimiento matemático que opera por medio de un mecanismo de predicción y corrección (Kalman Filter). En esencia, este algoritmo pronostica el nuevo estado a partir de su estimación previa añadiendo un término de corrección proporcional al error de predicción, de tal forma que este último es minimizado estadísticamente. La inicialización del filtro requiere asignar unos valores iniciales al vector de estados x y a su matriz de covarianza P, que representa la incertidumbre de la estimación inicial. Tras la inicialización, el algoritmo itera en bucle entre la fase de predicción y corrección.

En gran medida, su popularidad emerge de su buen comportamiento en un gran número de aplicaciones reales. La utilización de filtros de Kalman esta muy extendida tanto en el campo de la electrónica industrial como en la robótica [10, 11]. Uno de los primeros usos del filtro de Kalman fue el procesado de los datos para el guiado y el control de trayectoria de las naves en el programa Apollo. Desde ese momento, debido en gran parte al avance en el cálculo digital, el filtro de Kalman ha sido objeto de una extensiva investigación y aplicación, particularmente en el área de la navegación autónoma y asistida, en rastreo de misiles y en economía. A día de hoy, los filtros de Kalman constituyen una parte esencial de los sistemas de navegación modernos tanto inerciales como por satélite [7, 12]. El filtro se caracteriza por su bajo coste computacional y sus buenas propiedades recursivas. Es uno de los algoritmos de fusión de datos y sensores más comunes y más utilizados.

Debido al extenso volumen de trabajo dedicado a la investigación y el desarrollo de los filtros de Kalman, existe un gran número de variantes con diferentes dominios de aplicación. Entre ellas, podemos citar el filtro de Kalman extendido (EKF por sus siglas en inglés), el filtro de Kalman Unscented (UKF), el filtro de Kalman robusto (RKF) y el filtro de Kalman adaptativo (AKF). Una revisión sistemática de los diferentes tipos de filtros de Kalman puede encontrarse en el trabajo de Emer y Özbek [13].

No obstante, además del filtro de Kalman, existen otros algoritmos de fusión de datos que también se utilizan en la estimación de la actitud de vehículos. Entre ellos, cabe mencionar el filtro complementario. Este algoritmo realiza un filtrado pasa-baja de la estimación de la actitud obtenida de los datos del acelerómetro y combina estas medidas con un filtrado pasa-alta de la estimación de la actitud obtenida mediante la integración directa de las medidas del giróscopo.

El filtro complementario es más simple, ya que no considera ninguna descripción estadística del ruido de las señales y se basa únicamente en un análisis en el dominio de la frecuencia. Por ello, involucra menos operaciones, permitiendo una mayor frecuencia de iteración. Tal y como se demuestra en [14], en algunos escenarios el filtro complementario se comporta como un filtro de Kalman estacionario con ganancia constante. En la práctica, una gran número de aplicaciones emplean filtros complementarios lineales SISO (Single-Input-Single-Output). Por ejemplo, en [15] se presenta un filtro complementario no lineal para la estimación de la actitud de una aeronave de ala fija no tripulada. El trabajo de

Euston *et al.* [15] desarrolla un modelo de la aceleración no inercial de la nave que se utiliza para compensar la salida del acelerómetro y obtener unas lecturas del vector de gravedad sin sesgo. Finalmente, en [16] se compara el comportamiento de un filtro de Kalman extendido y de un filtro complementario para estimar la inclinación de un sistema integrado las medidas del acelerómetro y del giróscopo de una Unidad Inercial de Medida con sensores MEMS.

La unidad de procesado y control de vuelo es la encargada de estimar el estado de la aeronave. Para ello, esta unidad integra los datos recogidos por los distintos sensores mediante un algoritmo de fusión de datos como un filtro de Kalman o un filtro complementario. La información sobre la actitud del UAV es utilizada por el algoritmo de control para mantener la estabilidad y seguir la trayectoria prevista. En un sistema de navegación inercial, los datos recogidos por la IMU (Inertial Measuerement Unit) son procesados por el ordenador de vuelo para estimar la actitud del vehículo. La comunicación con los sensores y la implementación del algoritmo de fusión se realizan en la unidad de procesado cuyo soporte físico (hardware) es una computadora o un microcontrolador.

1.3. Microcontroladores

Un microcontrolador es un sistema embebido [17] que está programado de manera secuencial y que, a diferencia de un microprocesador, no requiere de un sistema operativo (OS) [18]. Los microcontroladores consisten en un circuito digital síncrono que incorpora una unidad central de procesado (CPU), una memoria que almacena el programa (memoria de programa) y una memoria que contiene los datos que se generan durante la ejecución del código (memoria de datos).

El uso de microcontroladores esta muy extendido en todo tipo de aplicaciones desde dispositivos médicos implantables, sistemas de procesado de control, dispositivos de instrumentación industrial y electrodomésticos [19]. Su utilización industrial se ha visto impulsada por su bajo consumo de potencia y por el abaratamiento de la electrónica debido al desarrollo de la tecnología de impresión de placas de circuitos.

Además, su gran versatilidad ha potenciado su uso para el prototipado y la educación [18]. A día de hoy, en el campo de la robótica existen un gran número de herramientas de propotipado accesibles al gran público tales como Arduino [19] o Raspberry pi [20]. Estos dispositivos se caracterizan por su flexibilidad, bajo coste y accesibilidad. Los microcontroladores de uso extendido como el Arduino Uno incorporan conectividad USB para la alimentación, la comunicación y la carga del código (bootloader). Todos estos aspectos, junto con la existencia de bibliotecas *ready to use* han facilitado su extensión y fomentado la filosofía del DIY (Do It Yourself).

De entre todos los microcontroladores de bajo coste, Arduino es la tecnología más

extendida, con una mayor comunidad de usuarios, soporte y bibliotecas. Existen diferentes modelos de circuitos impresos Arduino: Uno, Due, Mega, Nano, etc. El modelo más común es el Arduino Uno, que monta un procesador ATmega328 con una frecuencia de reloj de 16 MHz. La placa Arduino Uno cuenta con una memoria de programa de 32 kB, una memoria RAM de 2kB y una memoria EEPROM de 1kB. El microcontrolador posee 14 pines digitales de salida y entrada (I/O digital pins) y 6 pines de analógicos, además de dos pines de potencia a 5 V y 3.3 V. Las otras versiones de Arduino ofrecen un número diferente de entradas y salidas y/o niveles de memoria de programa y RAM marginalmente superiores [19].

Debido a sus limitadas prestaciones de memoria y capacidad de procesado, la tecnología Arduino no permite ni el uso extensivo de bibliotecas ni de elevadas frecuencias de ejecución. Esto limita enormemente la extensión, y por tanto, la complejidad y la sofisticación del código principal. La falta de memoria interna restringe, entre otros, la inclusión de bibliotecas para la implementación de un filtro de Kalman tales como (`Kalman_Filter_Library.h`). La utilización de cámaras de baja resolución que permitan, mediante el procesado de imágenes con herramientas de flujo óptico, la determinación de las velocidades de desplazamiento horizontal del vehículo queda totalmente descartada.

Otra de las fuertes limitaciones del microcontrolador Arduino Uno es que su chip, el ATMega328, que pertenece a la familia de microcontroladores AVR desarrollados por ATMEL [21], tiene una arquitectura de 8 bits. Esto implica que todas las operaciones básicas se realizan en paquetes de 8 bits. En aras de la comparación, las computadoras modernas y los celulares tienen típicamente arquitecturas de 64 y 32 bits respectivamente. Además, la mayoría de los microcontroladores del mercado carecen de Unidad de Procesado de Números de coma Flotante (FPU por sus siglas en inglés). Las operaciones con números de coma flotante están implementadas mediante software.

Por todas estas razones, la utilización de microcontroladores Arduino para el prototipado en vehículos autónomos condiciona enormemente sus prestaciones. En consecuencia, es necesario considerar el uso de microcontroladores con mayor capacidad de procesado y memoria interna. Además de Arduino, existen otros muchos microcontroladores disponibles en el mercado que se utilizan en aplicaciones de robótica tales como el ESP8266, el MSP430, el ESP32, el BBCmicrobit y el STM32. La Raspberry pi Pico es un microcontrolador recientemente desarrollado y comercializado por Raspberry. Las especificaciones detalladas pueden encontrarse en la sección 3. Su uso está menos extendido que el de Arduino pero su potencia y capacidad de procesado son mucho mayores. Esto lo convierte en un microcontrolador más adecuado para aplicaciones más exigentes donde se requieren mayores frecuencias de ejecución y un procesado de datos más complejo.

1.4. Objetivo

El presente trabajo sienta las bases para el futuro desarrollo de un vehículo autónomo basado en el microcontrolador Raspberry pi Pico. En particular, se aborda la sensorización de un sistema de control de actitud. Para ello, se implementan dos algoritmos de estimación de actitud de diferente nivel de complejidad y se desarrollan las bibliotecas de C/C++ necesarias para su ejecución en el microcontrolador. La programación de este dispositivo requiere un mayor esfuerzo inicial debido a,

- su mayor complejidad y flexibilidad,
- la necesidad de construir un entorno para el *bootloading*,
- su menor implantación, que se traduce en un menor nivel de documentación.

No obstante, la Raspberry pi Pico es un microcontrolador con prestaciones suficientes para alojar un algoritmo de control, un protocolo de telecomandos-telemetría y una sensorización avanzada. Esto permitirá extender las funcionalidades de un futuro vehículo autónomo desarrollado en base a este microcontrolador. El trabajo busca abarcar todo el proceso de implementación del control de actitud desde el más bajo nivel: acceso a los registros de memoria del sensor; hasta un nivel medio-alto: diseño preliminar y evaluación en lazo abierto de un control de actitud.

El documento se estructura de la siguiente forma: el capítulo 2 presenta los fundamentos teóricos, el capítulo 3 describe el montaje experimental y el capítulo 4 aborda la calibración de los sensores. Los capítulos 5 y 6 presentan dos algoritmos de estimación de actitud a partir de las lecturas de una IMU. Finalmente, el capítulo 7 describe el ajuste de un control de actitud descentralizado en un quadrotor y utiliza las lecturas del sensor en la planta real para evaluar la respuesta del controlador en lazo abierto.

Capítulo 2

Fundamentos Teóricos

2.1. Sistemas de referencia

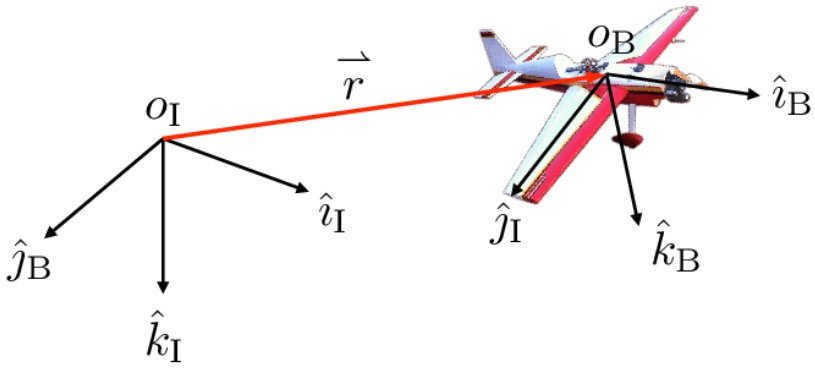

Figura 2.1: Sistemas de referencia inercial y ejes cuerpo. Fuente: [22].

En esta sección se presentan los sistemas de referencia empleados para definir la actitud de un vehículo aéreo. Los sistemas de referencia se denotan, de forma genérica por, $F(O, x, y, z)$ donde (O) indica el origen y (x, y, z) son los ejes del sistema de referencia. En la definición de la actitud de un vehículo es necesario considerar dos sistemas de referencia relevantes.

2.1.1. Sistema de referencia inercial: (O_I, x_I, y_I, z_I)

Es un sistema de referencia cartesiano inercial centrado en un punto arbitrario de la superficie terrestre, O_I. El eje x_I apunta en la dirección del norte geodésico, el eje y_I apunta en la dirección del este geodésico y el eje z_I apunta en la dirección del centro de la tierra. Este sistema de referencia, en ocasiones denotado por NED (North-East-Down) se utiliza

a menudo para la navegación en aeronaves de pequeño tamaño. Generalmente, se toma el punto de despegue del vehículo como origen y se utiliza este sistema de referencia para describir la posición, la velocidad y la aceleración del vehículo con respecto a este punto.

2.1.2. Sistema de referencia ejes cuerpo: (O_b, x_b, y_b, z_b)

El sistema de referencia ejes cuerpo (O_b, x_b, y_b, z_b) esta centrado en el centro de gravedad del vehículo. El eje x_b apunta hacia adelante y está contenido en el plano de simetría del vehículo, el eje y_b apunta a estribor (hacia la derecha desde el punto de vista de un observador que mira en la dirección del eje x_b) y el eje z_b apunta hacia abajo, en dirección al suelo, y está también contenido en el plano de simetría del vehículo.

En los vehículos aéreos, las rotaciones alrededor de los ejes cuerpo se conocen respectivamente como balance, cabeceo y guiñada o *roll, pitch* y *yaw* por sus nombres en inglés. Estos ángulos describen la orientación del vehículo con respecto a los ejes inerciales (O_I, x_I, y_I, z_I), lo que se conoce comúnmente como actitud del vehículo.

2.2. Formalismos para describir la actitud

Las rotaciones permiten pasar de unos sistemas de referencia a otros. Existen diferentes formalismos que se emplean para representar las transformaciones entre sistemas de referencia. El método más intuitivo para describir la orientación o actitud de un cuerpo en el espacio son los ángulos de Euler según la convención de Tait-Bryan. Sin embargo, los ángulos de Euler tienen ciertas limitaciones que hacen que los cuaternios sean el formalismo adoptado en los sistemas de estimación de la actitud. La descripción de la actitud mediante cuaternios puede traducirse a ángulos de Euler, lo que permite obtener un resultado más fácilmente interpretable.

2.2.1. Ángulos de Euler

Fundamentos

La forma más intuitiva de describir la actitud de un cuerpo en el espacio es mediante los ángulos de Euler. La orientación de un sistema de referencia con respecto a otro puede describirse mediante tres rotaciones sucesivas. Las rotaciones se representan mediante matrices ortonormales 3×3. La composición de rotaciones se obtiene multiplicando matrices de rotación individuales y la transformación inversa viene dada por la matriz transpuesta de la matriz de rotación. El orden en el que se realizan las rotaciones alrededor de los diferentes ejes se denomina convención. Nótese que las rotaciones finitas no son conmuta-

tivas. Existen diferentes convenciones de ángulos de Euler. La convención utilizada para describir la orientación de aeronaves y vehículos espaciales es la convención de Tait-Bryan zyx. En otros campos científicos como la Física de Partículas o la Astronomía, son usuales convenciones distintas como la zyz o la zxz. En aviónica, los ángulos de Euler se utilizan para describir la relación entre los ejes inerciales $-I$ y los ejes cuerpo $-b$,

- **Yaw** ψ: ángulo de rotación alrededor del eje z_I del sistema de referencia inercial. El rumbo del vehículo viene dado por el ángulo formado entre la proyección el eje x_b del vehículo sobre el plano $X_I Y_I$ y el eje x_I. La matriz de rotación asociada es,

$$R_\psi = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

- **Pitch** θ: ángulo formado entre el plano $X_I Y_I$ y el eje x_b. La matriz de rotación correspondiente a este giro es,

$$R_\theta = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\psi & 0 & \cos\psi \end{bmatrix} \tag{2.2}$$

- **Roll** ϕ: ángulo de rotación alrededor del eje x_b. La matriz de rotación asociada a esta transformación viene dada por,

$$R_\phi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \tag{2.3}$$

La matriz de rotación global se obtiene mediante la composición de las tres rotaciones *yaw, pitch, roll* y viene dada por la multiplicación de las matrices R_ψ, R_θ y R_ϕ. De esta manera, la matriz de rotación que permite transformar las componentes de un vector expresadas en ejes inerciales a ejes cuerpo viene dada por,

$$R_I^b = R_\phi R_\theta R_\psi \tag{2.4}$$

donde el orden de las rotaciones tiene lugar de derecha a izquierda. La expresión de la matriz de rotación resultante es,

$$R_I^b = \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\psi \\ \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \sin\psi\sin\theta\sin\psi + \cos\phi\cos\psi & \sin\phi\cos\theta \\ \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\sin\psi & \cos\phi\cos\theta \end{bmatrix} \tag{2.5}$$

dado que la matriz R_I^b es ortonormal, la transformación de un vector expresado en ejes cuerpo (O_b, x_b, y_b, z_b) a ejes inerciales (O_I, x_I, y_I, z_I) se realiza multiplicando por la matriz R_b^I, que satisface,

$$R_b^I = (R_I^b)^{-1} = (R_I^b)^T \tag{2.6}$$

Propagación de ángulos de Euler

El vector de velocidad angular del cuerpo con respecto al sistema de referencia inercial expresado en ejes cuerpo viene dado por,

$$\boldsymbol{\omega} = \begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix} \tag{2.7}$$

La relación entre la velocidad angular y la variación temporal de los ángulos de Euler viene dada por,

$$\begin{aligned} \omega_x &= \dot{\phi} - \dot{\psi}\sin\theta & (2.8)\\ \omega_y &= \dot{\theta}\cos\phi + \dot{\psi}\cos\theta\sin\phi & (2.9)\\ \omega_z &= -\dot{\theta}\sin\phi + \dot{\psi}\cos\theta\cos\phi & (2.10) \end{aligned}$$

y la relación inversa se expresa como,

$$\begin{aligned} \dot{\phi} &= \omega_x + (\omega_y\sin\phi + \omega_z\cos\phi)\tan\theta & (2.11)\\ \dot{\theta} &= \omega_y\cos\phi - \omega_z\sin\phi & (2.12)\\ \dot{\psi} &= (\omega_y\sin\phi + \omega_z\cos\phi)\frac{1}{\cos\theta} & (2.13) \end{aligned}$$

Nótese que la ecuación (2.13) es singular cuando el ángulo de cabeceo (pitch) es $\theta = \pm\pi/2$. Esto limita el uso de los ángulos de Euler para representar la actitud de un cuerpo haciendo que los cuaternios unitarios sean la alternativa escogida por la mayoría de sistemas de integración de la actitud.

2.2.2. Cuaternios

Fundamentos

Los cuaternios son un formalismo que permite describir la actitud de un cuerpo y las rotaciones entre diferentes sistemas de referencia. Un cuaternio esta formado por cuatro componentes,

$$\mathbf{q} = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix} \tag{2.14}$$

El cuaternio también puede expresarse como un número complejo con una componente real y tres componentes imaginarias,

$$\mathbf{q} = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k} \tag{2.15}$$

El producto entre dos cuaternios $\mathbf{q_1} = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix}$ y $\mathbf{q_2} = \begin{bmatrix} a & b & c & d \end{bmatrix}$ puede expresarse como el producto de matrices,

$$\mathbf{q_1} \cdot \mathbf{q_2} = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \tag{2.16}$$

Los cuaternios pueden usarse para describir las rotaciones entre sistemas de referencia. Un cuaternio de componentes $\begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix}$ representa un giro de un ángulo α alrededor de un eje $\mathbf{t} = \begin{bmatrix} t_x & t_y & t_z \end{bmatrix}$,

$$\mathbf{q} = \left[\cos\frac{\alpha}{2},\ \mathbf{t} \sin\frac{\alpha}{2} \right] \tag{2.17}$$

Sea $\mathbf{r}^b$ un vector cualquiera expresado en ejes cuerpo, este puede escribirse como un cuaternio cuya componente escalar es nula,

$$\mathbf{r}^b = 0 + r_x\mathbf{i} + r_y\mathbf{j} + r_z\mathbf{k} \tag{2.18}$$

Para expresar el vector $\mathbf{r}^b$ en ejes inerciales se aplica,

$$\mathbf{r}^I = \mathbf{q} \cdot \mathbf{r_b} \cdot \mathbf{q}^\star \tag{2.19}$$

donde $\mathbf{q}^\star$ denota el cuaternio conjugado de $\mathbf{q}$. Esta operación también puede expresarse de forma matricial como,

$$\mathbf{r}^I = \begin{bmatrix} 0 & 0 \\ 0 & R_b^I \end{bmatrix} \mathbf{r}^b \tag{2.20}$$

donde R_b^I es la matriz de rotación entre el sistema de referencia ejes cuerpo y el sistema inercial definida en (2.6). La matriz R_b^I expresada en función de las componentes de $\mathbf{q}$ viene dada por,

$$R_b^I = \begin{bmatrix} (q_0^2 + q_1^2 - q_2^2 - q_3^2) & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & (q_0^2 - q_1^2 + q_2^2 - q_3^2) & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & (q_0^2 - q_1^2 - q_2^2 + q_3^2) \end{bmatrix} \tag{2.21}$$

de nuevo, la matriz R_b^I es ortonormal y su transpuesta puede utilizarse para rotar un vector expresado en el sistema de referencia inercial al sistema de referencia ejes cuerpo.

Propagación de cuaternios

Dado un sistema de ejes cuerpo $-b$ que rota con un vector velocidad angular $\boldsymbol{\omega} = \begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix}$ con respecto al sistema de referencia inercial $-I$ y sea $\mathbf{q}$ el cuaternio que

describe la orientación del sistema de referencia ejes cuerpo con respecto al sistema de referencia inercial, la variación temporal de $\mathbf{q}$ viene dada por,

$$\dot{\mathbf{q}} = \frac{1}{2}\mathbf{q} \cdot \mathbf{\Omega} \tag{2.22}$$

donde $\mathbf{\Omega}$ se define como,

$$\mathbf{\Omega} = \begin{bmatrix} 0 \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{2.23}$$

La ecuación (2.22) puede expresarse en forma matricial como,

$$\dot{\mathbf{q}} = \frac{1}{2}\begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix}\begin{bmatrix} 0 \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \frac{1}{2}\begin{bmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & -q_3 & q_2 \\ q_3 & q_0 & -q_1 \\ -q_2 & q_1 & q_0 \end{bmatrix}\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \frac{1}{2}\tilde{S}(\mathbf{q})\boldsymbol{\omega} \tag{2.24}$$

Esta expresión puede reescribirse para expresarla como el producto entre una matriz $S(\boldsymbol{\omega})$ y el cuaternio $\mathbf{q}$,

$$\dot{\mathbf{q}} = \frac{1}{2}\begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix}\begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \frac{1}{2}S(\boldsymbol{\omega})\mathbf{q} \tag{2.25}$$

La integración de la ecuación (2.25) en el tiempo proporciona la evolución de la orientación del cuerpo descrita por el cuaternio $\mathbf{q}$ en función de la velocidad angular del sistema $-b$ con respecto al sistema inercial $-I$, $\boldsymbol{\omega}$.

Relación entre los cuaternios y los ángulos de Euler

El cuaternio $\mathbf{q}$ que representa la orientación del sistema ejes cuerpo $-b$ con respecto al sistema inercial $-I$ puede expresarse en función de los ángulos de Euler (ϕ, θ, ψ) como,

$$q_0 = \cos\frac{\phi}{2}\cos\frac{\theta}{2}\cos\frac{\psi}{2} + \sin\frac{\phi}{2}\sin\frac{\theta}{2}\sin\frac{\psi}{2} \tag{2.26}$$

$$q_1 = \sin\frac{\phi}{2}\cos\frac{\theta}{2}\cos\frac{\psi}{2} - \cos\frac{\phi}{2}\sin\frac{\theta}{2}\sin\frac{\psi}{2} \tag{2.27}$$

$$q_2 = \cos\frac{\phi}{2}\sin\frac{\theta}{2}\cos\frac{\psi}{2} + \sin\frac{\phi}{2}\cos\frac{\theta}{2}\sin\frac{\psi}{2} \tag{2.28}$$

$$q_3 = \cos\frac{\phi}{2}\cos\frac{\theta}{2}\sin\frac{\psi}{2} - \sin\frac{\phi}{2}\sin\frac{\theta}{2}\cos\frac{\psi}{2} \tag{2.29}$$

Igualmente, a partir del cuaternio $\mathbf{q}$ que describe la orientación del sistema $-b$ podemos obtener los ángulos de Euler que definen la actitud del cuerpo,

$$\phi = \arctan\left[\frac{2(q_2q_3 + q_0q_1)}{1 - 2(q_1^2 + q_2^2)}\right] \tag{2.30}$$

$$\theta = \arcsin\left[2(q_0q_2 - q_1q_3)\right] \tag{2.31}$$

$$\psi = \arctan\left[\frac{2(q_1q_2 + q_0q_3)}{1 - 2(q_2^2 + q_3^2)}\right] \tag{2.32}$$

Las funciones arcotangente implementadas en los ordenadores solo producen resultados entre $\pm\pi/2$. Para poder generar todas las orientaciones es necesario sustituir la función `atan` por `atan2`, que tiene en cuenta el signo del numerador y del denominador. De nuevo, estas expresiones son singulares para $\theta \to \pm\pi/2$.

Capítulo 3

Dispositivo experimental

El dispositivo experimental utilizado para la implementación de los algoritmos de estimación de actitud se representa en la figura 3.1. El circuito se monta sobre una placa de prototipado e incluye dos microcontroladores Raspberry Pi pico y la Unidad de Medición Inercial (IMU por sus siglas en inglés) MPU9250. A continuación se describen en detalle las características de los diferentes elementos del circuito.

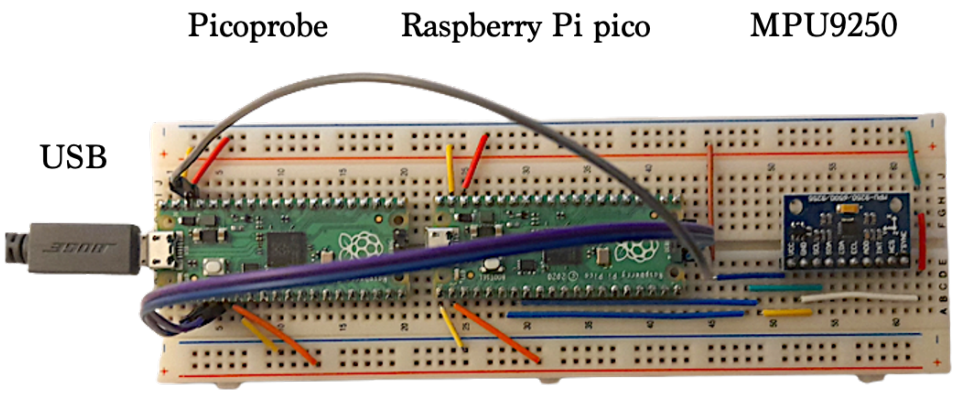

Figura 3.1: Montaje experimental

3.1. Raspberry Pi Pico

3.1.1. Especificaciones

La Raspberry pi Pico (`https://www.raspberrypi.com/products/raspberry-pi-pico/`) es un microcontrolador de alto rendimiento de tipo placa basado en el chip RP2040 diseñado por Raspberry Pi [23]. El microcontrolador Raspberry pi Pico (RpP) embarca un

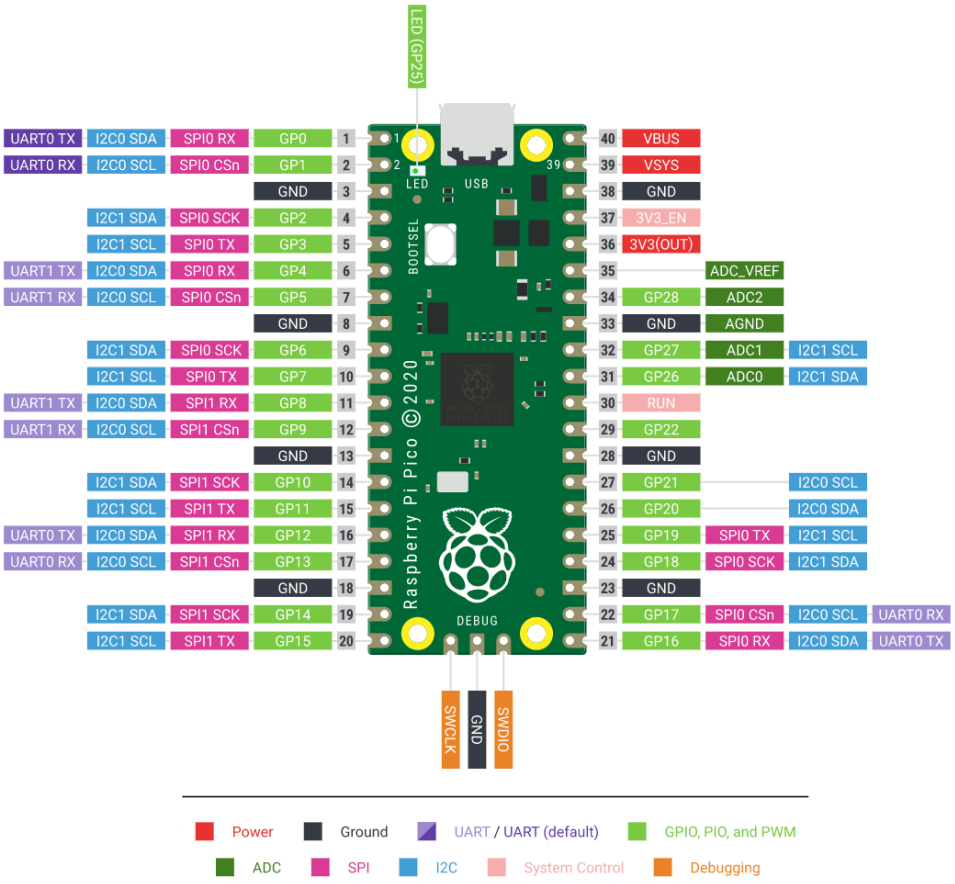

Figura 3.2: Raspberry pi Pico Pinout

procesador de doble núcleo dual-core ARM-CORTEX-M0+ (hasta 133 MHz) con una memoria SRAM interna de 264kB. La arquitectura del procesaor ARM-CORTEX-M0+ es de 32 bits. La RpP cuenta con 2MB de memoria flash y un chip de *power supply* que soporta voltajes de entrada de entre 1.8 y 5.5 V. El microcontrolador (ver figura 3.2) presenta 26 pines GPIO (General Purpose In-Out) incluyendo pines programables y soportando los protocolos de comunicación I^2C y SPI.

3.1.2. Librerías

La Rapsberry pi Pico puede programarse en MicroPython [24] o en C/C++ [25]. En el presente trabajo se ha optado por la programación en C/C++. Para empezar a programar la Raspberry pi Pico en C/C++ es necesario definir un *environment* e instalar una serie de librerías tales como SDK, OpenOCD, GDB y picoprobe. El proceso de instalación y los pasos a seguir para configurar el *environment* están descritos en [25].

La librería más importante es la librería SDK del chip RP2040 (`https://raspberrypi.`

`github.io/pico-sdk-doxygen/`) [25, 26]. La librería SDK (Software Development Kit) proporciona *headers*, funciones y métodos para escribir programas en dispositivos basados en el chip RP2040 tales como la Raspberry pi Pico. Estos métodos permiten controlar distintos elementos del hardware de la Raspberry pi Pico tales como los buses I2C, SPI y los *timers*.

Un único programa se ejecuta en el dispositivo cada vez bajo el método convencional `main()`. El sistema soporta las librerías estándar de C/C++ e incluye múltiples APIs (Application Programming Interface) para acceder al hardware del RP2040, incluyendo DMA, IRQs y una amplia variedad de pines programables y periféricos de función definida. Así mismo, la SDK provee librerías de alto nivel para manejar temporizadores, USB, sincronizaciones y programación multi-núcleo. La librería SDK utiliza CMake para compilar.

3.1.3. Picoprobe

Los archivos compilados pueden cargarse en el microcontrolador de dos formas. En primer lugar, es posible conectar la Raspberry pi Pico como si fuese un dispositivo de almacenamiento USB manteniendo pulsado el botón de BOOTSEL y conectando el cable USB. De esta manera podremos cargar los archivos compilados en la Raspberry copiándolos en la carpeta correspondiente. A continuación, deberemos desconectar el dispositivo y volver a enchufarlo para que el código cargado se ejecute. Este proceso es simple pero tedioso si se desean realizar pequeños cambios del código de manera constante.

La segunda opción es utilizar un segundo controlador Raspberry pi Pico conectado al ordenador por el puerto USB y a la Raspberry por el puerto UART. Esto permite utilizar `openOCD` para cargar los archivos compilados y resetear el microcontrolador desde la línea de comandos del ordenador. Este método, denominado `picoprobe` permite reprogramar el dispositivo de manera mucho más rapida y eficiente y es el que se ha adoptado en el presente trabajo. Los detalles pueden encontrarse en el Apéndice A de [25]. El circuito de la figura 3.1 muestra dos microcontroladores Raspberry pi Pico conectados en la configuración de `picoprobe`.

3.2. Sensor MPU 9250

El sensor MPU 9250 [27] es un módulo multi-chip que consiste en en dos *dies* integrados en un único paquete QFN (ver figura 3.3). El primero de los *dies* alberga un giróscopo y un acelerómetro de tres ejes. El segundo *die* contiene el magnetómetro de tres ejes AK8963. Por tanto, el sensor MPU 9250 es un dispositivo de *Motion Tracking* de 9 ejes que combina un giróscopo de 3 ejes, un acelerómetro de 3 ejes y un magnetómetro de 3 ejes así como un Procesador de Movimiento Digital (DMP).

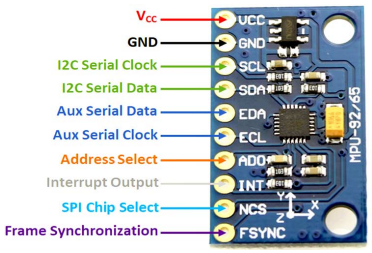

Figura 3.3: Sensor MPU 9250

El MPU9250 puede conectarse a otros dispositivos mediante los buses de comunicación I^2C (frecuencia de acceso a registros de memoria de 400 kHz) y SPI (frecuencia de acceso a registros de memoria de 1 MHz). El voltaje de operación del dispositivo es de 2.4 – 3.6 V. El sensor cuenta con nueve conversores analógico-digitales de 16 bits para muestrear las lecturas del giróscopo, el acelerómetro y el magnetómetro. Para la detección precisa de movimientos rápidos y lentos, los diferentes sensores cuentan con distintos rangos de medida programables: ±250, ±500, ±1000, ±2000°/sec (dps) para el giróscopo, y ±2 g, ±4 g, ±8 g, ±16 g para el acelerómetro. Si no se especifica lo contrario, en lo que sigue los rangos de medida programados para el giroscopio y el acelerómetro son de ±1000°/sec y ±4 g respectivamente. El magnetómetro tiene una rango de medida fijo de $\pm 4800 \mu T$. El sensor cuenta con filtros digitales programables integrados, un reloj de precisión con una deriva del 1 % en el rango de temperatura 40° C – 85° C, un sensor de temperatura e *interrupts* programables. Se trata de un sensor de bajo coste (a partir de 1.59 euros) lo que lo convierte en un buen candidato para la estimación de la actitud en vehículos autónomos de pequeño tamaño y en aplicaciones académicas y de prototipado.

La comunicación entre el MPU9250 y el microcontrolador se realiza mediante el protocolo de comunicación I^2C [28]. En el contexto del presente trabajo, se ha desarrollado una librería que permite acceder a las lecturas de los sensores que integran la Unidad de Medida Inercial. Esta librería se denomina `IMU.h` y permite inicializar los parámetros del acelerómetro y del giróscopo, así como aplicar la calibración determinada en el capítulo 4 a las medidas de los sensores. El acceso a los registros de memoria del sensor se realiza mediante las funciones `I2CwriteByte` e `I2Cread`. Los headers de esta librería pueden encontrarse en la sección 4.4

3.3. Entorno de validación

Durante la realización del presente trabajo, no se disponía de un laboratorio con suficientes medios experimentales para validar los algoritmos de control implementados. Se ha considerado el desarrollo de un dispositivo experimental que permitiese comparar la actitud estimada con la actitud real. Este dispositivo se construiría utilizando encoders

angulares orientados según las tres direcciones del espacio. Sin embargo, los encoders angulares disponibles a bajo coste tiene una muy baja resolución $\Delta\alpha \simeq 18°$. Esto obliga al uso de ruedas dentadas para desmultiplicar el movimiento de la planta. En consecuencia, la complejidad del montaje aumenta de forma sustancial, dejándolo fuera del alcance del presente trabajo.

No obstante, una mínima validación, aunque sea cualitativa, es necesaria. Para ello, se ha utilizado un código de python que permite leer los cuaternios de actitud calculados por el algoritmo que se ejecuta en la Raspberry pi Pico. Estos datos se envían por el puerto de serie en paquetes de `uint_8` y se leen con la librería `pyserial`. El script permite visualizar en tiempo real la orientación del dispositivo experimental. La representación virtual de la protoboard adopta la actitud calculada por el algoritmo que se ejecuta en la Raspberry (ver figura 3.4). Este método permite comparar de forma cualitativa la actitud de la planta con la actitud estimada. Ejemplos de la utilización del entorno de validación pueden encontrarse en los capítulos 5 y 6.

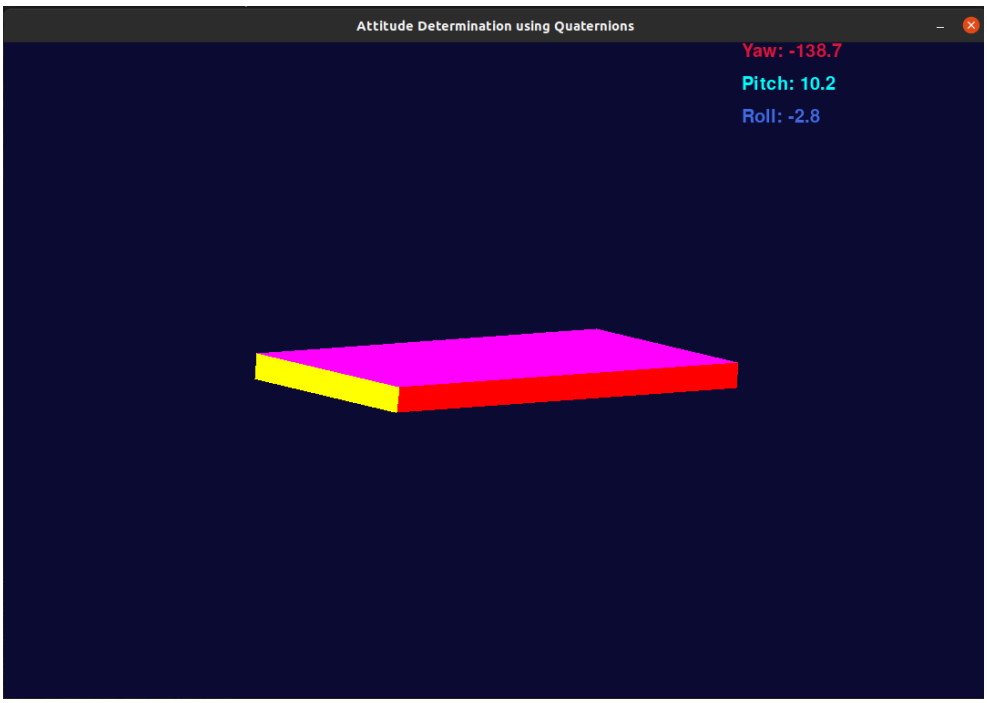

Figura 3.4: Entorno de validación en `python`

3.4. Header de la librería `matrix_operations.h`

La implementación de los algoritmos de estimación de la actitud de los capítulos 5 y 6 requiere de múltiples operaciones matriciales tales como la multiplicación, la transposición y la inversión de matrices. En el ámbito de los algoritmos de control destinados a pequeños microcontroladores, es habitual la elaboración de una librería matemáticas adaptada a las necesidades del código que implemente operaciones matriciales complejas [29, 30, 31, 32]. Esto permite las extensión de las funcionalidades más allá de las muy básicas operaciones matemáticas que permiten las bibliotecas estándar de C/C++. Además, evita el uso de extensas librerías como `Eigen.h` minimizando la extensión del código, el cual se encuentra siempre sujeto a restricciones significativas de memoria.

Siguiendo con esta práctica, se ha desarrollado una pequeña biblioteca de C++ que implementa algunas operaciones matriciales que aparecen con frecuencia en los algoritmos de estimación de actitud. Esta librería hace uso de la biblioteca estándar de C++ `<vector>` para la representación de matrices y vectores n–dimensionales. El header de esta biblioteca se reproduce a continuación.

```
#ifndef MATRIX_OPERATIONS_H
#define MATRIX_OPERATIONS_H
#include <vector>
//-----------------------------------------------------------------------------
// MATRIX OPERATIONS LIBRARY
//-----------------------------------------------------------------------------
// Author:  Enrique Flores
// Date:    14/07/2023

// Print a N x N matrix
void printMatrix(const std::vector<std::vector<double>>& matrix);
// Perform LU decomposition of a N x N matrix
void luDecomposition(const std::vector<std::vector<double>>& A,
                     std::vector<std::vector<double>>& L,
                     std::vector<std::vector<double>>& U);
// Check whether the a matrix has Inf or Nans
bool hasInfOrNaN(const std::vector<std::vector<double>>& matrix);
// Forward subsitution for matrix inversion
std::vector<double> forwardSubstitution(const std::vector<std::vector<double>>& L,
                                        const std::vector<double>& b);
// Backward substitution for matrix inversion
std::vector<double> backwardSubstitution(const std::vector<std::vector<double>>& U,
                                         const std::vector<double>& y);
// Product between a matrix and a column vector
std::vector<double> matrixVectorProduct(const std::vector<std::vector<double>>& matrix,
                                        const std::vector<double>& vector);
// Concatenate 1D vectors
std::vector<double> concatenateVectors(const std::vector<double>& v1,
```

```
                                    const std::vector<double>& v2);
// Create a diagonal matrix
std::vector<std::vector<double>> createDiagMatrix(int size, double diag);
// Transpose a matrix
std::vector<std::vector<double>> transposeMatrix(
                                const std::vector<std::vector<double>>& matrix
                                );
// Multiply two matrices
std::vector<std::vector<double>> matrixMultiplication(
                                const std::vector<std::vector<double>>& A,
                                const std::vector<std::vector<double>>& B
                                );
// Compute the inverse of a matrix
std::vector<std::vector<double>> matrixInverse(
                                const std::vector<std::vector<double>>& A
                                );

#endif // MATRIX_OPERATIONS_H
```

Capítulo 4

Caracterización de los sensores

4.1. Calibración del giróscopo.

El MPU9250 utiliza la tecnología Sistema Micro-Electro-Mecánico (MEMS por sus siglas en inglés) para medir la velocidad angular y la aceleración [27]. El término sensor MEMS se aplica a dispositivos miniaturizados fabricados mediante técnicas propias de la industria de la microelectrónica y que permiten medir una fuerza o una magnitud derivada como una aceleración, una presión o una velocidad angular [33, 34]. La transducción mecánica se consigue mediante materiales piezorresistivos, materiales piezoeléctricos o técnicas capacitivas. La medida de la velocidad angular se obtiene gracias al efecto Coriolis que consiste en la aparición de una fuerza ficticia cuando un cuerpo se desplaza con respecto a un sistema de referencia en rotación (ver figura 4.1).

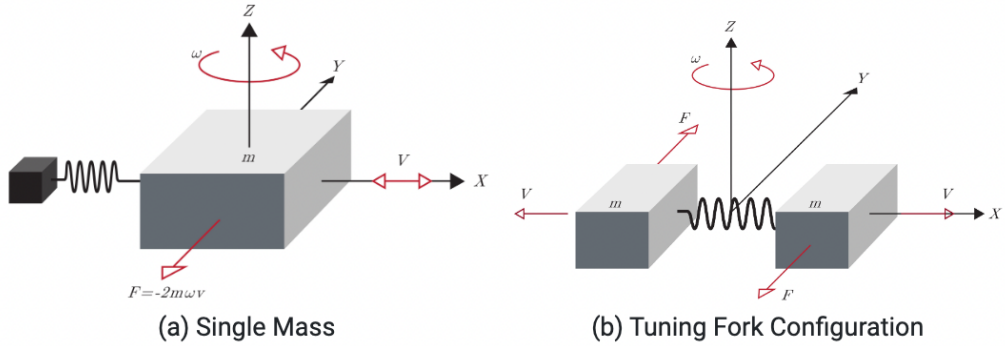

Figura 4.1: Modelo de un giróscopo. Fuente: VECTORNAV: MEMS Operation

El giróscopo proporciona las componentes del vector de velocidad angular, $\boldsymbol{\omega}$, en ejes cuerpo, $\begin{bmatrix}\omega_x & \omega_y & \omega_z\end{bmatrix}$. En una situación ideal, la orientación del cuerpo expresada en forma de ángulos de Euler podría obtenerse integrando las ecuaciones (2.11) – (2.13) a partir

de unas condiciones iniciales. Sin embargo, debido al ruido presente en las medidas del giróscopo, la integración directa de la velocidad angular presenta una deriva temporal, esto es, el valor de la integral diverge con respecto al valor real de la magnitud que se busca estimar.

La calibración del giróscopo consiste en determinar los valores del bias (sesgo), lo que permite reducir la deriva sistemática de las medidas del sensor. Existen otras técnicas de calibración más sofisticadas tales como las descritas en [35, 36]. No obstante, estos métodos de calibración más avanzados, que permiten corregir la desalineación entre los ejes y los factores de escala, están fuera del alcance del presente trabajo. Para obtener los valores de bias del giróscopo se miden los valores de la velocidad angular en los tres ejes con el sensor estático. El bias en cada dirección se determina calculando el valor medio de la señal. En este caso, la calibración del giróscopo se ha realizado utilizando un tiempo de muestreo de 50 ms y un rango de medida de $\pm 1000\,^{\circ}/\mathrm{s}$. La figura 4.2 representa los valores de la velocidad angular medidos durante un intervalo de tiempo de 30 s con y sin bias.

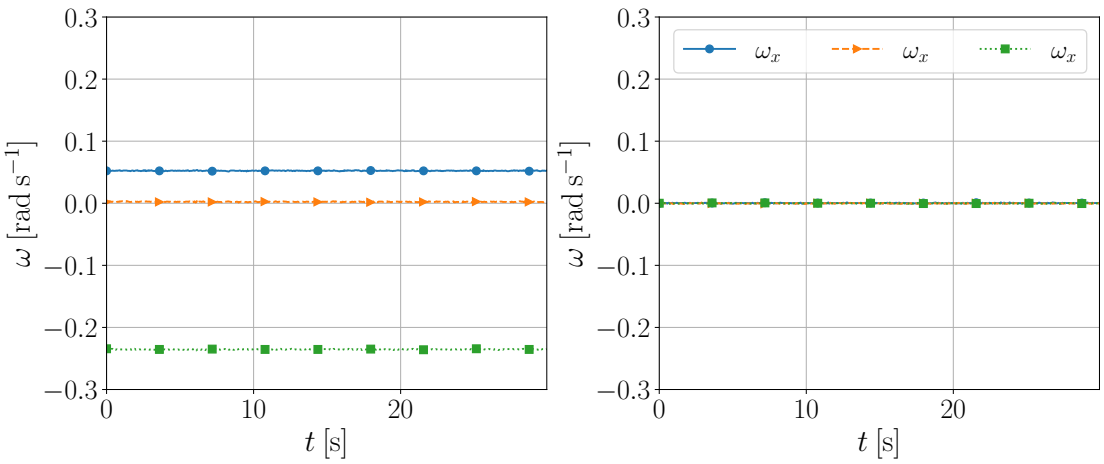

Figura 4.2: Mediciones estáticas del giroscopio: con bias (izquierda) y sin bias (derecha)

La tabla 4.1 representa los valores del bias y de la desviación típica de las lecturas del giróscopo en cada eje. La desviación típica de las medidas del giróscopo se utilizará más adelante para estimar la incertidumbre de la etapa de predicción del filtro de Kalman.

Eje	**Bias [rad/s]**	**STD [rad/s]**
ω_x	0.0523	4.199×10^{-4}
ω_y	-0.0023	4.204×10^{-4}
ω_z	0.2354	4.391×10^{-4}

Tabla 4.1: Caracterización del giróscopo.

4.2. Calibración del acelerómetro

El acelerómetro proporciona los valores de la aceleración en las tres direcciones del espacio. El MPU9250 utiliza la tecnología MEMS para medir el vector de aceleración. Debido a su construcción, en reposo, el acelerómetro mide una aceleración de módulo g según el sentido de la gravedad mientras que, en caída libre, mide una aceleración nula. En ausencia de aceleraciones externas, la medida del vector de gravedad permite tener una referencia de en qué dirección se encuentra el suelo. Las medidas del acelerómetro están sometidas a errores de sesgo, factores de escala y desalineación de los ejes.

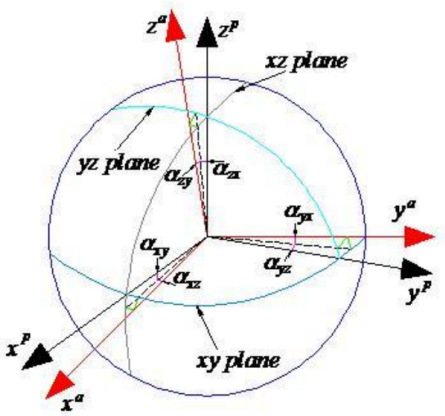

Figura 4.3: Errores de alineamiento entre los ejes del acelerómetro y el sistema de ejes cuerpo. Fuente: [37].

Idealmente, las direcciones en las que el sensor mide la aceleración deberían de ser ortogonales entre sí. Sin embargo, debido a las imprecisiones en la construcción de la IMU, a menudo existen desajustes en el alineamiento de los ejes. Esto se traduce en una falta de ortogonalidad entre los ejes del sensor que causa errores en las lecturas del acelerómetro. La figura 4.3 esquematiza los errores de desalineamineto entre los ejes del sensor $-a$ y el sistema de ejes cuerpo $-b$. En esta sección se presenta un método para corregir los errores de alineamiento mediante una calibración estática del acelerómetro. Los detalles del método pueden encontrarse en [37]. En principio, se requiere conocer un total de seis ángulos de desviación, $(\alpha_{xy}, \alpha_{xz}, \alpha_{yx}, \alpha_{yz}, \alpha_{zx}, \alpha_{zy})$, entre los ejes del sensor y los ejes cuerpo para corregir los errores de ortogonalidad. Sin embargo, en [38] se demuestra que el modelo de error puede simplificarse si se asume que el eje x_b coincide con el eje x_a. Las lecturas del sensor pueden modelarse de acuerdo al siguiente modelo de error,

$$\mathbf{a}_i = ST^{-1}\mathbf{g}_i + \mathbf{b}^a + \boldsymbol{\epsilon} \tag{4.1}$$

donde $\mathbf{a}_i$ es una lectura del acelerómetro, S es una matriz diagonal que contiene los factores de escala, T es la matriz de desalineación, $\mathbf{b}^a$ es el bias constante de cada uno de los

ejes del sensor y $\boldsymbol{\epsilon}$ es el ruido. El vector $\mathbf{g}_i$ es el vector de gravedad y el subíndice $-i$ denota el número de muestra. La matriz T permite transformar un vector expresado en ejes acelerómetro $-a$ a ejes cuerpo $-b$. Su expresión es,

$$T = \begin{bmatrix} 1 & -\alpha_{yz} & -\alpha_{zy} \\ 0 & 1 & -\alpha_{zx} \\ 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

A partir de la ecuación (4.1), un total de nueve parámetros recogidos en (4.3) deben ser determinados para corregir las lecturas del sensor.

$$\begin{bmatrix} k_x & k_y & k_z & \alpha_{yz} & \alpha_{zy} & \alpha_{zx} & b_x & b_y & b_z \end{bmatrix} \tag{4.3}$$

La ecuación (4.1) puede expandirse para obtener,

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}_i = \begin{bmatrix} b_x + k_x g_{xi} + \alpha_{yz} k_x g_{yi} - k_x g_{zi}(\alpha_{zy} - \alpha_{yz}\alpha_{zx}) \\ b_y + k_y g_{yi} + \alpha_{zx} k_y g_{zi} \\ b_z + k_z g_{zi} \end{bmatrix}_i \tag{4.4}$$

El examen detenido de la ecuación (4.4) revela que, por cada eje, existe un número k de parámetros de calibración a estimar. En la dirección z, las incógnitas son el sesgo, b_z, y el factor de escala k_z, resultando en un número de incógnitas $k = 2$. Se puede comprobar que el número de parámetros de calibración a estimar en las direcciones x e y es de 4 y 3 respectivamente. Por tanto, utilizando un número i de muestras igual al número de incógnitas en cada eje, la ecuación (4.4) puede descomponerse en tres sistemas lineales de ecuaciones,

$$\begin{bmatrix} a_{z1} \\ a_{z2} \end{bmatrix} = \begin{bmatrix} 1 & g_{z1} \\ 1 & g_{z2} \end{bmatrix} \begin{bmatrix} b_z \\ k_z \end{bmatrix} \tag{4.5}$$

$$\begin{bmatrix} a_{y1} \\ a_{y2} \\ a_{y3} \end{bmatrix} = \begin{bmatrix} 1 & g_{y1} & g_{z1} \\ 1 & g_{y2} & g_{z2} \\ 1 & g_{y3} & g_{z3} \end{bmatrix} \begin{bmatrix} b_y \\ k_y \\ k_{yzx} \end{bmatrix} \tag{4.6}$$

$$\begin{bmatrix} a_{x1} \\ a_{x2} \\ a_{x3} \\ a_{x4} \end{bmatrix} = \begin{bmatrix} 1 & g_{x1} & g_{y1} & -g_{z1} \\ 1 & g_{x2} & g_{y2} & -g_{z2} \\ 1 & g_{x3} & g_{y3} & -g_{z3} \\ 1 & g_{x4} & g_{y4} & -g_{z4} \end{bmatrix} \begin{bmatrix} b_x \\ k_x \\ k_{xyz} \\ k_{xzy} \end{bmatrix} \tag{4.7}$$

donde

$$\alpha_{zx} = \frac{k_{yzx}}{k_y} \tag{4.8}$$

$$\alpha_{yz} = \frac{k_{xyz}}{k_x} \tag{4.9}$$

$$\alpha_{zy} = \frac{k_{xzy}}{k_x} - \alpha_{yz}\alpha_{zx} \tag{4.10}$$

Para determinar los coeficientes de calibración del acelerómetro (4.3) es necesario realizar n mediciones del campo gravitatorio en n orientaciones diferentes para las que se conozca el valor teórico de la dirección del vector de gravedad $\mathbf{g}$. Siendo k el número de incógnitas por eje y n el número de orientaciones evaluadas, existen un total de,

$$\mathcal{C} = \frac{n!}{k!(n-k)!} \tag{4.11}$$

combinaciones de medidas por cada eje para calcular los coeficientes de calibración. Aumentando el número de direcciones evaluadas n es posible reducir el error en la estimación de (4.3). Teniendo en cuenta que la determinación de los parámetros de calibración en el eje x requiere evaluar al menos cuatro orientaciones, el número mínimo de medidas independientes que permite obtener una calibración completa es de cuatro. La tabla 4.2 muestra los parámetros de calibración obtenidos tras haber evaluado cuatro orientaciones del sensor. Nótese que los valores de los factores de escala son adimensionales y pueden utilizarse para reescalar el módulo total del vector gravedad.

k_x	k_y	k_z	α_{yz} [rad]	α_{zy} [rad]	α_{zx} [rad]	b_x [m s^{-2}]	b_y [m s^{-2}]	b_z [m s^{-2}]
0.9961	0.9908	0.9989	-0.01359	-0.0550	0.0013	0.8600	-1.5036	-1.8833

Tabla 4.2: Parámetros de calibración del acelerómetro

La figura 4.4 representa las lecturas del sensor en tres orientaciones distintas antes y después de aplicar la calibración. En las orientaciones evaluadas, el vector de la gravedad apunta según la dirección de uno de los ejes del acelerómetro. Por ello, las lecturas del acelerómetro en las otras dos direcciones deberían ser nulas. La calibración del sensor permite reducir de manera muy significativa los errores de bias y de ortogonalidad. Es posible comprobar que, tras la calibración, el módulo de la aceleración se mantiene casi constante e igual a g en todas las orientaciones evaluadas.

4.3. Calibración del magnetómetro.

El magnetómetro actúa como una brújula, proporcionando las tres componentes del vector del campo magnético terrestre medidas en ejes cuerpo $-b$.

$$\mathbf{m} = \begin{bmatrix} m_x, & m_y & m_z \end{bmatrix}^b \tag{4.12}$$

El vector en ejes cuerpo debe ser transformado a ejes inerciales para poderlo comparar con la dirección del campo magnético terrestre en esa posición, que es un vector de referencia conocido. La dirección del vector del campo magnético terrestre varía a lo largo de la superficie del planeta pero puede considerarse constante para pequeños desplazamientos.

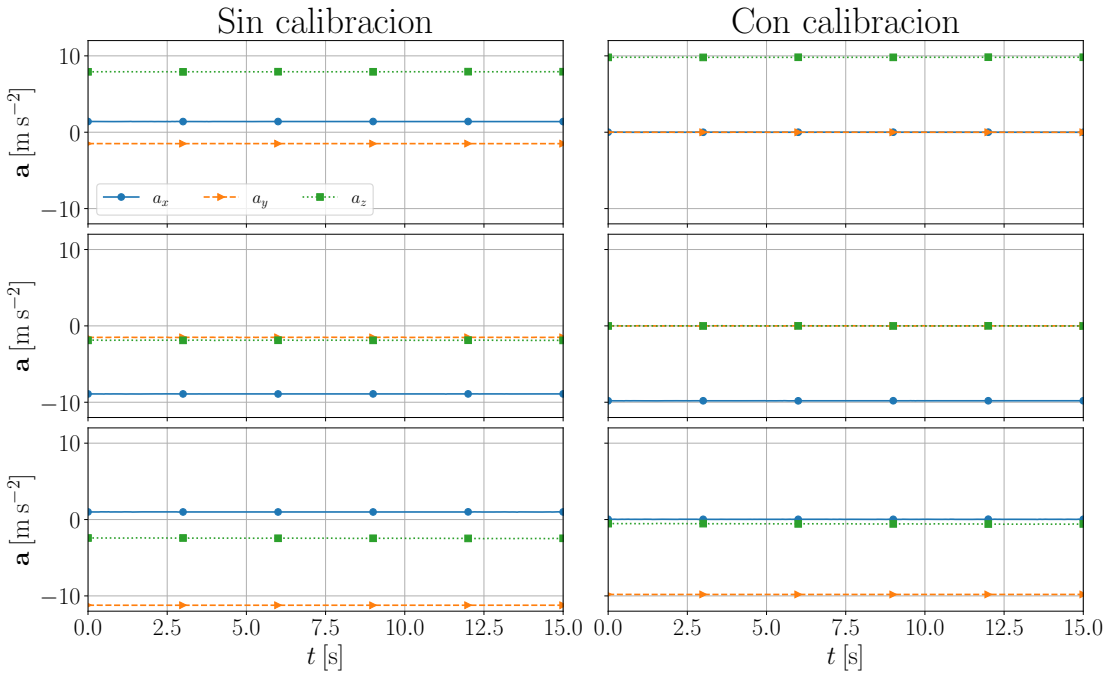

Figura 4.4: Lecturas estáticas del acelerómetro para tres orientaciones diferentes previas a la calibración (izquierda) y posteriores a la calibración (derecha).

A partir de las coordenadas de un punto de la superficie terrestre es posible obtener los valores de las componentes del vector del campo magnético utilizando herramientas tales como NCEI Geomagnetic Calculators. En la tabla 4.3 se recogen las componentes del campo magnético para dos localizaciones distintas junto con sus incertidumbres.

Localización	Latitud [N]	Longitud [W]	m_x^I [nT]	m_y^I [nT]	m_z^I [nT]	δm_x^I [nT]	δm_y^I [nT]	δm_z^I [nT]
Madrid	40° 25' 10"	3° 41' 31	25749.4	97.1	36931.8	131	94	157
Toulouse	43° 36' 16"	1° 26' 39"	23869.8	637.5	39984.3	131	94	157

Tabla 4.3: Componentes del vector del campo magnético terrestre en ejes inerciales (NED) para dos localizaciones de referencia.

La comparación entre las lecturas del magnetómetro y el valor de referencia del campo magnético puede utilizarse para estimar la actitud del sistema de referencia ejes cuerpo $-b$. En contraste con el acelerómetro, el vector de referencia del campo magnético tiene componentes no nulas en el plano $X_I Y_I$ por lo que permite estimar el ángulo de guiñada ψ. Otra de las ventajas del magnetómetro es que sus medidas no se ven afectadas por las aceleraciones externas, que sí afectan a las lecturas del acelerómetro.

A día de hoy existen varios tipos de sensores del campo magnético basados en diferentes tecnologías. Por una parte, podemos encontrar los sensores magnetorresistivos anisotrópicos

(AMR por sus siglas en inglés). En este tipo de sensores existen elementos resistivos cuya resistencia efectiva cambia cuando son sometidos a un campo magnético. Su sensibilidad al campo magnético depende de la dirección del campo magnético incidente con respecto a la orientación del elemento magnetorresistivo, de ahí la denominación de anisotrópicos. Para la medición del campo magnético es necesario disponer cuatro elementos magnetorresistivos en configuración de puente de Wheatstone y medir el voltaje de salida. Es posible encontrar más detalles sobre esta tecnología en [39, 40].

Por otra parte se encuentran los sensores magnéticos de efecto Hall [41]. Los sensores magnéticos de efecto Hall son dispositivos electrónicos que detectan la presencia de un campo magnético utilizando el efecto Hall, descubierto por el físico estadounidense Edwin Hall en 1879. El efecto Hall se refiere al fenómeno que consiste en la aparición de una diferencia de potencial en un conductor cuando se coloca en un campo magnético perpendicular y fluye una corriente a través de él. Los sensores de efecto Hall consisten en una delgada tira de material semiconductor (como arseniuro de galio o silicio) a través de la cuál se hace pasar una corriente eléctrica. Al aproximar el sensor a un campo magnético perpendicular a la dirección de la corriente, se generará un voltaje proporcional al producto de la fuerza del campo magnético y la corriente. Conociendo el valor de la corriente, es posible calcular la fuerza del campo magnético.

De entre las diferentes tecnologías de detección del campo magnético, los sensores de efecto Hall son los más extendidos. Esto es debido a que es posible producir sensores de gran calidad utilizando las técnicas convencionales de construcción de circuitos integrados. Esto permite integrar el sensor dentro de un chip de manera sencilla y barata. Los sensores de efecto Hall se utilizan en una amplia variedad de aplicaciones, incluyendo sistemas de control automotriz e industrial, robótica y electrónica de consumo. La IMU MPU9250 utiliza un sensor de tipo efecto Hall para medir las componentes del campo magnético.

No obstante, el magnetómetro sufre de una serie de limitaciones que hacen que su calibración sea extremadamente importante para poderlo utilizar de forma efectiva en la estimación de la actitud del vehículo. Una descripción exhaustiva sobre las fuentes de error en los sensores de campo magnético puede encontrarse en [39]. De forma resumida, este tipo de sensores están sometidos a errores en los factores de escala asociados a la falta de proporcionalidad entre la salida del sensor y el campo magnético detectado, errores de offset debido a la presencia de campos parásitos, errores debido a la sensibilidad cruzada entre ejes o desalineamiento y, finalmente, a errores debido al ruido. De entre las fuentes de error que causan errores de offset y de desalineamiento podemos destacar los errores de hierro. Los errores de hierro están causados por las perturbaciones magnéticas originadas por la presencia de materiales ferromagnéticos y sistemas electromagnéticos en las proximidades del sensor. Estos errores están causados por los diferentes elementos de la planta donde se instala el sensor y dependen por tanto de su configuración. Esto hace que la calibración del sensor sea diferente en función de si este se instala en un placa de prototipado o en la planta final. Los errores de hierro se clasifican a su vez en errores de hierro duro y

blando. Los errores de hierro duro están causados por fuentes de campo magnético cuyas componentes son constantes en todas la direcciones. En otras palabras, son perturbaciones del campo magnético generadas por los diferentes subsistemas electrónicos presentes en las proximidades del sensor y que no dependen del campo magnético terrestre. Por otra parte, los errores de hierro blando surgen de la interacción entre el campo magnético terrestre y los elementos ferromagnéticos presentes en las proximidades del sensor. Estas perturbaciones tienen una naturaleza más compleja ya que dependen de la dirección del campo magnético terrestre incidente y, por tanto, de la orientación del sensor.

A continuación se presenta el modelo de error empleado para calibrar el magnetómetro y se describe el procedimiento de calibración. Sea $\hat{\mathbf{m}}$ el vector del campo magnético medido por el sensor y $\mathbf{m}$ el vector del campo magnético real en ejes cuerpo, el modelo de error de las medidas del magnetómetro que permite relacionar ambos términos viene dado por la ecuación (4.13).

$$\hat{\mathbf{m}} = SM(A_{si}\mathbf{m} + \mathbf{b}_{hi}) + \mathbf{b}_m + \epsilon \tag{4.13}$$

donde,

- S es la matriz diagonal de factores de escala,

  $$S = \text{diag}(s_x, s_y, s_z) \tag{4.14}$$

  Los factores de escala permiten modelar los errores de proporcionalidad entre la salida del sensor y las componentes del campo magnético en las tres direcciones del espacio. Además, este término permite normalizar la salida del sensor.

- M es la matriz de desalineamiento, modela los errores de ortogonalidad entre los ejes del sensor y viene dada por

  $$M = N^{-1} = \begin{bmatrix}\mathbf{n}_x & \mathbf{n}_y & \mathbf{n}_z\end{bmatrix}^{-1} \tag{4.15}$$

  donde $\mathbf{n}_x$, $\mathbf{n}_y$ y $\mathbf{n}_z$ son vectores columna de dimensión tres que definen la orientación de los ejes del sensor $F(O_s, x_s, y_s, z_s)$ con respecto a los ejes cuerpo $F(O_b, x_b, y_b, z_b)$.

- $\mathbf{b}_m$ representa el sesgo (bias) en la salida del sensor modelado como un *offset* (desplazamiento)

  $$\mathbf{b}_m = \begin{bmatrix}b_{m,x} & b_{m,y} & b_{m,z}\end{bmatrix}^T \tag{4.16}$$

- $\mathbf{b}_{hi}$ modela los errores de desviación magnética de tipo hierro duro. La presencia de campos magnéticos parásitos causados por fuentes de campo magnético presentes en las proximidades del sensor se representa mediante un desplazamiento o sesgo.

  $$\mathbf{b}_{hi} = \begin{bmatrix}b_{hi,x} & b_{hi,y} & b_{hi,z}\end{bmatrix}^T \tag{4.17}$$

- A_{si} es la matriz que modela los errores de hierro suave causados por fuentes externas de campo magnético que cambian en función de la orientación del sensor. La matriz A_{si} viene dada por,

$$A_{si} = \begin{bmatrix} a_{11}^{si} & a_{12}^{si} & a_{13}^{si} \\ a_{21}^{si} & a_{22}^{si} & a_{23}^{si} \\ a_{31}^{si} & a_{32}^{si} & a_{33}^{si} \end{bmatrix} \tag{4.18}$$

La ecuación (4.13) puede reescribirse como,

$$\hat{\mathbf{m}} = A \cdot \mathbf{m} + \mathbf{b}^m + \epsilon \tag{4.19}$$

donde

$$A = SMA_{si} \quad \text{y} \quad \mathbf{b}^m = SM\mathbf{b}_{hi} + \mathbf{b}_m \tag{4.20}$$

La matriz A combina los errores de factor de escala, desalineamiento y perturbaciones de hierro suave mientras que el vector $\mathbf{b}^m$ agrupa los errores de sesgo (bias).

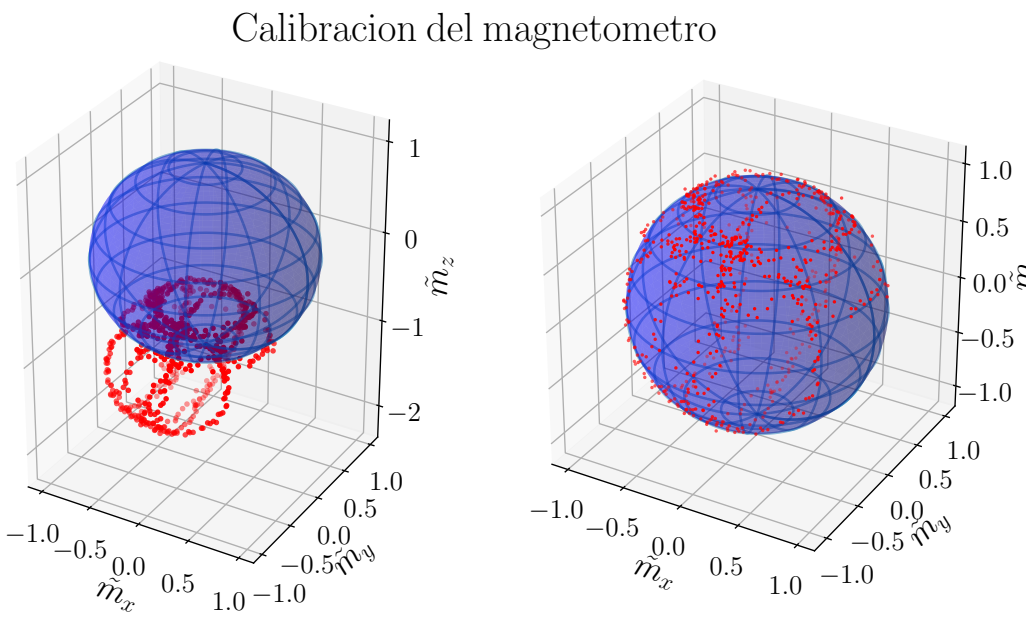

Figura 4.5: Lugar geométrico de los puntos del espacio descrito por el vector del campo magnético medido por el sensor cuando le se somete a rotaciones aleatorias en las tres direcciones del espacio. A la izquierda se representan las lecturas brutas previas a la calibración y una esfera de radio unitario centrada en el origen. A la derecha se representan esas mismas lecturas tras aplicar la calibración descrita. Las lecturas del magnetómetro se han normalizado con el módulo del vector del campo magnético en Toulouse (ver tabla 4.3).

A partir del modelo de error descrito podemos manipular la ecuación (4.20) para aislar el campo magnético $\mathbf{m}$ real en ejes cuerpo.

$$\mathbf{m} = A^{-1}(\hat{\mathbf{m}} - \mathbf{b}^m) \tag{4.21}$$

done $\mathbf{m}$ es el vector del campo magnético real en ejes cuerpo y $\hat{\mathbf{m}}$ es el vector del campo magnético medido por el sensor y sometido a las diferentes fuentes de error descritas. La calibración consiste en estimar los términos de la matriz A y del vector $\mathbf{b}^m$ para poder corregir las lecturas del sensor empleando la ecuación (4.21). De ahora en adelante, las lecturas del sensor se asumen normalizadas por el módulo del campo magnético de manera que el vector del campo magnético $\mathbf{m}$ satisface,

$$\mathbf{m}^T\mathbf{m} = 1 \tag{4.22}$$

Substituyendo la ecuación (4.21) en (4.22), se obtiene.

$$(\hat{\mathbf{m}} - \mathbf{b}^m)^T(A^{-1})^T A^{-1}(\hat{\mathbf{m}} - \mathbf{b}^m) = 1 \tag{4.23}$$

Se denota por Q el producto $Q = (A^{-1})^T A^{-1}$. Es fácil comprobar que Q es una matriz simétrica ya que es el producto de una matriz multiplicada por su transpuesta. Sustituyendo se obtiene,

$$(\hat{\mathbf{m}} - \mathbf{b}^m)^T Q(\hat{\mathbf{m}} - \mathbf{b}^m) = 1 \tag{4.24}$$

Expandiendo la ecuación anterior y simplificando se llega a,

$$\hat{\mathbf{m}}^T Q\hat{\mathbf{m}} - \hat{\mathbf{m}}^T Q\mathbf{b}^m - \mathbf{b}^{m^T} Q\hat{\mathbf{m}} + \mathbf{b}^{m^T} Q\mathbf{b}^m - 1 = 0 \tag{4.25}$$

dado que Q es una matriz simétrica, $\mathbf{b}^{m^T} Q\hat{\mathbf{m}} = \hat{\mathbf{m}}^T Q\mathbf{b}^m$. Por tanto, podemos reescribir la ecuación (4.25) como

$$\hat{\mathbf{m}}^T Q\hat{\mathbf{m}} - 2\hat{\mathbf{m}}^T Q\mathbf{b}^m + \mathbf{b}^{m^T} Q\mathbf{b}^m - 1 = 0 \tag{4.26}$$

La ecuación (4.26) puede finalmente escribirse como la ecuación de una forma cuadŕática

$$\hat{\mathbf{m}}^T Q\hat{\mathbf{m}} + \hat{\mathbf{m}}^T n + d = 0 \tag{4.27}$$

donde

$$Q = (A^{-1})^T A^{-1} \tag{4.28}$$
$$n = -2Q\mathbf{b}^m \tag{4.29}$$
$$d = \mathbf{b}^{m^T} Q\mathbf{b}^m - 1 \tag{4.30}$$

El método de calibración del magnetómetro consiste en una calibración tridimensional que emplea el ajuste por mínimos cuadrados y se presenta con detalle en [42, 39, 43]. En ausencia de perturbaciones y errores de desalineamineto, el módulo del vector del campo magnético medido por el sensor debe ser igual al módulo campo magnético terrestre independientemente de la orientación del sensor. En consecuencia, cuando el sensor se gira en los tres ejes del espacio, el vector del campo magnético medido por el sensor en ejes cuerpo debe describir una esfera de radio igual al módulo del vector del campo magnético.

Debido a las diferentes fuentes de error descritas anteriormente, la traza del vector del campo magnético obtenida al girar el sensor formará un elipsoide.

La calibración consiste en encontrar los parámetros Q, n y d que satisfacen la ecuación (4.27). Se trata de un problema de ajuste de elipsoides utilizando las lecturas del sensor recopiladas para diferentes orientaciones. De entre los diferentes métodos de ajuste de elipsoides existentes es posible citar los algoritmos algebraicos [44, 45], geométricos [46] y adaptativos [47]. En la presente implementación se ha utilizado la metodología descrita en [44].

La calibración requiere medir el campo magnético en un gran número de orientaciones espaciales diferentes. A diferencia del acelerómetro, que requiere una calibración estática, el magnetómetro puede ser calibrado de forma dinámica, registrando los valores de las lecturas del magnetómetro durante un periodo de tiempo y girando el sensor en las tres direcciones del espacio. No obstante, debido a los ya mencionados errores de hierro duros y suaves, para que la calibración sea efectiva, esta debe realizarse directamente en la planta final. Esta operación puede resultar complicada en sistemas reales. Otros métodos de calibración 2D como el presentado por [48] permiten simplificar la fase de adquisición de datos para la calibración.

Tras ajustar los valores de Q, n y d utilizando los datos experimentales es posible calcular la matriz A^{-1} y el vector de bias $\mathbf{b}^m$ aplicando,

$$\mathbf{b}^m = -Q^{-1}n \tag{4.31}$$

$$A^{-1} = \frac{Q^{1/2}}{\sqrt{n^T Q^{-1} n - d}} \tag{4.32}$$

La calibración representada en la figura 4.5 arroja los siguientes valores para la matriz A^{-1} y el vector de sesgo $\mathbf{b}^m$,

$$A^{-1} = \begin{bmatrix} 1.3334 & 0.0230 & -0.0790 \\ 0.0230 & 1.3369 & -0.0145 \\ -0.0790 & -0.0145 & 1.3272 \end{bmatrix} \qquad \mathbf{b}^m = \begin{bmatrix} -0.2733 \\ 0.2027 \\ -1.3613 \end{bmatrix} \tag{4.33}$$

Nótese que los valores del sesgo $\mathbf{b}^m$ son adimensionales ya que se ha normalizado cada lectura del magnetómetro con el módulo del vector del campo magnético $\mathbf{m}$.

4.4. Header de la clase `IMU.h`

```
#ifndef IMU_H_
#define IMU_H_
// Include dependencies
#include <stdio.h>
#include "pico/stdlib.h"
#include "hardware/i2c.h"
#include <vector>
//-----------------------------------------------------------------------------
// IMU Library
//-----------------------------------------------------------------------------
// Author:  Enrique Flores
// Date:    15/07/2023
// Define I2C communication port
#define I2C_PORT i2c0
// Define I2C port pins: these correspond to the physical GPIO in the
// Raspberry pi Pico device
#define I2C_SDA 4
#define I2C_SCL 5
//-------------------------------------------------
//------------------- MPU9250 ---------------------
//-------------------------------------------------
//---------------- CONFIGURATION ------------------
// MPU9259 is a 9DoFs Inertial Measurement Unit (IMU)
// MPU9250 = Accelerometer + Gyroscope + Magnetometer
// Memory address
#define    MPU9250_ADDRESS            0x68
// The magnetometer has a different address
#define    MAG_ADDRESS                0x0C
// Gyroscope range options
#define    GYRO_FULL_SCALE_250_DPS    0x00
#define    GYRO_FULL_SCALE_500_DPS    0x08
#define    GYRO_FULL_SCALE_1000_DPS   0x10
#define    GYRO_FULL_SCALE_2000_DPS   0x18
// Accelerometer range options
#define    ACC_FULL_SCALE_2_G        0x00
#define    ACC_FULL_SCALE_4_G        0x08
#define    ACC_FULL_SCALE_8_G        0x10
#define    ACC_FULL_SCALE_16_G       0x18

//#################################################
// ------------------- IMU CLASS -------------------
//#################################################
class IMU {
  private:
    //#################################################
```

```
    // Magnetometer calibration
    //#####################################################
    // Module of the reference magnetic field vector in mT
    double mag_mod = 46.5671;
    // Calibration matrix eq 4.20
    std::vector<std::vector<double>>
                    mag_A_inv = {{ 1.33336896, 0.02303043,-0.07898045},
                                 { 0.02303043, 1.33692563,-0.01454740},
                                 {-0.07898045,-0.01454740, 1.32716053}};
    // Magnetometer bias vector eq 4.20
    std::vector<double> mag_bias = {-0.27334742, 0.20273002, -1.36129358};

    //#####################################################
    // Accelerometer calibration
    //#####################################################
    // Acelerometer scale factors
    double kx = 0.99612761, ky = 0.99079333, kz = 0.99889998;
    // Acelerometer misalignment angles
    double ayz = -0.01358966, azy = -0.05504814, azx = 0.00130575;
    // Misalignment matrix
    std::vector<std::vector<double>> acc_T = {{1, -ayz, azy},{0, 1, -azx},{0, 0, 1}};
    // Scale factor matrix
    std::vector<std::vector<double>> acc_S = {{kx, 0, 0},{0, ky, 0},{0, 0, kz}};
    // Accelerometer bias
    std::vector<double> acc_bias = {0.86002407, -1.50361382, -1.8833217 };

    //#####################################################
    // Gyroscope calibration
    //#####################################################
    // Gyroscope bias
    std::vector<double> gyro_bias = {0.05236351, -0.00233378, 0.23537489};

  public:
    // Initialize the IMU: set COM frequency, I2C pins, define sensors ranges
    void initialize();
    // Access to the MPU9250 memory registers
    void readsensor(int16_t imusensor[3][3]);
    // Apply sensor ranges to the int16_t raw measurements
    void applyrange(int16_t imusensor[3][3],
                    std::vector<double> &gyro,
                    std::vector<double> &mag,
                    std::vector<double> &acc);
    // Apply calibration
    void applycalibration(std::vector<double> &gyro,
                          std::vector<double> &acc,
                          std::vector<double> &mag);
};
#endif
```

Capítulo 5

Filtro de Kalman Extendido

5.1. Ecuaciones del filtro de Kalman

5.1.1. Variables aleatorias

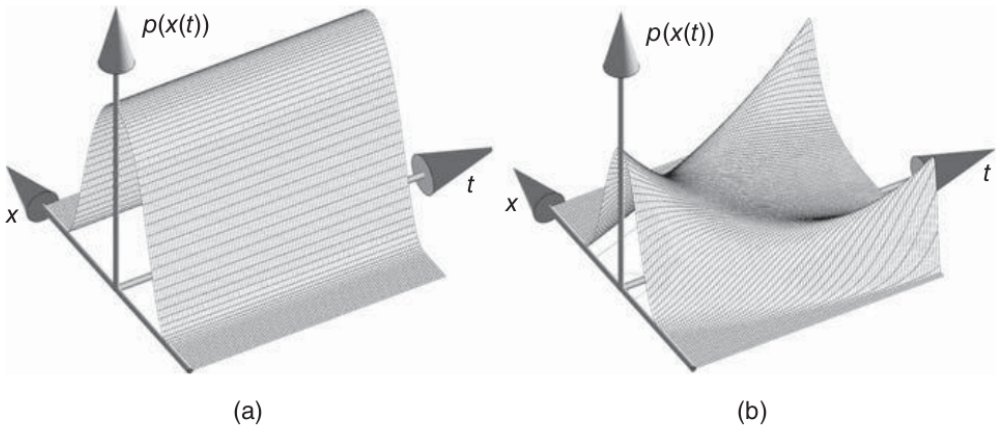

Figura 5.1: a) Variable aleatoria idénticamente distribuida: la distribución de probabilidad no cambia con el tiempo. b) Variable aleatoria no idénticamente distribuida: la distribución de probabilidad evoluciona con el tiempo. Fuente [12].

El filtro de Kalman busca estimar el estado de un sistema estocástico [49, 50]. En estos sistemas, el estado está formado por un conjunto de variables aleatorias caracterizadas por su distribución de probabilidad. En cada instante k, la distribución de probabilidad de la variable aleatoria tiene un valor esperado y una varianza (ver figura 5.1). Sea X una

variable aleatoria, su esperanza en el instante k se denota como,

$$E(X_k) = \mu_{X_k} = \int_{-\infty}^{+\infty} X(t_k) p\left(X(t_k)\right) dX(t_k) \tag{5.1}$$

donde μ_{X_k} y $p(X(t_k))$ son la media de la variable aleatoria y su distribución de probabilidad en el instante k respectivamente. Así mismo, la varianza de la variable aleatoria X en el instante k se denota como,

$$V(X_k) = \sigma_{X_k}^2 = E\left[(X_k - \mu_{X_k})^2\right] = E(X_k^2) - \mu_{X_k}^2 \tag{5.2}$$

La covarianza entre dos variables aleatorias X e Y en el instante k viene dada por,

$$COV(X_k, Y_k) = E\left[(X_k - \mu_{X_k})(Y_k - \mu_{Y_k})\right] = E(X_k Y_k) - \mu_{X_k}\mu_{Y_k} \tag{5.3}$$

En el caso de variables aleatorias multidimensionales $\mathbf{X}_k \in \mathbb{R}^n$, la matriz de covarianza viene dada por,

$$P = E\left[(\mathbf{X}_k - \mu_{\mathbf{X}_k})(\mathbf{X}_k - \mu_{\mathbf{X}_k})^T\right] = \begin{bmatrix} \sigma_{X_1}^2 & \sigma_{X_1}\sigma_{X_2} & \dots & \sigma_{X_1}\sigma_{X_n} \\ \sigma_{X_2}\sigma_{X_1} & \sigma_{X_2}^2 & \dots & \\ \vdots & & \ddots & \\ \sigma_{X_n}\sigma_{X_1} & \dots & & \sigma_{X_n}^2 \end{bmatrix}_k \tag{5.4}$$

5.1.2. Dinámica de sistemas lineales discretos

Las ecuaciones algebraico-diferenciales que describen el comportamiento de un sistema dinámico no lineal son,

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{w}) \quad \mathbf{x} \in \mathbb{R}^n,\ \mathbf{u} \in \mathbb{R}^p,\ \mathbf{w} \in \mathbb{R}^n \tag{5.5}$$

$$\mathbf{y} = \mathbf{h}(\mathbf{x}, \mathbf{u}, \mathbf{v}) \quad \mathbf{y} \in \mathbb{R}^m,\ \mathbf{v} \in \mathbb{R}^m \tag{5.6}$$

donde $\mathbf{x}$ es el vector de estados, $\mathbf{y}$ es la salida y $\mathbf{u}$ es la entrada del sistema. Los términos $\mathbf{w}$ y $\mathbf{v}$ denotan el ruido de la planta y de las sensores. Linealizando las ecuaciones (5.5) y (5.6) en torno a un estado, una entrada, y una salida de referencia, $(\overline{\mathbf{x}}, \overline{\mathbf{y}}, \overline{\mathbf{u}})$, se obtiene,

$$\dot{\mathbf{x}}' = A\mathbf{x}' + B\mathbf{u}' + \mathbf{w} \tag{5.7}$$

$$\mathbf{y}' = C\mathbf{x}' + D\mathbf{u}' + \mathbf{v} \tag{5.8}$$

donde $z = \overline{z} + z'$, siendo z una variable genérica, $A_{ij} = \partial f_i/\partial x_j$, $B_{ik} = \partial f_i/\partial u_k$, $C_{li} = \partial h_l/\partial x_i$ y $D_{lk} = \partial h_l/\partial u_k$. En lo que sigue, se consideran sistemas dinámicos en los que las entradas no tienen una influencia directa en la salida, luego la matriz de transición

directa, $D_{lk} = 0$, es nula. Asumiento de la entrada del sistema, $\mathbf{u}'$ es constante a trozos, la integración temporal de la ecuación de estado linealizada (5.7) proporciona,

$$\mathbf{x}'(t + \Delta t) = \mathbf{x}'(t)e^{A\Delta t} + \int_0^{\Delta t} e^{At} dt B\mathbf{u}(t) \tag{5.9}$$

donde,

$$F = e^{A\Delta t} = I + A\Delta t + \frac{(A\Delta t)^2}{2!} + \frac{(A\Delta t)^3}{3!} + \dots \tag{5.10}$$

$$G = \int_0^{\Delta t} e^{At} dt B = \Delta t \left(I + A\Delta t + \frac{(A\Delta t)^2}{2!} + \frac{(A\Delta t)^3}{3!} + \dots \right) B \tag{5.11}$$

De esta manera, en tiempo discreto, las ecuaciones en diferencias que describen la dinámica del sistema lineal toman la forma,

$$\mathbf{x}_{k+1} = F\mathbf{x}_k + G\mathbf{u}_k + \mathbf{w}_k \tag{5.12}$$

$$\mathbf{y}_k = H\mathbf{x}_k + \mathbf{v}_k \tag{5.13}$$

donde $z_k = z'(t_k)$ representa el valor de la variable z' en el instante de tiempo $t_k = k\Delta T$ con $k \in \mathbb{N}$ y ΔT igual al periodo de muestreo. La matriz de observación en tiempo discreto se denota por H.

5.1.3. Filtro de Kalman en sistemas lineales

El filtro de Kalman es un estimador óptimo consta de dos etapas denominadas predicción y actualización. A continuación se describen las ecuaciones del filtro de Kalman en sistemas dinámicos lineales e invariantes en el tiempo. De ahora en adelante, se denota por $\hat{z}_k$ a la estimación de la variable genérica z en el instante k. Esta notación se introduce para distinguir entre la estimación del estado obtenida mediante el filtro del valor real de la variable z. Así mismo, se diferencia entre la estimación a priori $\hat{z}^-$ (denotada también como $\hat{z}_{k|k-1}$ en otros textos) y la estimación a posteriori, $\hat{z}_k$ (denotada como $\hat{z}_{k|k}$ en la literatura relacionada).

Predicción

Ecuación de transición de estado

La ecuación de transición de estado emplea el modelo de la dinámica del sistema para obtener una estimación a priori del estado en el instante $k+1$, $\hat{\mathbf{x}}_{k+1}^-$, a partir de la estimación del estado en el instante k, $\hat{\mathbf{x}}_k$,

$$\hat{\mathbf{x}}_{k+1}^{-} = F\hat{\mathbf{x}}_k + G\mathbf{u}_k \tag{5.14}$$

En esta ecuación intervienen la matriz de transición en tiempo discreto $F \in \mathbb{R}^n \times \mathbb{R}^n$, la matriz de control $G \in \mathbb{R}^n \times \mathbb{R}^p$ y el vector de entradas del sistema en el instante k, $\mathbf{u}_k \in \mathbb{R}^p$.

Ecuación de extrapolación de la varianza

La ecuación de extrapolación de la varianza busca predecir el valor de la matriz de covarianza del estado en el instante $k+1$, $P \in \mathbb{R}^n \times \mathbb{R}^n$ a partir del valor de la matriz de covarianza en la iteración precedente P_k. La ecuación proporciona una estimación a priori de la matriz de covarianza P_{k+1}^{-}.

$$P_{k+1}^{-} = FP_kF^T + Q \tag{5.15}$$

Para obtener esta ecuación, consideramos la definición de matriz de covarianza de la estimación del estado $\hat{\mathbf{x}}_{k+1}^{-}$ que se define como la esperanza del cuadrado de la desviación de la estimación con respecto al valor real del estado $\mathbf{x}_{k+1}$. Matemáticamente, la definición se expresa como,

$$P_{k+1}^{-} = E\left[(\hat{\mathbf{x}}_{k+1} - \mathbf{x}_{k+1})(\hat{\mathbf{x}}_{k+1} - \mathbf{x}_{k+1})^T\right] \tag{5.16}$$

Introduciendo en la ecuación anterior las ecuaciones de transición de estado en tiempo discreto (5.12) y la ecuación de extrapolación del estado del filtro de Kalman (5.14) se obtiene,

$$\begin{aligned}
P_{k+1}^{-} &= E\left[(F\hat{\mathbf{x}}_k + G\mathbf{u}_k - F\mathbf{x}_k - G\mathbf{u}_k - \mathbf{w}_k)(F\hat{\mathbf{x}}_k + G\mathbf{u}_k - F\mathbf{x}_k - G\mathbf{u}_k - \mathbf{w}_k)^T\right] \\
&= E\left[(F(\hat{\mathbf{x}}_k - \mathbf{x}_k) - \mathbf{w}_k)(F(\hat{\mathbf{x}}_k - \mathbf{x}_k) - \mathbf{w}_k)^T\right] \\
&= E\left[(F(\hat{\mathbf{x}}_k - \mathbf{x}_k) - \mathbf{w}_k)((\hat{\mathbf{x}}_k - \mathbf{x}_k)^T F^T - \mathbf{w}_k^T)\right] \\
&= E\left[F(\hat{\mathbf{x}}_k - \mathbf{x}_k)(\hat{\mathbf{x}}_k - \mathbf{x}_k)^T F^T - 2F(\hat{\mathbf{x}}_k - \mathbf{x}_k)\mathbf{w}_k^T + \mathbf{w}_k\mathbf{w}_k^T\right] \\
&= FE\left[(\hat{\mathbf{x}}_k - \mathbf{x}_k)(\hat{\mathbf{x}}_k - \mathbf{x}_k)^T\right] F^T - 2FE\left[(\hat{\mathbf{x}}_k - \mathbf{x}_k)\mathbf{w}_k^T\right] + E\left[\mathbf{w}_k\mathbf{w}_k^T\right] \\
P_{k+1}^{-} &= FP_kF^T + Q
\end{aligned} \tag{5.17}$$

donde se ha tenido en cuenta que,

$$P_k = E\left[(\hat{\mathbf{x}}_k - \mathbf{x}_k)(\hat{\mathbf{x}}_k - \mathbf{x}_k)^T\right] \tag{5.18}$$

$$Q = E\left[\mathbf{w}_k\mathbf{w}_k^T\right] \tag{5.19}$$

y que el término $2FE\left[(\hat{\mathbf{x}}_k - \mathbf{x}_k)\mathbf{w}_k^T\right] = 0$ ya que el ruido $\mathbf{w}_k$ y el error $(\hat{\mathbf{x}}_k - \mathbf{x}_k)$ no están correlacionados y por tanto su esperanza es nula. La matriz $Q = E\left[\mathbf{w}_k\mathbf{w}_k^T\right]$ es la matriz de ruido del proceso. Cuando el ruido de las diferentes variables de estado no esta

correlacionado, la matriz de ruido del proceso Q es una matriz diagonal.

$$Q = \begin{bmatrix} q_{11} & 0 & \dots & 0 \\ 0 & q_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & q_{nn} \end{bmatrix} \tag{5.20}$$

Actualización

En la fase de actualización se utilizan las medidas de la salida del sistema, $\mathbf{y}_k$ para corregir las estimaciones del estado $\hat{\mathbf{x}}$ y de la matriz de covarianza P.

Ecuación de ganancia de Kalman

La primera de las ecuaciones de la fase de actualización es la ecuación de la ganancia de Kalman. En esta ecuación, se actualiza el valor de la matriz de ganancia de Kalman teniendo en cuenta la incertidumbre de las mediciones y la incertidumbre del proceso.

$$K_k = \left(P_k^- H^T\right)\left(HP_k^- H^T + R\right)^{-1} \tag{5.21}$$

En la ecuación (5.21), $K_k \in \mathbb{R}^n \times \mathbb{R}^m$ es la matriz de ganancia de Kalman en el paso de tiempo k, $R \in \mathbb{R}^m \times \mathbb{R}^m$ es la matriz de ruido de las mediciones, $R = E\left[\mathbf{v}_k \mathbf{v}_k^T\right]$, $H \in \mathbb{R}^m \times \mathbb{R}^n$ es la matriz de observación del proceso y P_k^- es la estimación a priori de la matriz de covarianza calculada en la fase de predicción del filtro. El filtro de Kalman busca minimizar el error de la estimación teniendo en cuenta el ruido del proceso, Q y de las medidas de los sensores R. Imponiendo que la estimación del estado en el instante k es una combinación lineal de la estimación a priori del estado $\hat{\mathbf{x}}_k^-$ y de la salida del sistema $\mathbf{y}_k$,

$$\hat{\mathbf{x}}_k = \mathcal{K}_x \hat{\mathbf{x}}_k^- + \mathcal{K}_y \mathbf{y}_k \tag{5.22}$$

se puede demostrar que el error en la estimación del estado se minimiza para,

$$\mathcal{K}_x = I + K_k H \tag{5.23}$$

$$\mathcal{K}_y = K_k = P_k^- H^T \left(HP_k^- H^T + R\right)^{-1} \tag{5.24}$$

Para una demostración completa el lector es referido a [12, 51]. Introduciendo las expresiones $\mathcal{K}_x$ y $\mathcal{K}_y$ en la ecuación (5.22) se obtiene la ecuación de actualización del estado.

Ecuación de actualización del estado

La segunda ecuación de la fase de actualización es la ecuación de actualización del estado. En esta ecuación, se corrige la estimación a priori del estado obtenida en la etapa

de predicción del filtro, $\hat{\mathbf{x}}_k^-$. La corrección del estado esta ponderada por la ganancia de Kalman K_k y es proporcional a la diferencia entre los valores de la salida observados $\mathbf{y}_k$ y los valores de la salida estimados a partir de la ecuación de observación del sistema $\hat{\mathbf{y}}_k^- = H\hat{\mathbf{x}}_k^-$. El filtro de Kalman es un filtro óptimo. Esto implica que la ganancia de Kalman en cada iteración se actualiza para minimizar el error en la estimación del estado $\hat{\mathbf{x}}_k$ teniendo en cuenta la incertidumbre de las mediciones R y la incertidumbre del proceso Q. La ecuación de actualización del estado se lee,

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + K_k(\mathbf{y}_k - H\hat{\mathbf{x}}_k^-) \tag{5.25}$$

donde $H \in \mathbb{R}^m \times \mathbb{R}^n$ es la matriz de observación, $\hat{\mathbf{x}}_k$ es la estimación del estado $\mathbf{x}_k$ en el instante k (también denotada como $\hat{\mathbf{x}}_{k|k}$) e $\mathbf{y}_k$ es el vector de variables observadas.

Ecuación de actualización de la varianza

La última ecuación de la fase de actualización del filtro es la ecuación de actualización de la covarianza cuya expresión es,

$$P_k = (I - K_k H)P_k^- \tag{5.26}$$

La ecuación de actualización de la covarianza puede derivarse teniendo en cuenta cual es la definición de la matriz de covarianza de la estimación del estado,

$$P_k = E\left[(\mathbf{x}_k - \hat{\mathbf{x}}_k)(\mathbf{x}_k - \hat{\mathbf{x}}_k)^T\right] \tag{5.27}$$

donde $\mathbf{x}_k$ representa el valor verdadero del estado en el instante k. Considerando la ecuación de actualización del estado (5.25), podemos escribir,

$$\begin{aligned} \mathbf{x}_k - \hat{\mathbf{x}}_k &= \mathbf{x}_k - \hat{\mathbf{x}}_k^- + K_k(H\mathbf{x}_k + \mathbf{v}_k - H\hat{\mathbf{x}}_k^-) \\ &= (I - K_k H)(\mathbf{x}_k - \hat{\mathbf{x}}_k^-) - K_k\mathbf{v}_k \end{aligned} \tag{5.28}$$

donde se ha tenido en cuenta que $\mathbf{y}_k = H\mathbf{x}_k + \mathbf{v}_k$. Introduciendo la ecuación (5.28) en la definición de la matriz de covarianza (5.27) y desarrollando se obtiene,

$$\begin{aligned} P_k &= E\left[((I - K_k H)(\mathbf{x}_k - \hat{\mathbf{x}}_k^-) - K_k\mathbf{v}_k)((I - K_k H)(\mathbf{x}_k - \hat{\mathbf{x}}_k^-) - K_k\mathbf{v}_k)^T\right] \\ &= E\left[((I - K_k H)(\mathbf{x}_k - \hat{\mathbf{x}}_k^-) - K_k\mathbf{v}_k)((\mathbf{x}_k - \hat{\mathbf{x}}_k^-)^T(I - K_k H)^T - \mathbf{v}_k^T K_k^T)\right] \\ &= E\left[(I - K_k H)(\mathbf{x}_k - \hat{\mathbf{x}}_k^-)(\mathbf{x}_k - \hat{\mathbf{x}}_k^-)^T(I - K_k H)^T\right] + \\ &+ E\left[K_k\mathbf{v}_k\mathbf{v}_k^T K_k^T\right] - 2E\left[(I - K_k H)(\mathbf{x}_k - \hat{\mathbf{x}}_k^-)\mathbf{v}_k^T K_k^T\right] \\ P_k &= (I - K_k H)P_k^-(I - K_k H)^T + K_k R K_k^T \end{aligned} \tag{5.29}$$

donde se ha considerado que,

$$P_k^- = E\left[(\mathbf{x}_k - \hat{\mathbf{x}}_k^-)(\mathbf{x}_k - \hat{\mathbf{x}}_k^-)^T\right] \tag{5.30}$$

$$R = E\left[\mathbf{v}_k\mathbf{v}_k^T\right] \tag{5.31}$$

$$0 = 2E\left[(\mathbf{x}_k - \hat{\mathbf{x}}_k^-)\mathbf{v}_k^T\right] \tag{5.32}$$

Finalmente, substituyendo la ecuación de la ganancia de Kalman (5.21) en la expresión (5.29) se obtiene,

$$P_k = (I - K_k H)P_k^- \tag{5.33}$$

Resumen de las ecuaciones

A continuación, se presenta un breve resumen de las ecuaciones del filtro de Kalman para sistemas lineales,

Predicción

$$\hat{\mathbf{x}}_{k+1}^- = F\hat{\mathbf{x}}_k + G\mathbf{u}_k \tag{5.34}$$
$$P_{k+1}^- = FP_kF^T + Q \tag{5.35}$$

Actualización

$$K_k = \left(P_k^- H^T\right)\left(HP_k^- H^T + R\right)^{-1} \tag{5.36}$$
$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + K_k(\mathbf{y}_k - H\hat{\mathbf{x}}_k^-) \tag{5.37}$$
$$P_k = (I - K_k H)P_k^- \tag{5.38}$$

5.1.4. Filtro de Kalman extendido

El filtro de Kalman extendido constituye una ampliación del filtro de Kalman a sistemas no lineales en los que las ecuaciones del sistema vienen dadas por,

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{w}_k \tag{5.39}$$
$$\mathbf{y}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{v}_k \tag{5.40}$$

En este caso, las ecuaciones del filtro son,

Predicción

$$\hat{\mathbf{x}}_{k+1}^{-} = \mathbf{f}(\hat{\mathbf{x}}_k, \mathbf{u}_k) \tag{5.41}$$
$$P_{k+1}^{-} = F_k P_k F_k^T + Q \tag{5.42}$$

Actualización

$$K_k = \left(P_k^{-} H_k^T\right)\left(H_k P_k^{-} H_k^T + R\right)^{-1} \tag{5.43}$$
$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^{-} + K_k(\mathbf{y}_k - \mathbf{h}(\hat{\mathbf{x}}_k^{-})) \tag{5.44}$$
$$P_k = (I - K_k H_k) P_k^{-} \tag{5.45}$$

donde se han remplazado la ecuación de estado y la ecuación de observación lineales por sus versiones no lineales. Nótese que las matrices de transición de estado F_k y de obervación H_k ya no son invariantes en el tiempo y deben calcularse en cada iteración k como,

$$F_k = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_k, \mathbf{u}_k} \tag{5.46}$$

$$H_k = \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_k^{-}} \tag{5.47}$$

5.2. Estimación de la actitud

En esta sección se describe la aplicación del filtro de Kalman extendido para la estimación de la actitud de un sistema a partir de las medidas del giróscopo, acelerómetro y magnetómetro.

5.2.1. Ecuación de extrapolación del estado

Para aplicar el filtro de Kalman descrito en la sección 5.1 a la estimación de la actitud, primero debemos definir cuál es el vector de estados del filtro. El vector de estados viene dado por,

$$\mathbf{x} = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 & b_x^g & b_y^g & b_z^g \end{bmatrix}^T \tag{5.48}$$

donde $\mathbf{q} = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix}^T$ es el cuaternio que describe la orientación del sistema ejes cuerpo con respecto al sistema de referencia inercial y $\mathbf{b}^g = \begin{bmatrix} b_x^g & b_y^g & b_z^g \end{bmatrix}^T$ son los valores

del sesgo del giróscopo en cada una de las direcciones del espacio. De forma más compacta, el estado del filtro puede escribirse como,

$$\mathbf{x} = \begin{bmatrix} \mathbf{q} \\ \mathbf{b}^g \end{bmatrix} \tag{5.49}$$

La ecuación que describe la dinámica del sistema eq. (2.25) fue presentada en el Capítulo 2 y se reproduce a continuación para facilitar la comprensión del desarrollo,

$$\dot{\mathbf{q}} = \frac{1}{2} S(\boldsymbol{\omega})\mathbf{q} = \frac{1}{2}\tilde{S}(\mathbf{q})\boldsymbol{\omega} \tag{5.50}$$

donde las matrices $S(\boldsymbol{\omega})$ y $\tilde{S}(\mathbf{q})$ vienen dadas por,

$$S(\boldsymbol{\omega}) = \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix} \qquad \tilde{S}(\mathbf{q}) = \begin{bmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & -q_3 & q_2 \\ q_3 & q_0 & -q_1 \\ -q_2 & q_1 & q_0 \end{bmatrix} \tag{5.51}$$

A fin de compensar el sesgo del giróscopo, es necesario incluir este término en la ecuación de la dinámica del sistema (5.50),

$$\dot{\mathbf{q}} = \frac{1}{2} S(\boldsymbol{\omega} - \mathbf{b}^g)\mathbf{q} \tag{5.52}$$

$$= \frac{1}{2} S(\boldsymbol{\omega})\mathbf{q} - \frac{1}{2} S(\mathbf{b}^g)\mathbf{q} \tag{5.53}$$

$$= \frac{1}{2} \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} + \frac{1}{2} \begin{bmatrix} 0 & -b_x^g & -b_y^g & -b_z^g \\ b_x^g & 0 & b_z^g & -b_y^g \\ b_y^g & -b_z^g & 0 & b_x^g \\ b_z^g & b_y^g & -b_x^g & 0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \tag{5.54}$$

Para poder implementar esta ecuación en un código es necesario discretizarla. En primera aproximación, vamos a considerar un esquema numérico explícito de primer orden de tipo Euler,

$$\dot{\mathbf{q}}_k = [\mathbf{q}_{k+1} - \mathbf{q}_k]/\Delta T \tag{5.55}$$

donde ΔT es el tiempo de muestreo. Reorganizando la ecuación anterior, se obtiene,

$$\mathbf{q}_{k+1} = \frac{\Delta T}{2} S(\boldsymbol{\omega})\mathbf{q}_k - \frac{\Delta T}{2} S(\mathbf{b}^g)\mathbf{q}_k + \mathbf{q}_k \tag{5.56}$$

A fin de poder reescribir la ecuación anterior de forma matricial, reorganizamos los términos para obtener

$$\mathbf{q}_{k+1} = \frac{\Delta T}{2} \tilde{S}(\mathbf{q}_k)\boldsymbol{\omega} - \frac{\Delta T}{2} \tilde{S}(\mathbf{q}_k)\mathbf{b}^g + \mathbf{q}_k \tag{5.57}$$

Finalmente, la ecuación de estado discretizada del sistema queda,

$$\mathbf{x}_{k+1} = \begin{bmatrix} I_{4\times4} & -\frac{\Delta T}{2}\tilde{S}(\mathbf{q}_k) \\ 0_{3\times4} & I_{3\times3} \end{bmatrix} \mathbf{x}_k + \begin{bmatrix} \frac{\Delta T}{2}\tilde{S}(\mathbf{q}_k) \\ 0_{3\times3} \end{bmatrix} \boldsymbol{\omega}_k \tag{5.58}$$

Substituyendo los valores reales del estado por sus estimaciones $\mathbf{x}_k \rightarrow \hat{\mathbf{x}}_k$ podemos comprobar como la ecuación anterior representa la ecuación de extrapolación del estado,

$$\hat{\mathbf{x}}_{k+1}^{-} = F_k\hat{\mathbf{x}}_k + G_k\mathbf{u}_k \tag{5.59}$$

donde, identificando términos, se llega a las expresiones de las matrices de transición y de control linealizadas del filtro de Kalman extendido,

$$F_k = \begin{bmatrix} I_{4\times4} & -\frac{\Delta T}{2}\tilde{S}(\mathbf{q}_k) \\ 0_{3\times4} & I_{3\times3} \end{bmatrix} \tag{5.60}$$

$$G_k = \begin{bmatrix} \frac{\Delta T}{2}\tilde{S}(\mathbf{q}_k) \\ 0_{3\times3} \end{bmatrix} \tag{5.61}$$

$$\mathbf{u}_k = \boldsymbol{\omega}_k \tag{5.62}$$

5.2.2. Ecuación de observación

Hasta ahora, se han utilizado las lecturas del giróscopo para obtener una estimación a priori del estado en $k+1$, $\hat{\mathbf{x}}_{k+1}^{-}$. A continuación, se describe el papel que juegan las lecturas del magnetómetro y del acelerómetro en la estimación de la actitud en el filtro de Kalman extendido. El conjunto de variables observadas, o salida del sistema, esta compuesto por el vector de dirección del campo gravitatorio, $\mathbf{g}^b$, y el vector de dirección del campo magnético, $\mathbf{m}^b$, expresados en ejes cuerpo.

$$\mathbf{y}_k = \begin{bmatrix} g_x^b & g_y^b & g_z^b & m_x^b & m_y^b & m_z^b \end{bmatrix}^T \tag{5.63}$$

Se trata de vectores unitarios que apuntan en la dirección del campo gravitatorio y magnético terrestre respectivamente. De ahora en adelante, se denota por $\hat{\mathbf{y}}_k$ al vector de variables observadas medido por los sensores en el instante k y por $\hat{\mathbf{y}}_k^{-}$ al vector de variables observadas calculado a partir de la estimación a priori del estado $\hat{\mathbf{x}}_k^{-}$. Nótese que, en realidad, el acelerómetro no mide exactamente la dirección del campo gravitatorio expresada en ejes cuerpo, $\mathbf{g}^b$, sino la aceleración a la que está sometido el sensor expresada en ejes cuerpo $\mathbf{a}^b$. Por tanto, hay una hipótesis implícita en el desarrollo: se asume que las aceleraciones parásitas a las que está sometido el sistema son pequeñas con respecto a la aceleración de la gravedad $g = 9.81\ \mathrm{m\,s^{-2}}$, de manera que se puede suponer $\mathbf{a}^b \simeq \mathbf{g}^b$. En consecuencia, el error en la predicción aumentará cuando se someta al sistema a aceleraciones significativas.

Aclarado este punto, asumimos que vector $\hat{\mathbf{y}}_k$ puede obtenerse directamente normalizando los vectores $\mathbf{a}_k$ y $\mathbf{m}_k$ medidos por el acelerómetro y el magnetómetro, previa aplicación de la calibración descrita en el Capítulo 4.

Nótese que los vectores de referencia de dirección del campo gravitatorio y del campo magnético expresados en ejes inerciales, $\mathbf{g}_r^I$ y $\mathbf{m}_r^I$ respectivamente, son conocidos,

$$\mathbf{g_r}^I = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{m}_r^I = \begin{bmatrix} m_x^I & m_y^I & m_z^I \end{bmatrix} \tag{5.64}$$

donde las componentes del vector $\mathbf{m}_r^I$ pueden obtenerse mediante herramientas tales como NCEI Geomagnetic Calculators. Conocida la matriz de rotación del sistema de ejes inerciales a ejes cuerpo $R_I^b = (R_b^I)^T$ es posible calcular las componentes de estos dos vectores en ejes cuerpo. De esta manera, podemos escribir,

$$\mathbf{y} = \begin{bmatrix} \mathbf{a}^b \\ \mathbf{m}^b \end{bmatrix} = \begin{bmatrix} R_I^b \\ R_I^b \end{bmatrix} \begin{bmatrix} \mathbf{g}_r^I \\ \mathbf{m}_r^I \end{bmatrix} \tag{5.65}$$

La matriz de rotación R_I^b tiene la siguiente expresión,

$$R_I^b = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 + q_0q_3) & 2(q_1q_3 - q_0q_2) \\ 2(q_1q_2 - q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 + q_0q_1) \\ 2(q_1q_3 + q_0q_2) & 2(q_2q_3 - q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \tag{5.66}$$

Por lo que puede expresarse en función de la estimación a priori del estado del sistema $\hat{\mathbf{x}}_k^-$,

$$\hat{\mathbf{y}}_k^- = \mathbf{h}(\hat{\mathbf{x}}_k^-) = \begin{bmatrix} R_I^b(\hat{\mathbf{x}}_k^-) \\ R_I^b(\hat{\mathbf{x}}_k^-) \end{bmatrix} \begin{bmatrix} \mathbf{g}_r^I \\ \mathbf{m}_r^I \end{bmatrix} \tag{5.67}$$

Denotando por q_i los elementos del vector de estados $\hat{\mathbf{x}}_k^-$ correspondientes al cuaternio de actitud podemos escribir,

$$\hat{\mathbf{y}}_k^- = \begin{bmatrix} (q_0^2 + q_1^2 - q_2^2 - q_3^2)g_x^I & + & 2(q_1q_2 + q_0q_3)g_y^I & + & 2(q_1q_3 - q_0q_2)g_z^I \\ 2(q_1q_2 - q_0q_3)g_x^I & + & (q_0^2 - q_1^2 + q_2^2 - q_3^2)g_y^I & + & 2(q_2q_3 + q_0q_1)g_z^I \\ 2(q_1q_3 + q_0q_2)g_x^I & + & 2(q_2q_3 - q_0q_1)g_y^I & + & (q_0^2 - q_1^2 - q_2^2 + q_3^2)g_z^I \\ (q_0^2 + q_1^2 - q_2^2 - q_3^2)a_x^I & + & 2(q_1q_2 + q_0q_3)a_y^I & + & 2(q_1q_3 - q_0q_2)a_z^I \\ 2(q_1q_2 - q_0q_3)a_x^I & + & (q_0^2 - q_1^2 + q_2^2 - q_3^2)a_y^I & + & 2(q_2q_3 + q_0q_1)a_z^I \\ 2(q_1q_3 + q_0q_2)a_x^I & + & 2(q_2q_3 - q_0q_1)a_y^I & + & (q_0^2 - q_1^2 - q_2^2 + q_3^2)a_z^I \end{bmatrix}_{\mathbf{q}=\hat{\mathbf{q}}_k^-} \tag{5.68}$$

La ecuación (5.68) es la ecuación de observación no lineal del sistema.

No obstante, para poder implementar el filtro de Kalman, es necesario calcular la Jacobiana de la función de observación $\mathbf{h}(\mathbf{x}_k)$ con respecto al vector de estados. Nótese, que es posible escribir la misma ecuación de observación para dos vectores de referencia $\mathbf{r}$ cualesquiera siempre que sus componentes en el sistema de referncia inercial $\mathbf{r}^I$ sean conocidas

y sus componentes en el sistema de referencia ejes cuerpo puedan ser medidas por algún sensor. Por tanto, de forma genérica, para un vector de referencia $\mathbf{r}$ podemos escribir,

$$\mathbf{r}^b = R_I^b \mathbf{r}^b = \begin{bmatrix} q_0^2+q_1^2-q_2^2-q_3^2 & 2(q_1q_2+q_0q_3) & 2(q_1q_3-q_0q_2) \\ 2(q_1q_2-q_0q_3) & q_0^2-q_1^2+q_2^2-q_3^2 & 2(q_2q_3+q_0q_1) \\ 2(q_1q_3+q_0q_2) & 2(q_2q_3-q_0q_1) & q_0^2-q_1^2-q_2^2+q_3^2 \end{bmatrix} \begin{bmatrix} r_x^I \\ r_y^I \\ r_z^I \end{bmatrix} \tag{5.69}$$

Expandiendo la ecuación se obtiene,

$$\mathbf{r}^b = \begin{bmatrix} (q_0^2+q_1^2-q_2^2-q_3^2)r_x^I + 2(q_1q_2+q_0q_3)r_y^I + 2(q_1q_3-q_0q_2)r_z^I \\ 2(q_1q_2-q_0q_3)r_x^I + (q_0^2-q_1^2+q_2^2-q_3^2)r_y^I + 2(q_2q_3+q_0q_1)r_z^I \\ 2(q_1q_3+q_0q_2)r_x^I + 2(q_2q_3-q_0q_1)r_y^I + (q_0^2-q_1^2-q_2^2+q_3^2)r_z^I \end{bmatrix} \tag{5.70}$$

Linealizando, se tiene,

$$\mathbf{r}^b(\mathbf{q}_{k+1}) = \mathbf{r}^b(\mathbf{q}_k) + \partial r_i^b/\partial q_j|_{\mathbf{q}=\mathbf{q}_k}(\mathbf{q}_{k+1}-\mathbf{q}_k) + \mathcal{O}\left((\mathbf{q}_{k+1}-\mathbf{q}_k)^2\right) \tag{5.71}$$

donde $\tilde{H}_k(\mathbf{r},\mathbf{q}_k) = \partial r_i^b/\partial q_j|_{\mathbf{q}=\mathbf{q}_k}$ es la matriz Jacobiana correspondiente al vector de referencia $\mathbf{r}$ evaluada en $\mathbf{q}_k$, cuya expresión general en función de $\begin{bmatrix} r_x^I & r_y^I & r_z^I \end{bmatrix}^T$ es,

$$\frac{\partial \mathbf{r}_b}{\partial \mathbf{q}} = 2 \begin{bmatrix} q_0r_x^I + q_3r_y^I - q_2r_y^I & q_1r_x^I + q_2r_y^I + q_3r_z^I & -q_2r_x^I + q_1r_y^I - q_0r_z^I & -q_3r_x^I + q_0r_y^I + q_1r_z^I \\ -q_3r_x^I + q_0r_y^I + q_1r_y^I & q_2r_x^I + -q_1r_y^I + q_0r_z^I & q_1r_x^I + q_2r_y^I + q_3r_z^I & -q_0r_x^I + -q_3r_y^I + q_2r_z^I \\ q_2r_x^I - q_1r_y^I + q_0r_y^I & q_3r_x^I - q_0r_y^I - q_1r_z^I & q_0r_x^I + q_3r_y^I - q_2r_z^I & q_1r_x^I + q_2r_y^I + q_3r_z^I \end{bmatrix} \tag{5.72}$$

De esta manera, la matriz de observación del sistema H_k obtenida como la Jacbiana de la función de observación, para los vectores de referencia $\mathbf{g}^I$ y $\mathbf{m}^I$ viene dada por,

$$H_k = \begin{bmatrix} \tilde{H}_k(\mathbf{g}_r^I, \hat{\mathbf{x}}_k^-) & 0_{3\times 3} \\ \tilde{H}_k(\mathbf{m}_r^I, \hat{\mathbf{x}}_k^-) & 0_{3\times 3} \end{bmatrix} \tag{5.73}$$

5.2.3. Aspectos prácticos de la implementación

En la práctica, la implementación del filtro requiere clarificar algunos aspectos importantes que son a menudo pasados por alto.

Inicialización

El algoritmo requiere asignar unos valores iniciales tanto al estado del sistema $\hat{\mathbf{x}}_0$ como a la matriz de covaianza P_0. En la práctica, el valor inicial asignado a estos elementos tiene poca relevancia ya que convergen rápidamente a sus valores de equilibrio. En la actual implementación, los valores iniciales por defecto son,

$$\hat{\mathbf{x}}_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \tag{5.74}$$

$$P_0 = \sigma_0^2 I_{7\times 7} \quad \text{con} \quad \sigma_0 = 0.01 \tag{5.75}$$

Matrices de ruido

El algoritmo también requiere asignar unos valores a las matrices de ruido del proceso Q y de las mediciones R. Las matrices de ruido cuantifican el grado de confianza relativo que tenemos en la estimación de la actitud obtenida de la integración del giróscopo y las lecturas de los sensores. Existe un amplio abanico de posibilidades a la hora de asignar los valores de las matrices Q y R. A menudo se asume que el ruido entre las diferentes variables de estado no esta correlacionado, por lo que la matriz Q es diagonal. Análogamente, es común asumir que el ruido de las lecturas de los sensores no esta correlacionado, luego R es diagonal. En ausencia de una mejor estimación, es posible asignar un valor homogéneo a las diagonales de R y Q y probar diferentes ratios de ruido hasta encontrar una combinación de valores que proporcione unos resultados satisfactorios. Así pues, se ha comprobado que la elección $Q = 0.001 \cdot I_{7\times7}$ y $R = 0.1 \cdot I_{6\times6}$ proporciona unos buenos resultados para la estimación de la actitud.

Sin embargo, esta opción parece ciertamente arbitraria y carente de sentido físico. Además, a falta de un sistema de validación preciso, la evaluación de los resultados queda al arbitrio del usuario. Esto hace que la elección de los valores de Q y R pueda ser subóptima. Una estimación de la matriz de ruido de las mediciones puede obtenerse calculando la desviación típica de las lecturas estáticas del acelerómetro y del magnetómetro. Es importante destacar que la estimación del ruido de los sensores debe realizarse normalizando los vectores de gravedad y campo magnético por su módulo.

$$R = \begin{bmatrix} \sigma^2_{a^b_x} & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma^2_{a^b_y} & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma^2_{a^b_z} & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma^2_{m^b_x} & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma^2_{m^b_y} & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma^2_{m^b_z} \end{bmatrix} \tag{5.76}$$

La construcción de R utilizando las varianzas de las lecturas del sensor subestima el error en las lecturas del acelerómetro y sobrestima el error en las medidas del magnetómetro. El problema radica en suponer que existe una equivalencia entre la dispersión de las medidas del acelerómetro y su desviación del valor real (diferencia entre *precision* y *accuracy* en inglés). Esta es una buena hipótesis en condiciones estáticas, pero su validez se deteriora enormemente en condiciones dinámicas, cuando el sensor está sometido a aceleraciones parásitas, lo que inevitablemente desvía el valor de las lecturas del sensor de la dirección real del campo gravitatorio. Por otro lado, la gran dispersión (alto nivel de ruido) en las medidas del magnetómetro se traduce en que la estimación del ángulo de guiñada pasa a apoyarse únicamente en las medidas del giróscopo, por lo que aparece una clara deriva en este ángulo de Euler.

Con respecto a la matriz de ruido del proceso, Q, siguiendo el modelo de error discreto

presentado en [51], podemos escribir,

$$Q = G \begin{bmatrix} \sigma^2_{\omega_x} & 0 & 0 \\ 0 & \sigma^2_{\omega_y} & 0 \\ 0 & 0 & \sigma^2_{\omega_z} \end{bmatrix} G^T \tag{5.77}$$

donde G es la matriz de control del sistema en tiempo discreto y $\sigma^2_{\omega_i}$ es la varianza de las lecturas del giróscopo en la dirección i con el sensor estático. Este modelo de ruido consiste en proyectar los errores en la estimación de la entrada del sistema, $\mathbf{u}$, en el vector de estados, $\mathbf{x}$, a través de la matriz de control del sistema discreto, G. El modelo posee un sentido físico claro, ya que tiene en cuenta la influencia de la actitud y del paso de tiempo ΔT en la magnitud del error en la estimación del estado. Sin embargo, dada la matriz de transferencia del sistema G, los valores de la entrada no tienen ningún impacto en la parte del vector de estados correspondiente al bias del giróscopo. En consecuencia, los términos de la matriz Q resultante que impliquen la varianza o covarianza de algún término b^g_i serán nulos. Tras diferentes ensayos de prueba y error, se ha comprobado que se obtienen mejores resultados cuando se aumenta artificialmente el valor de la varianza de los términos de la diagonal de Q correspondientes al sesgo del giróscopo b^g_i.

Referencia del campo magnético

Tal y como se explicó en el Capítulo 4, el magnetómetro esta sujeto a múltiples fuentes de ruido, lo que hace que sus lecturas, a pesar de la calibración, sean poco fidedignas. Por ello, cuando son utilizados en filtros de Kalman para la determinación de la actitud, una práctica común consiste en restringir su influencia sobre los ángulos de cabeceo y balance (*pitch* y *roll*) [5, 52]. Esto se consigue alterando artificialmente tanto las lecturas del campo magnético en ejes cuerpo realizadas por el sensor, $\mathbf{m}^b$, como la referencia del vector del campo magnético en ejes inerciales, $\mathbf{m}^I_r$. La referencia, $\mathbf{m}^I_r$, se modifica anulando las componentes del campo magnético en la dirección vertical y tomando un vector de referencia unitario en la dirección x_I o y_I,

$$\mathbf{m}^I_r = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \tag{5.78}$$

Por otra parte, las lecturas del sensores deben ser alteradas de para anular la parte correspondiente a la componente vertical en ejes inerciales, m^I_z. La dificultad radica en que las lecturas del sensor están expresadas en ejes cuerpo. En consecuencia, en cada iteración temporal se hace uso de la matriz de rotación, R^I_b, para transformar las lecturas del sensor a ejes inerciales, $m^b \xrightarrow{R^I_b} m^I$. A continuación, se anula la componente vertical de la lectura del campo magnético en ejes inerciales y se normaliza el vector por su nuevo módulo obteniendo el vector $\tilde{\mathbf{m}}^I$. Finalmente, se utiliza la matriz de rotación, R^b_I para volver a expresar el vector de lecturas del campo magnético en ejes cuerpo.

$$\mathbf{m}^b \xrightarrow{R^I_b} \mathbf{m}^I \rightarrow \hat{\mathbf{m}}^I = \begin{bmatrix} m^I_x & m^I_y & 0 \end{bmatrix}^T \rightarrow \tilde{\mathbf{m}}^I = \begin{bmatrix} m^I_x & m^I_y & 0 \end{bmatrix}^T / \sqrt{{m^I_x}^2 + {m^I_y}^2} \xrightarrow{R^b_I} \tilde{\mathbf{m}}_b \tag{5.79}$$

5.3. Demostración de funcionamiento

Una breve demostración del filtro de Kalman extendido descrito anteriormente corriendo en la Raspberry pi Pico y ejecutándose con un periodo de muestreo de 60 ms puede encontrarse en Demostración del Filtro de Kalman Extendido[1].

La demostración muestra como el algoritmo es capaz de estimar con una gran precisión la orientación del sensor. El vídeo muestra como en los movimientos rápidos, aparecen pequeñas desviaciones con respecto a la orientación real del sensor. Estas desviaciones pueden atribuirse a las aceleraciones parásitas medidas por el acelerómetro que desvían la dirección del vector de gravedad medido con respecto a su dirección real.

Es importante destacar que el periodo de muestreo empleado es relativamente grande. En la práctica, la Raspberry pi Pico es capaz de soportar periodos de muestreo mucho más pequeños. Sin embargo, en la presente apliación existe una limitación debido a la trasmisión de datos por el puerto de serie y el script de python utilizado para medir los valores de la actitud en tiempo real.

[1]Si el lector no es automáticamente redirigido hacia su navegador web puede consultar el vídeo en la dirección `https://www.youtube.com/watch?v=M0d3wxBY-zM&ab_channel=tfmUNED_kalman`

5.4. Header de la clase `EKF.h`

```
#ifndef EKF_H_
#define EKF_H_
//-----------------------------------------------------------------------------------------
// EXTENDED KALMAN FILTER LIBRARY
//-----------------------------------------------------------------------------------------
// Author:  Enrique Flores
// Date:    15/07/2023
// Import dependencies
#include <stdio.h>
#include "pico/stdlib.h"
#include "hardware/i2c.h"
#include <vector>
#include "../../lib/matrix_operations/matrix_operations.h"

// Euler struct to store Euler angles
struct Euler{
  // Euler angles in rad
  double yaw;
  double pitch;
  double roll;
  // Euler angles in degrees
  double yaw_deg() const;
  double pitch_deg() const;
  double roll_deg() const;
  // Compute Euler angles from attitude quaternion q
  void getEulerangles(const std::vector<double> &q);
  // Compute Euler angles from rotation matrix R
  void Rot2Euler(const std::vector<std::vector<double>> &R);
};
//#####################################################
// ------------------- EKF CLASS --------------------
//#####################################################
class EKF{
  private:
      // Covarianze matrix P --> P_{k|k}
      std::vector<std::vector<double>> P      = createDiagMatrix(7, 0.01);
      // A priori estimate of the covarianze matrix P_bar --> P_{k|k-1}
      std::vector<std::vector<double>> P_bar  = createDiagMatrix(7, 0.01);
      // Process noise matrix
      std::vector<std::vector<double>> Q      = createDiagMatrix(7, 0.001);
      // Measurements noise matrix
      std::vector<std::vector<double>> R      = createDiagMatrix(6, 0.1);
      // Kalman Gain matrix
      std::vector<std::vector<double>> K;
      // Process matrix F
```

```
    std::vector<std::vector<double>> F;
    // Process control matrix G
    std::vector<std::vector<double>> G;
    // Observation matrix H
    std::vector<std::vector<double>> H;
    // A priori estimate of the state vector \hat{x}_{k|k-1}
    std::vector<double> xhat_bar  = {1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0};
    // Estimate of the state in the previous iteration \hat{x}_{k-1|k-1}
    std::vector<double> xhat_prev = {1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0};
    // Vector of observed variables computed from a priori estimate of
    // the state
    std::vector<double> yhat_bar  = {0.0, 0.0, 0.0, 0.0, 0.0, 0.0};
    // Reference gravity vector in the Inertial frame
    std::vector<double> accel_ref = {0.0, 0.0, 1.0};
    // Reference magnetic field vector in the Inertial frame
    std::vector<double> mag_ref   = {1.0, 0.0, 0.0};
  public:
    // Attitude quaternion
    std::vector<double> q         = {1.0, 0.0 ,0.0, 0.0};
    // Estimate of the state vector
    std::vector<double> xhat      = {1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0};
    // Compute the observation Jacobian matrix H
    std::vector<std::vector<double>> getJacobianMatrix(const std::vector<double> &ref);
    // Predict the measured variables from the state vector
    std::vector<double> predictAccelMag();
    // Compute the discrete time state matrix F
    void getF(const double dt);
    // Compute the discrete time control matrix G
    void getG(const double dt);
    // Method to perform the predict stage of the Kalman filter:
    // 1. State extrapolation equation eq. 5.41
    // 2. Covariance extrapolation equation eq. 5.42
    void predict(const std::vector<double> &gyro, const double dt);
    // Method to perform the update stage of the Kalman filter:
    // 3. Kalman Gain equation eq. 5.43
    // 4. State update equation eq. 5.44
    // 5. Covariance update equation eq. 5.45
    void update(const std::vector<double> &acc, const std::vector<double> &mag);
};

#endif /* EKF_H_ */
```

Capítulo 6

Filtro complementario + TRIAD

En este capítulo se implementa un método alternativo al filtro de Kalman para la determinación de la actitud basado en las medidas de los sensores de la IMU MPU9250. El método hace uso del algoritmo TRIAD para calcular la actitud a partir de las medidas del vector de gravedad y del vector del campo magnético. A continuación, la actitud calculada por este método se combina mediante un filtro complementario con la actitud determinada mediante la integración en el tiempo de las lecturas del giróscopo.

6.1. TRIAD

El algoritmo TRIAD permite determinar la actitud del sistema de ejes cuerpo $-b$ con respecto al sistema inercial $-I$ si se conocen las componentes de dos vectores no colineales en ambos sistemas de referencia [53, 54, 55]. Se trata de un algoritmo de determinación de la actitud determinista y puede utilizarse tanto por vehículos aéreos como en satélites. De forma general, se denota por $\mathbf{w}_1$ y $\mathbf{v_1}$ al primero de los dos vectores expresado en ejes cuerpo e inerciales respectivamente. En la misma línea, se denota por $\mathbf{w}_2$ y $\mathbf{v_2}$ a las componentes en ejes cuerpo e inerciales del segundo vector.

El método proporciona mejores resultados si se escoge como vector $\mathbf{v}_1$ al vector cuyas componentes se conocen con mayor precisión. En la presente aplicación, dado que la dirección del vector de gravedad puede verse afectada por otras aceleraciones a las que este sometida la planta, se toma como vector $\mathbf{v}_1$ la dirección del campo magnético terrestre. Las dos direcciones conocidas en ejes inerciales y ejes cuerpo deben normalizarse por su

módulo para obtener vectores unitarios,

$$\mathbf{w}_1 = \begin{bmatrix} w_{1x} & w_{1y} & w_{1z} \end{bmatrix} = \begin{bmatrix} m_x^b & m_y^b & m_z^b \end{bmatrix} / \sqrt{{m_x^b}^2 + {m_y^b}^2 + {m_z^b}^2} \tag{6.1}$$

$$\mathbf{w}_2 = \begin{bmatrix} w_{2x} & w_{2y} & w_{2z} \end{bmatrix} = \begin{bmatrix} a_x^b & a_y^b & a_z^b \end{bmatrix} / \sqrt{{a_x^b}^2 + {a_y^b}^2 + {a_z^b}^2} \tag{6.2}$$

$$\mathbf{v}_1 = \begin{bmatrix} v_{1x} & v_{1y} & v_{1z} \end{bmatrix} = \begin{bmatrix} m_x^I & m_y^I & m_z^I \end{bmatrix} / \sqrt{{m_x^I}^2 + {m_y^I}^2 + {m_z^I}^2} \tag{6.3}$$

$$\mathbf{v}_1 = \begin{bmatrix} v_{2x} & v_{2y} & v_{2z} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \tag{6.4}$$

donde los vectores $\mathbf{w}_1$ y $\mathbf{w}_2$ se obtienen, en cada iteración, a partir de las medidas del magnetómetro y del acelerómetro respectivamente. El vector $\mathbf{v}_2$ puede obtenerse a partir de herramientas como NCEI Geomagnetic Calculators para una localización cualquiera definida por una longitud y una latitud. A continuación se procede de la siguiente forma,

1. Construcción de la triada $\mathcal{R} \in \mathbb{R}^3 \times \mathbb{R}^3$,

   $$\mathcal{B} = \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 \end{bmatrix} \tag{6.5}$$

   donde,

   $$\mathbf{r}_1 = \mathbf{v}_1 \qquad \mathbf{r}_2 = \frac{\mathbf{v}_1 \times \mathbf{v}_2}{|\mathbf{v}_1 \times \mathbf{v}_2|} \qquad \mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2 \tag{6.6}$$

   En la ecuación anterior, $(\times)$ denota el producto vectorial en $\mathbb{R}^3$. Nótese que la matriz $\mathcal{B}$ no cambia con el curso de las iteraciones, por lo que solo es necesario calcularla una vez, durante la inicialización del algoritmo.

2. Construcción de la triada $\mathcal{S} \in \mathbb{R}^3 \times \mathbb{R}^3$,

   $$\mathcal{A} = \begin{bmatrix} \mathbf{s}_1 & \mathbf{s}_2 & \mathbf{s}_3 \end{bmatrix} \tag{6.7}$$

   donde,

   $$\mathbf{s}_1 = \mathbf{w}_1 \qquad \mathbf{s}_2 = \frac{\mathbf{w}_1 \times \mathbf{w}_2}{|\mathbf{w}_1 \times \mathbf{w}_2|} \qquad \mathbf{s}_3 = \mathbf{s}_1 \times \mathbf{s}_2 \tag{6.8}$$

   La matriz $\mathcal{A}$ varía al cambiar la orientación del sistema ejes cuerpo y debe calcularse en cada iteración del algoritmo.

3. La matriz de rotación del sistema viene dada por,

   $$R_I^b = \mathcal{A}\mathcal{B}^T \tag{6.9}$$

   donde $R_b^I = \left(\mathcal{A}\mathcal{B}^T\right)^T = \mathcal{B}\mathcal{A}^T$

La actitud calculada a mediante el método TRIAD se expresa a través de la matriz de rotación del sistema R_b^I. La estimación de la orientación del sistema ejes cuerpo con respecto al sistema inercial obtenida mediante método TRIAD no sufre de deriva, pero proporciona lecturas irregulares debido al ruido de los sensores. Para mejorar la estimación de la actitud del sistema es posible combinar la salida del algoritmo TRIAD con la integración de las lecturas del giróscopo. Esto permite obtener una medida de la actitud suave y sin deriva.

6.2. Integración de las medidas del giróscopo

La integración de las medidas del giróscopo se obtiene mediante la ecuación,

$$\hat{\mathbf{q}}_{k+1}^{-} = \hat{\mathbf{q}}_k + \frac{\Delta T}{2}\tilde{S}(\mathbf{q})\boldsymbol{\omega} \tag{6.10}$$

donde $\boldsymbol{\omega} = \begin{bmatrix}\omega_x & \omega_y & \omega_z\end{bmatrix}$. Como ya sabemos, la integración de las medidas del giróscopo, $\hat{\mathbf{q}}^{-}$ presenta una deriva debido a las errores en la integración y al sesgo de las lecturas. Para corregir esta deriva, debemos combinar la estimación obtenida mediante la integración en el tiempo con la estimación obtenida mediante el algoritmo TRIAD, $\hat{\mathbf{q}}^{\triangle}$.

6.3. Filtro complementario

Existen diferentes formas de combinar la estimación de la actitud resultante del algoritmo TRIAD, $\hat{\mathbf{q}}^{\triangle}$, y la estimación de la actitud resultante de la integración temporal, $\hat{\mathbf{q}}^{-}$. En primera aproximación, consideramos el siguiente método de fusión de las estimaciones,

$$\hat{\mathbf{q}}_k = \alpha\hat{\mathbf{q}}_k^{-} + (1-\alpha)\hat{\mathbf{q}}_k^{\triangle} \tag{6.11}$$

donde $\alpha \in [0,\ 1]$. Inicialmente supondremos que α toma un valor constante. Una vez aplicada la ecuación (6.11), es necesario normalizar el cuaternio resultante para evitar divergencias en la integración temporal.

No obstante, nótese que la salida del algoritmo TRIAD es la matriz de rotación del sistema ejes cuerpo con respecto al sistema inercial, R_b^I. Para poder combinar las estimaciones de la actitud de ambos métodos mediante la ecuación (6.11) debemos primero transformar la matriz de rotación R_b^I en un cuaternio que represente la actitud del sistema ejes cuerpo $R_b^I \rightarrow \hat{\mathbf{q}}_k^{\triangle}$. Para transformar un cuaternio en una matriz de rotación se utiliza la ecuación (2.21) que fue presentada en el capítulo 2 y que es reproducida aquí para facilitar el seguimiento del desarrollo,

$$R_b^I = \begin{bmatrix} (q_0^2+q_1^2-q_2^2-q_3^2) & 2(q_1q_2-q_0q_3) & 2(q_1q_3+q_0q_2) \\ 2(q_1q_2+q_0q_3) & (q_0^2-q_1^2+q_2^2-q_3^2) & 2(q_2q_3-q_0q_1) \\ 2(q_1q_3-q_0q_2) & 2(q_2q_3+q_0q_1) & (q_0^2-q_1^2-q_2^2+q_3^2) \end{bmatrix} \tag{6.12}$$

El problema surge cuando queremos transformar la matriz de rotación, R_b^I en un cuaternio, ya que existe una ambigüedad en el signo: el cuaternio $\mathbf{q}$ y el cuaternio $-\mathbf{q}$ representan la misma rotación. Para reducir los errores numéricos resultantes de dividir por números próximos a cero, la conversión de matriz de rotación genérica, R, a cuaternio se realiza de

la siguiente manera,

$$\text{Si } \mathrm{tr}(R) > 0 \quad \begin{cases} s = \dfrac{1}{2\sqrt{\mathrm{tr}(R)+1}} \\ q_0 = \dfrac{1}{4s} \\ q_1 = (R_{32} - R_{23})s \\ q_2 = (R_{13} - R_{31})s \\ q_3 = (R_{21} - R_{12})s \end{cases} \tag{6.13}$$

$$\text{Si } \mathrm{tr}(R) < 0 \quad \text{y} \quad R_{11} > R_{22} \quad \text{y} \quad R_{11} > R_{33} \quad \begin{cases} s = 2\sqrt{1 + R_{11} - R_{22} - R_{33}} \\ q_0 = \dfrac{R_{32} - R_{23}}{s} \\ q_1 = \dfrac{s}{4} \\ q_2 = \dfrac{R_{12} + R_{21}}{s} \\ q_3 = \dfrac{R_{13} + R_{31}}{s} \end{cases} \tag{6.14}$$

$$\text{Si } \mathrm{tr}(R) < 0 \quad \text{y} \quad R_{22} > R_{23} \quad \begin{cases} s = 2\sqrt{1 + R_{22} - R_{11} - R_{33}} \\ q_0 = \dfrac{R_{13} - R_{31}}{s} \\ q_1 = \dfrac{R_{12} + R_{21}}{s} \\ q_2 = \dfrac{s}{4} \\ q_3 = \dfrac{R_{23} + R_{32}}{s} \end{cases} \tag{6.15}$$

$$\text{En caso contrario} \quad \begin{cases} s = 2\sqrt{1 + R_{33} - R_{11} - R_{22}} \\ q_0 = \dfrac{R_{21} - R_{12}}{s} \\ q_1 = \dfrac{R_{13} + R_{31}}{s} \\ q_2 = \dfrac{R_{23} + R_{32}}{s} \\ q_3 = \dfrac{s}{4} \end{cases} \tag{6.16}$$

donde R_{ij} corresponde al elemento en la fila i y columna j de la matriz de rotación R y $\mathrm{tr}(R)$ denota la traza o suma de los elementos de la diagonal de la matriz R. Sin embargo, al aplicar esta conversión, aparecen discontinuidades en el valor del cuaternio resultante, $\mathbf{q}$, en el entorno de ciertas orientaciones.

Estas discontinuidades pueden observarse, por ejemplo, alrededor del cuaternio $\mathbf{q} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}$, correspondiente a un giro de π radianes en torno al eje z_I, o equivalentemente,

a un ángulo de guiñada, ψ, de 180°. En este caso, el algoritmo TRIAD proporciona un cuaternio de actitud de $\mathbf{q}^{\triangle} \rightarrow \begin{bmatrix}0 & 0 & 0 & 1\end{bmatrix}$ para $\psi \xrightarrow{-} 180°$ y un cuaternio de actitud de $\mathbf{q}^{\triangle} \rightarrow \begin{bmatrix}0 & 0 & 0 & -1\end{bmatrix}$ para $\psi \xrightarrow{+} 180°$. Aquí, $\xrightarrow{-}$ y $\xrightarrow{+}$ denotan, "tendiendo a"por la izquierda y por la derecha respectivamente. Un ejemplo de los errores que aparecen en la estimación de la actitud alrededor de la orientación $\psi = \pi$ puede encontrarse en Errores en la estimación de la actitud filtro complementario + TRIAD[1].

Por otro lado, la integración en el tiempo de las medidas del giróscopo no esta sujeta a discontinuidades, con lo que al fusionar ambas medidas en el entorno de la orientación $\psi = 180°$ se produciran grandes errores en la estimación de la actitud. Concretamente, al cruzar por $\psi = 180°$ en la dirección de ψ creciente, el cuaternio de actitud estimado a través de la integración temporal se mantendrá en torno a valores cercanos $\mathbf{q} \simeq \begin{bmatrix}0 & 0 & 0 & 1\end{bmatrix}$ mientras que el algoritmo de TRIAD, proporcionara una salida cercana a $\mathbf{q}^{\triangle} \simeq \begin{bmatrix}0 & 0 & 0 & -1\end{bmatrix}$. A pesar de que estos dos cuaternios corresponden a actitudes muy próximas, se encuentran alejados en el espacio tetradimensional de los cuaternios. Por este motivo, al fusionarlos mediante la ecuación (6.11), aparecerán grandes errores en la estimación de la actitud. Estos errores se verán acrecentados por el ruido de los sensores que causaran saltos en los valores de $\hat{q}^{\triangle}$ en torno a ciertas orientaciones.

Para evitar estos problemas de discontinuidad y obtener una estimación de la actitud suave y sin saltos es necesario añadir una etapa suplementaria al algoritmo. Previa a la fusión de las estimaciónes mediante la ecuación (6.11), se calculan la norma entre la estimación de la actitud del algoritmo TRIAD $\hat{\mathbf{q}}_k^{\triangle}$ y la estimación de la actitud resultante de la integración temporal $\hat{\mathbf{q}}_k^{-}$,

$$n^{+} = |\hat{\mathbf{q}}_k^{\triangle} - \hat{\mathbf{q}}_k^{-}| \tag{6.17}$$

así como la norma entre $-\hat{\mathbf{q}}_k^{\triangle}$ y $\hat{\mathbf{q}}_k^{-}$,

$$n^{-} = |-\hat{\mathbf{q}}_k^{\triangle} - \hat{\mathbf{q}}_k^{-}| \tag{6.18}$$

Finalmente, se toma como estimación del algoritmo TRIAD, el cuaternio $\hat{\mathbf{q}}_k^{\triangle}$ o $-\hat{\mathbf{q}}_k^{\triangle}$ cuya distancia a $\hat{\mathbf{q}}_k^{-}$ en el espacio tetradimensional sea menor. De entre las dos cuaternios que representan la misma actitud, se promedia la salida del integrador con aquel cuya norma en el espacio $\mathbb{R}^4$ es menor, lo que permite evitar saltos en la estimación global de la actitud $\hat{\mathbf{q}}$.

6.4. Aspectos prácticos de la implementación

A fin de acelerar la convergencia inicial del filtro TRIAD+Complementario en la estimación de la actitud, se ha añadido al filtro una fase de inicialización para establecer las condiciones iniciales de la integración temporal. Esta fase de inicialización proporciona una

[1]Si el lector no es automáticamente redirigido hacia su navegador web puede consultar el vídeo en la dirección `https://www.youtube.com/watch?v=LD8TtckTH1Y&ab_channel=tfmUNED_kalman`

estimación de la actitud en $k = 0$ para la integración de la actitud en el tiempo. La estimación inicial se obtiene iterando $n_{\triangle}$ veces el algoritmo TRIAD y promediando el cuaternio de actitud resultante de todas las iteraciones. Durante esta fase, se requiere que el sensor esté estático para maximizar la precisión de la estimación. La estimación de la actitud utilizando únicamente el método TRIAD proporciona una salida ruidosa, por lo que no es apropiada para su uso dinámico. Sin embargo, promediando sobre un gran número de iteraciones, la estimación de la actitud resultante permite obtener buenos resultados para inicializar el algoritmo.

6.5. Demostración de funcionamiento

Una breve demostración del funcionamiento del algoritmo implementado ejecutándose en la Raspberry pi Pico con un periodo de muestreo de 60 ms puede encontrarse en Demostración Filtro Complementario + TRIAD[2].

De nuevo, podemos observar que el filtro presenta una buena respuesta dinámica. La actitud estimada por el algoritmo es capaz de reproducir la orientación real del sensor. Cualitativamente, no se observan grandes diferencias con respecto al filtro de Kalman.

6.6. Comparación con el filtro de Kalman

A continuación, se comparan las estimaciones de la actitud de la planta proporcionadas por los dos algoritmos presentados. Para ello, se evalúan tres series temporales en las que se generan oscilaciones en la actitud según los tres ejes de giro, ϕ, θ y ψ. La figura 6.1 representa la evolución de los ángulos de Euler estimados por el filtro de Kalman y por el filtro TRIAD+complementario cuando se realizan oscilaciones en el ángulo de cabeceo ϕ. Como podemos comprobar, existe un pequeño *offset* entre los ángulos de cabeceo y balance estimados por los dos algoritmos. Para el ángulo de balance, ϕ, la diferecia es de aproximadamente $\Delta\phi \simeq 5°$ mientras que para el ángulo de cabeceo es $\Delta\theta \simeq 3.3°$.

Resultados similares se muestran en las figuras 6.2 y 6.3 para los ángulos de balance, θ, y guiñada, ψ, respectivamente. Existe un aspecto interesante de la figura 6.2 que merece la pena destacar. Pese a que en esta serie temporal únicamente se han generado oscilaciones en el ángulo de cabeceo, θ, los datos también muestran pequeñas oscilaciones de los ángulos de balance y guiñada. Estas oscilaciones están en fase con las perturbaciones principales, lo que sugiere la existencia de un ligero acoplamiento entre los diferentes ángulos de Euler. Parte de este acoplamiento se debe, sin lugar a dudas, a las limitaciones del dispositivo

[2]Si el lector no es automáticamente redirigido hacia su navegador web puede consultar el vídeo en la dirección `https://www.youtube.com/watch?v=BQJR56gZeAg&ab_channel=tfmUNED_kalman`

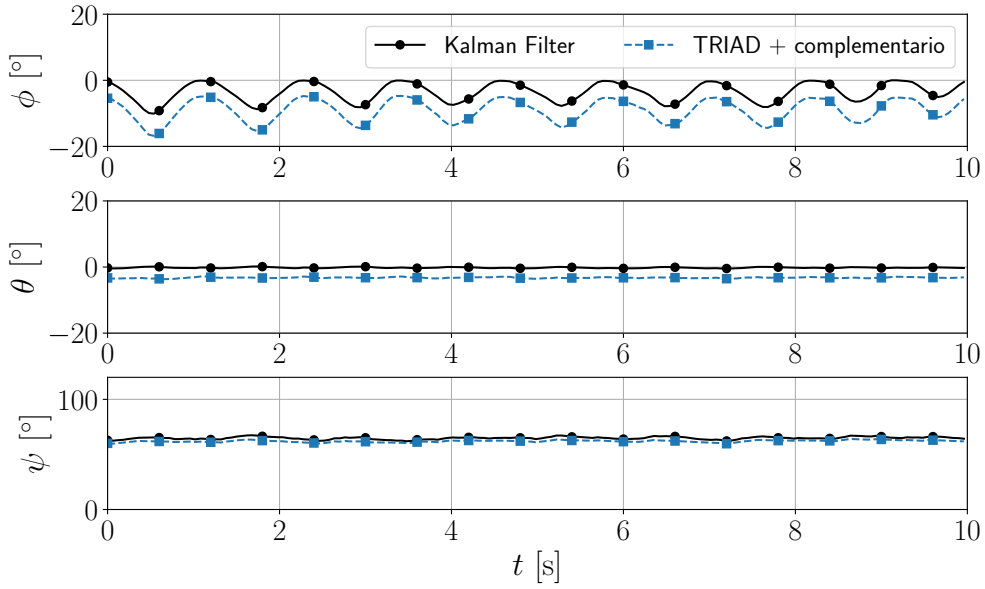

Figura 6.1: Comparación entre el filtro de Kalman y el filtro TRIAD + complementario para entradas en ϕ

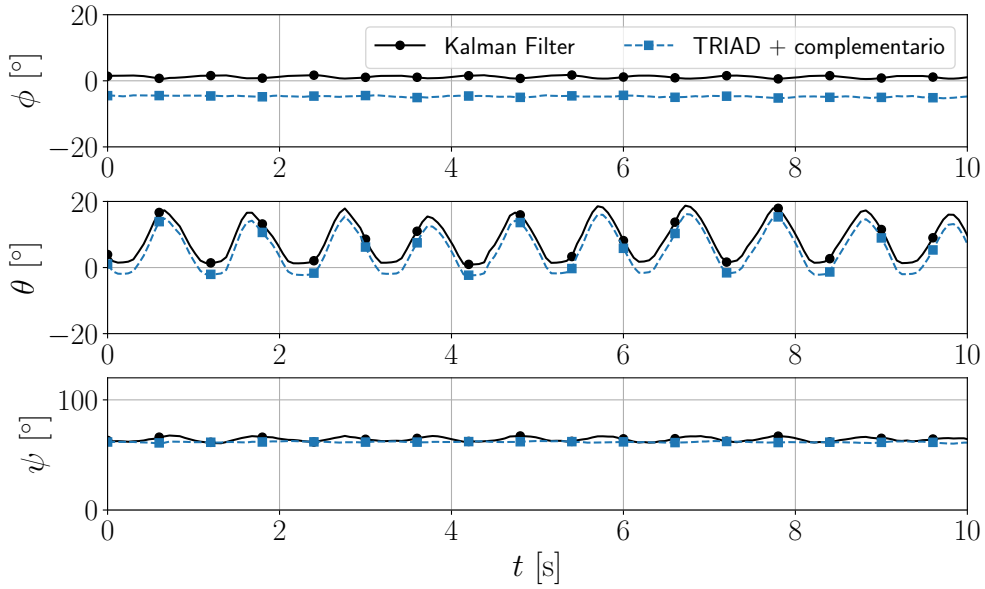

Figura 6.2: Comparación entre el filtro de Kalman y el filtro TRIAD + complementario para entradas en θ

experimental, que impide generar entradas completamente independientes. Sin embargo, el hecho de que este desacoplamiento es menor en el algoritmo TRIAD sugiere que parte del origen se encuentra en la fusión de los datos.

En resumen, el algoritmo TRIAD+complementario proporciona una estimación de la

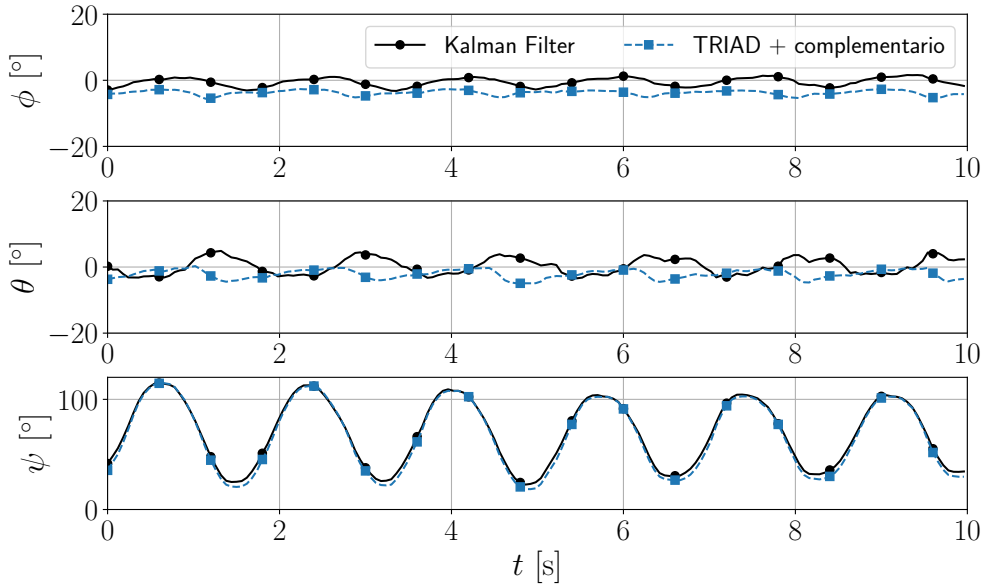

Figura 6.3: Comparación entre el filtro de Kalman y el filtro TRIAD + complementario para entradas en ψ

actitud de una calidad comparable al filtro de Kalman. Este algoritmo involucra un menor número de operaciones y, a diferencia del filtro de Kalman, no requiere la inversión de matrices para ejecutarse. En la presente implementación, se han combinado las dos estimaciones de la actitud de la manera mas simple posible. Un futuro desarrollo del filtro podría considerar métodos de fusión más sofisticados que impliquen un filtrado pasa baja de los datos del acelerómetro y del magnetómetro y un filtro pasa alta de los datos del giróscopo.

En cuanto a sus prestaciones, el filtro complementario presenta un ligero *offset* en los ángulos de cabeceo y balance. Aunque este pequeño error parece despreciable, hemos de tener en cuenta que el control de la actitud de un vehículo es muy sensible a pequeños ángulos de inclinación. Por ello, en un futuro será necesario encontrar un método para corregir esta pequeña desviación, a fin de que el algoritmo pueda ser utilizado en aplicaciones prácticas.

6.7. Header de la clase TCF.h

```
#ifndef TCF_H_
#define TCF_H_
//-------------------------------------------------------------------------------------
// TRIAD + COMPLEMENTARY FILTER LIBRARY
//-------------------------------------------------------------------------------------
// Author:  Enrique Flores
// Date:    15/07/2023
// Import dependencies
#include <stdio.h>
#include "pico/stdlib.h"
#include "hardware/i2c.h"
#include <vector>
#include "../../lib/matrix_operations/matrix_operations.h"
#include "../../lib/EKF/EKF.h"
#include "../../lib/IMU/IMU.h"

//#####################################################
// ------------------- TCF CLASS --------------------
//#####################################################
class TCF{

  private:
    // Magnetic field vector reference in mT
    std::vector<double> mag_I           = {23.872, -0.6498 , 39.991};
    // Gravity vector reference
    std::vector<double> acc_I           = {0.0, 0.0, 9.81};
    // R triad matrix
    std::vector<std::vector<double>> A  = createDiagMatrix(3, 1.0);
    // S triad matrix
    std::vector<std::vector<double>> B  = createDiagMatrix(3, 1.0);

  public:
    // Attitude quaternion estimation
    std::vector<double> qHat        = {1.0, 0.0 ,0.0, 0.0};
    // Attitude quaternion estimation from TRIAD algorithm
    std::vector<double> qTriad      = {1.0, 0.0 ,0.0, 0.0};
    // Attitude quaternion estimation in previous iteration
    std::vector<double> qHat_prev   = {1.0, 0.0 ,0.0, 0.0};
    // Attitude quaternion estimation from time integration
    std::vector<double> qHat_bar  = {1.0, 0.0 ,0.0, 0.0};
    // Euler angles computed from TRIAD algorithm
    Euler triad_yawpitchroll;
    // Euler angles estimated by the TRIAD+Complementary filter
    // algorithm
    Euler tcf_ypr;
```

```
    // Initialize Complementary Filter: Compute R triad and
    // get initial conditions for time integration
    void initialize(IMU &imu);
    // Estimate system attitude using TRIAD algorithm
    void TRIAD(const std::vector<double> &acc, const std::vector<double> &mag);
    // Integrate gyroscope measurements
    void integrate(const std::vector<double> &gyro, const double dt);
    // Combine TRIAD estimation with time integration using a
    // complementary filter
    void combine();
};

#endif /* TCF_H_ */
```

Capítulo 7

Diseño y evaluación en lazo abierto de un control de actitud

En el presente capítulo se diseña un controlador para realizar el control de actitud de un quadrotor. El controlador se diseña para el modelo linealizado del sistema en torno a una condición de vuelo a punto fijo (*hoover*). A continuación se evalúa la respuesta del controlador en lazo abierto a perturbaciones de los ángulos de Euler (yaw, pitch y roll) generadas mediante el dispositivo experimental presentado en el capítulo 3. Los ángulos de Euler se determinan utilizando el algoritmo de estimación de actitud implementado en el capítulo 5.

7.1. Diseño del controlador

Las ecuaciones que gobiernan la dinámica de la actitud de un quadrotor son [56, 57, 58, 59],

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \dfrac{\sin\phi}{\cos\theta} & \dfrac{\cos\phi}{\cos\theta} \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{7.1}$$

$$\begin{bmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix} = \begin{bmatrix} \dfrac{J_y - J_z}{J_x}\omega_y\omega_z \\ \dfrac{J_z - J_x}{J_y}\omega_x\omega_z \\ \dfrac{J_x - J_y}{J_z}\omega_x\omega_y \end{bmatrix} + \begin{bmatrix} \dfrac{\tau_\phi}{J_x} \\ \dfrac{\tau_\theta}{J_y} \\ \dfrac{\tau_\psi}{J_z} \end{bmatrix} \tag{7.2}$$

donde ψ, θ y ϕ son los ángulos de *yaw*, *pitch* y *roll* respectivamente y $\boldsymbol{\omega} = \begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix}$ es la velocidad angular del vehículo con respecto al sistema de referencia inercial. Por otro

lado,

$$\mathbf{J} = \begin{bmatrix} J_x & J_{xy} & J_{xz} \\ J_{yx} & J_y & J_{yz} \\ J_{zx} & Jzy & J_z \end{bmatrix}^b \tag{7.3}$$

es la matriz de inercia del vehículo expresada en ejes cuerpo. Normalmente, se asume que la configuración del vehículo es simétrica alrededor de las tres direcciones x_b, y_b y z_b por lo que los productos de inercia son nulos, $J_{xy} = J_{yz} = J_{zx} = 0$. Finalmente, los términos τ_ϕ, τ_θ y τ_ψ representa los momentos de balance, cabeceo y guiñada respectivamente. Estos últimos constituyen las entradas del sistema.

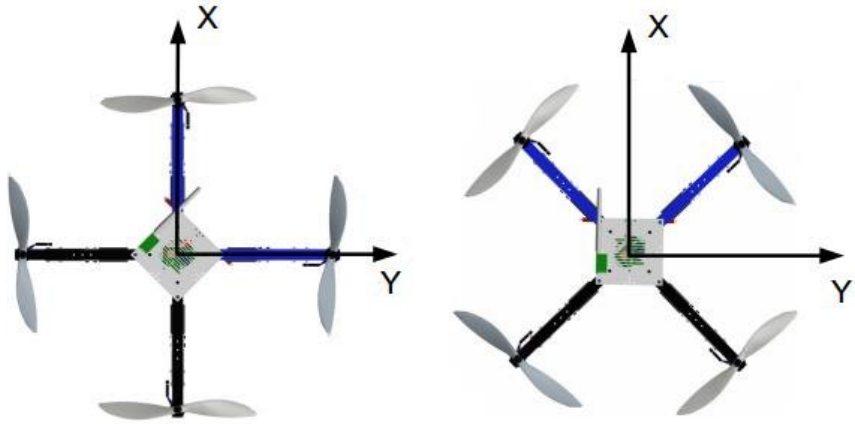

Figura 7.1: Distribución de los rotores en la configuración en + (izquierda) y × (derecha) [60].

Típicamente, se asume que tanto la fuerza como el torque producido por los motores es proporcional al cuadrado de la velocidad angular de estos [57].

$$F_i = \tilde{k}_f \Omega_i^2 \tag{7.4}$$
$$M_i = \tilde{k}_m \Omega_i^2 \tag{7.5}$$

donde $\tilde{k}_f$ y $\tilde{k}_m$ son las constantes de proporcionalidad entre la velocidad angular del motor i, Ω_i, y la fuerza de sustentación y el par de giro proporcionados por dicho motor respectivamente. Suponiendo que la velocidad angular es directamente proporcional a la señal de entrada PWD (Pulse Width Modulation) enviada a cada motor, $0 < U_i < 1$, la fuerza y el par de giro del motor i pueden expresarse como,

$$F_i = k_f U_i^2 \tag{7.6}$$
$$M_i = k_m U_i^2 \tag{7.7}$$

Para un quadrotor en configuración de equis, × (ver Fig. 7.1), cuyos motores se numeran en sentido horario comenzando por el motor situado en el primer cuadrante de los ejes cuerpo, el par de giro de balance viene dado por,

$$\tau_\phi = -lF_1 - lF_2 + lF_3 + lF_4 \tag{7.8}$$

donde l es la distancia de los motores al eje x_b. Así mismo, el par de giro de cabeceo se expresa como,

$$\tau_\theta = lF_1 - lF_2 - lF_3 + lF_4 \tag{7.9}$$

donde se ha asumido que la distancia de los motores al eje y_b es l. Finalmente, suponiendo que el motor $i = 1$ gira en sentido antihorario, el par de giro en guiñada viene dado por,

$$\tau_\psi = M_1 - M_2 + M_3 - M_4 \tag{7.10}$$

Para que el drone se mantenga en equilibrio, es necesario satisfacer que la componente vertical de la suma de las fuerzas de los motores es igual al peso del vehículo. Para valores pequeños de los ángulos de cabeceo y balance, esta condición se expresa como,

$$mg = \sum_{i=1}^{4} F_i \tag{7.11}$$

donde m es la masa del quadrotor en kilogramos. Matricialmente, podemos escribir estas ecuaciones como,

$$\begin{bmatrix} mg \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} k_f & k_f & k_f & k_f \\ -lk_f & -lk_f & lk_f & lk_f \\ lk_f & -lk_f & -lk_f & lk_f \\ k_m & -k_m & -k_m & k_m \end{bmatrix} \begin{bmatrix} U_1^2 \\ U_2^2 \\ U_3^2 \\ U_4^2 \end{bmatrix} = \mathcal{M} \begin{bmatrix} U_1^2 \\ U_2^2 \\ U_3^2 \\ U_4^2 \end{bmatrix} \tag{7.12}$$

Lo que permite obtener las consignas de los motores a partir de los momentos de giro en las tres direcciones del espacio mediante la ecuación,

$$\begin{bmatrix} U_1^2 \\ U_2^2 \\ U_3^2 \\ U_4^2 \end{bmatrix} = \mathcal{M}^{-1} \begin{bmatrix} mg \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} \tag{7.13}$$

Los valores numéricos de las constantes del modelo aparecen recogidos en la tabla 7.1.

El vector de estados del sistema viene dado por,

$$\mathbf{x} = \begin{bmatrix} \phi & \theta & \psi & \omega_x & \omega_y & \omega_z \end{bmatrix} \tag{7.14}$$

El vector de entradas es,

$$\mathbf{u} = \begin{bmatrix} \tau_\phi & \tau_\theta & \tau_\psi \end{bmatrix} \tag{7.15}$$

J_x [kg m^2]	J_y [kg m^2]	J_z [kg m^2]	l [m]	k_f [N input^{-1}]	k_m [N m input^{-1}]	m [kg]	g [m s^{-2}]
0.01	0.01	0.02	0.1	5	0.5	1.0	9.81

Tabla 7.1: Valores numéricos de las constantes del modelo

y la salida son los ángulos de Euler de balance y cabeceo y junto con la derivada temporal del ángulo de guiñada,

$$\mathbf{y} = \begin{bmatrix} \phi & \theta & \dot{\psi} \end{bmatrix} \tag{7.16}$$

Nótese que en el caso del ángulo de guiñada, lo que se busca controlar es su tasa de variación, ya que la dinámica del vehículo es invariante a cambios en el ángulo de *yaw*. Otra forma de verlo es que para estabilizar el vehículo no se requiere un valor específico del ángulo de dirección, sino que el *heading* del drone sea constante. Para diseñar el controlador, linealizamos el sistema dado por las ecuaciones (7.1) y (7.2) en torno a una condición de vuelo estacionario a punto fijo,

$$\mathbf{x}_0 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{7.17}$$

$$\mathbf{y}_0 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \tag{7.18}$$

La matriz de transferencia del sistema linealizado en torno a esta condición de equilibrio viene dada por,

$$G(s) = \begin{bmatrix} \dfrac{100}{s^2} & 0 & 0 \\ 0 & \dfrac{100}{s^2} & 0 \\ 0 & 0 & \dfrac{50}{s} \end{bmatrix} \tag{7.19}$$

Dado que las entradas y las salidas del sistema están desacopladas entre sí, se diseña un controlador descentralizado formado por una matriz diagonal de controladores PID, $K_d(s)$.

$$K_d(s) = \begin{bmatrix} PID_\phi(s) & 0 & 0 \\ 0 & PID_\theta(s) & 0 \\ 0 & 0 & PID_{\dot{\psi}}(s) \end{bmatrix} \tag{7.20}$$

donde,

$$PID(s) = K_P + \frac{K_I}{s} + K_D \frac{s}{T_f s + 1} \tag{7.21}$$

Los coeficientes K_P, K_I, K_D, T_f representan la ganancia proporcional, integral, derivativa y la constante de tiempos del filtro del PID respectivamente. La constante de tiempos del filtro, T_f, determina la frecuencia de corte del filtro pasa baja utilizado para procesar la derivada temporal del error. Esto permite suavizar la respuesta del sistema en presencia de ruido, lo que es crucial en aplicaciones prácticas. Para determinar los parámetros del

controlador, se define un sistema con bloques PID ajustables, `tunablePID`, y se emplea la función de MATLAB `looptune`. La determinación de los coeficientes K_P, K_I, K_D, y T_f para los lazos de control de cabeceo, balance y guiñada se realiza en el sistema linealizado en torno al punto de equilibrio de vuelo a punto fijo y arroja los siguientes valores numéricos para las constantes de los controladores PID.

	K_P	K_I	K_D	T_f
$\phi \rightarrow \tau_\phi$	0	0	0.0514	0.232
$\theta \rightarrow \tau_\theta$	0	0	0.0514	0.232
$\dot{\psi} \rightarrow \tau_{\dot{\psi}}$	0.0252	0.0251	0.0251	2

Tabla 7.2: Constantes de los PID para los diferentes lazos de control del controlador.

Los detalles sobre la herramienta looptune de MATLAB pueden encontrarse en *MathWorks:looptune*. El uso de la función looptune requiere especificar un rango de frecuencias $[\omega_{min},\ \omega_{max}]$. El comando looptune trata de ajustar los lazos en el sistema de control de forma tal que la ganancia en lazo abierto cruce el nivel 0 dB dentro del rango de frecuencias especificado. La función puede recibir argumentos adicionales para realizar el ajuste tales como el margen de ganancia y fase, el número de inicializaciones aleatorias de los parámetros para su optimización y la tasa mínima de *decay* de los polos del sistema en lazo cerrado. Teniendo en cuenta los tiempos característicos de la dinámica de un quadrotor de pequeño tamaño, se ha fijado un rango de frecuencias de $[\omega_{min},\ \omega_{max}] = [10^2,\ 10^3]$ rad/s para el ajuste del controlador. No se ha especificado ni un decaimiento mínimo de los polos del sistema en lazo cerrado, `MinDecay`, ni un número de inicializaciones aleatorias, `RandomStart`.

Para lograr un buen comportamiento del controlador, ha sido necesario reducir manualmente la ganancia en el lazo de control que pilota el giro en guiñada. En caso contrario, el controlador se vuelve demasiado sensible a la presencia de pequeñas velocidades angulares que aparecen como resultado del giros en las otras direcciones del espacio. El proceso de búsqueda de los coeficientes de los PID es iterativo. Para asignar los valores finales recogidos en la tabla 7.2 se han utilizado las lecturas de la actitud de la planta real estimadas por el filtro de Kalman. Gracias al uso de las medidas reales ha sido posible detectar esta sensibilidad del controlador al giro en guiñada, lo que pone en valor el análisis en lazo abierto del controlador de la sección 7.3.

7.2. Respuesta del controlador

7.2.1. Matriz de transferencia del lazo $L(s)$ y diagrama de Nyquist generalizado.

La matriz de transferencia del lazo $L = G(s)K(s)$ determina la estabilidad del lazo si no existen cancelaciones entre los ceros y los polos de G y K. Para el modelo de actitud del drone y el controlador desarrollados en la sección anterior se tiene,

$$L(s) = \begin{bmatrix} \dfrac{22.12}{s^2 + 4.307s} & 0 & 0 \\ 0 & \dfrac{22.12}{s^2 + 4.307s} & 0 \\ 0 & 0 & \dfrac{1.89s^2 + 1.885s + 0.627}{s^3 + 0.5s^2} \end{bmatrix} \tag{7.22}$$

La matriz de transferencia del lazo $L(s)$ tiene siete polos reales: $p_1 = -4.3070$, $p_2 = -4.3070$, $p_3 = -0.5$, $p_{4,5,6,7} = 0$ y un cero complejo conjugado $z_{1,2} = -0.4986 \pm 0.2884i$. Todos los polos tienen parte real negativa o nula (estabilidad indiferente) por lo que el sistema no es inestable en lazo abierto.

El diagrama de Nyquist directo (DNA) de un sistema multivariable es el conjunto de los diagramas de Nyquist de cada uno de los elementos de la matriz $L(s)$. Para un sistema SISO, el diagrama de Nyquist representa la evolución de la parte real (eje de abscisas) e imaginaria (eje de ordenadas) de la función de transferencia cuando se varía la frecuencia $s \in (-\infty, +\infty)$. La ganancia de la función de transferencia viene dada por la distancia al origen (coordenada radial) y la fase por la coordenada azimutal. El criterio de Nyquist establece que, siendo P el número de polos con parte real positiva de la función de transferencia en lazo abierto $L(s)$, N el número de circunscripciones del punto $(-1, 0)$ en sentido horario del contorno de Nyquist, Γ (las circunscripciones en sentido anti-horario restan, es decir, se cuentan como valores de N negativos), y Z el número de polos del sistema en lazo cerrado con parte real positiva,

$$Z = N + P. \tag{7.23}$$

Para que el sistema en lazo cerrado sea estable, Z debe ser nulo y, por tanto, el diagrama de Nyquist debe realizar tantas circunvalaciones del punto $(-1, 0)$ en sentido anti-horario como polos con parte real positiva tenga la matriz de transferencia $L(s)$ del sistema en lazo abierto. El contorno de Nyquist debe recorrerse en el sentido creciente de s para determinar su sentido.

El criterio de Nyquist generalizado para sistemas MIMO [61] establece que la estabilidad

de un sistema MIMO puede analizarse mediante el diagrama de Niquist de $\det(I + L(j\omega))$ [62]. Si la matriz $L(s)$ es estable (no tiene polos con parte real positiva), el sistema en lazo cerrado será estable si el diagrama de Nyquist de $\det(I + L(j\omega))$ no encierra al origen. Generalmente, a fin de utilizar el mismo análisis que el empleado en sistemas SISO, se representa la función escalar,

$$l^{\star}(j\omega) = \det(I + L(j\omega)) - 1 \tag{7.24}$$

y se comprueban los rodeos o circunscripciones del punto $(-1,\, 0)$.

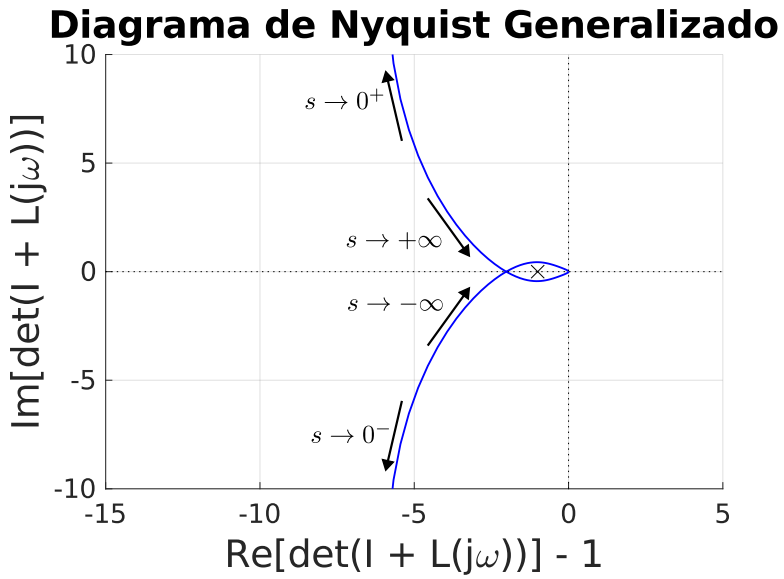

Figura 7.2: Diagrama de Nyquist generalizado del modelo de actitud linealizado y el controlador descentralizado

La figura 7.2 representa la parte imaginaria de la función $l^{\star}(j\omega)$ frente a su parte real, es decir, el diagrama de Nyquist generalizado del modelo de actitud y el controlador descentralizado desarrollados en la sección precedente. La matriz de transferencia del lazo $L(s)$ no tiene polos con parte real positiva por lo que $P = 0$. El diagrama de Nyquist generalizado presenta tantas circunscripciones negativas como positivas del punto $(-1, 0)$, por tanto $N = 0$ y, en consecuencia $Z = 0$, luego el sistema en lazo cerrado es estable.

7.2.2. Matriz de sensibilidad complementaria $T(s)$.

La matriz de sensibilidad complementaria $T = [I + G(s)K(s)]^{-1} G(s)K(s)$ determina el efecto de la realimentación en el rechazo del ruido. Esta matriz también define la respuesta de las salidas frente a las señales de referencia. Dado que las entradas y las salidas están desacopladas, la matriz de transferencia en lazo cerrado $T(s)$ es diagonal y los sus términos no nulos tienen la siguiente expresión,

$$T_{11}(s) = \frac{22.12}{s^2 + 4.307s + 22.12} \tag{7.25}$$

$$T_{22}(s) = \frac{22.12}{s^2 + 4.307s + 22.12} \tag{7.26}$$

$$T_{33}(s) = \frac{1.8903(s^2 + 0.9972s + 0.3317)}{(s + 1.325)(s^2 + 1.066s + 0.4733)} \tag{7.27}$$

Los polos de la matriz son: $p_1 = -1.3247$, $p_{2,3} = -2.1535 \pm 4.1808i$, $p_{4,5} = -2.1535 \pm 4.1808i$ y $p_{6,7} = -0.5328 \pm 0.4353i$. Todos los polos tienen parte real negativa luego el sistema $T(s)$ es estable. Una condición necesaria para todo controlador es que su matriz de sensibilidad complementaria en estado estacionario $s = 0$ sea una matriz unidad,

$$T(s = 0) = I_{n \times n}. \tag{7.28}$$

Lo que se traduce en que las salidas se igualan a los valores de referencia para tiempos suficientemente grandes $\mathbf{y} \xrightarrow{t \to \infty} \mathbf{r}$. El controlador presentado en la sección 7.1 satisface la condición dada por la eq. (7.28).

Dos herramientas que permiten analizar el comportamiento de la tupla control-planta son la respuesta a una entrada escalón y el diagrama de Bode de la matriz $T(s)$. La figura 7.3 representa el diagrama de Bode de los términos de la diagonal de la matriz de transferencia en lazo cerrado $T(s)$. Para los lazos de control en ϕ y θ, la respuesta en lazo cerrado es razonablemente plana en amplitud para frecuencias menores a $\omega = 2$ rad s^{-1}. Estos bucles de control presentan una ligera amplificación en torno a los $\omega \simeq 4$ rad s^{-1} y, a continuación, la magnitud de la respuesta y la fase decrecen rápidamente. En el lazo de control en $\dot{\psi}$, la amplitud de la respuesta en lazo cerrado comienza a decrecer para frecuencias en torno al 1 rad s^{-1}.

La respuesta temporal del sistema a un cambio en los valores de referencia $\mathbf{r} = \begin{bmatrix} \phi_r & \theta_r & \dot{\psi}_r \end{bmatrix}$ puede evaluarse mediante la respuesta a una entrada escalón del sistema en lazo cerrado. La figura 7.4 representa la respuesta a una entrada escalón unitario de los términos de la diagonal de la matriz de transferencia en lazo cerrado $T(s)$. Los bucles de control llevan la salida del sistema hacia los valores de referencia en pocos segundos. Los lazos de control presentan un cierto grado de *overshooting* en la respuesta. No obstante, dado que el modelo dinámico del sistema no incluye elementos disipativos tales como la fuerza de fricción con el aire o los efectos de masa añadida, un menor grado de desbordamiento es esperable en la planta real. Cabe destacar que el lazo de control en $\dot{\psi}$ presenta un tiempo característico mayor, por lo que requiere más tiempo para alcanzar el valor de referencia.

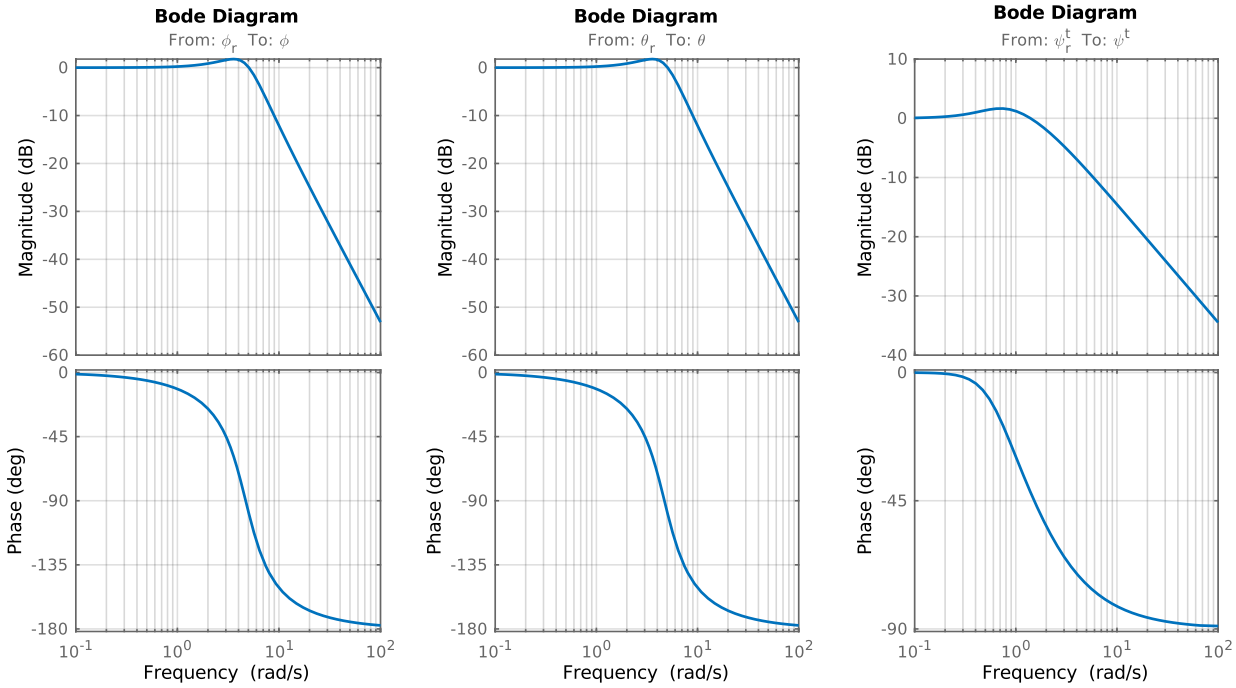

Figura 7.3: Diagrama de Bode de los términos de la diagonal de la matriz de transferencia en lazo cerrado del sistema $T(s)$

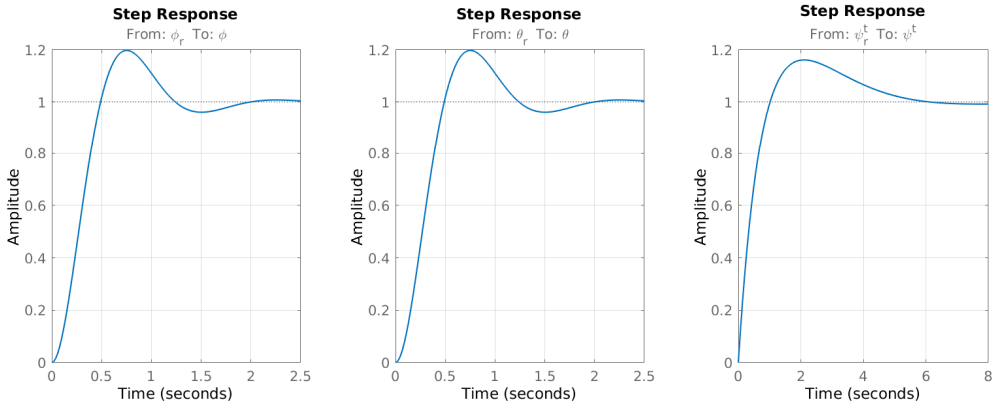

Figura 7.4: Respuesta escalón de los términos de la diagonal de la matriz de transferencia en lazo cerrado del sistema $T(s)$

7.3. Evaluación del controlador

En esta sección se generan una serie de entradas utilizando el dispositivo experimental presentado en el capítulo 3 y el algoritmo de estimación de la actitud desarrollado en el capítulo 5. A continuación, suponiendo unos valores de referencia para las salidas del

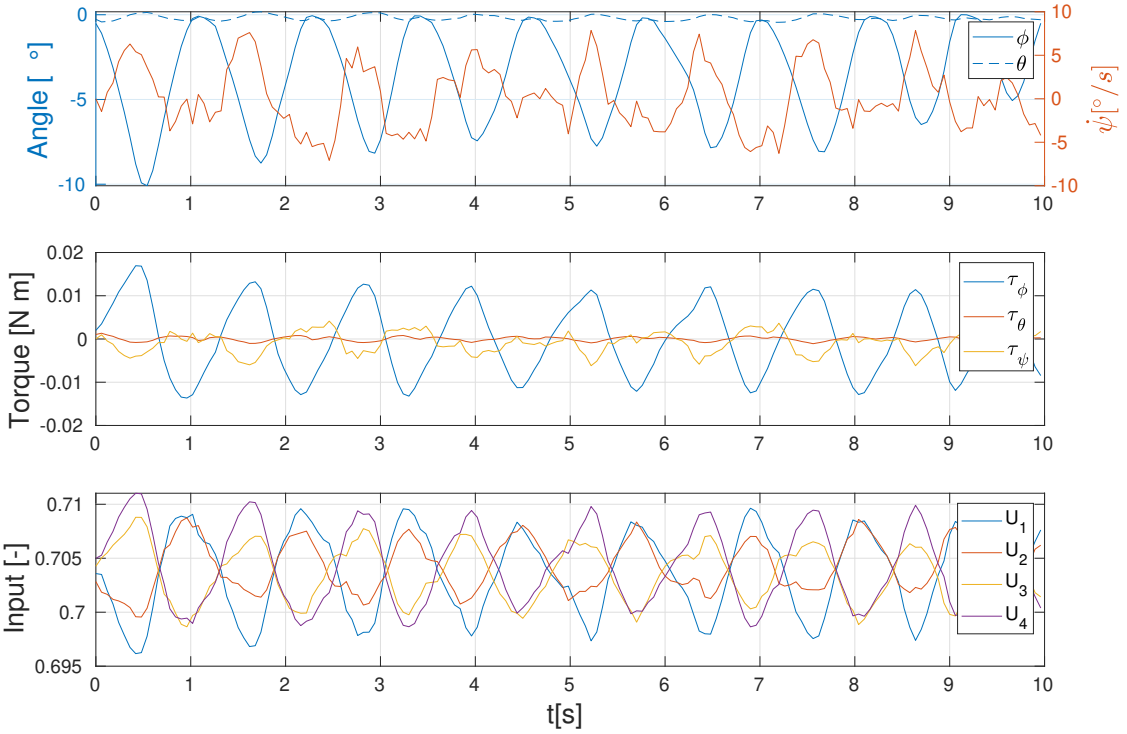

Figura 7.5: Respuesta del control a una entrada oscilatoria en ϕ

modelo de,

$$\mathbf{r} = \begin{bmatrix} \phi_r & \theta_r & \dot{\psi}_r \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \tag{7.29}$$

se evalúa la respuesta del controlador diseñado en la sección 7.1.

Para evaluar el comportamiento del control se utiliza la estimación de los ángulos de Euler proporcionada por el filtro de Kalman. Analizamos la salida del controlador para las series temporales presentadas en el capítulo anterior (ver figuras 6.1, 6.2 y 6.3).

Tengamos en cuenta que el algoritmo de fusión de datos nos proporciona el valor absoluto del ángulo de guiñada, ψ, pero el control de actitud se realiza sobre la tasa de variación de este ángulo, $\dot{\psi}$. Por ello, debemos primero calcular la derivada temporal del ángulo de guiñada. El cálculo de la derivada en una señal con ruido es siempre problemático ya que el ruido se amplifica en la derivación numérica.

Para garantizar el funcionamiento del control, es necesario realizar un filtrado pasa bajos de la derivada $\dot{\psi}$. El esquema numérico que permite calcular la derivada de la señal $\psi(t)$ realizando un filtrado pasa bajos viene dado por,

$$\dot{\psi}_k = \dot{\psi}_{k-1} + \frac{\Delta T}{RC + \Delta T}\left(\frac{\psi_k - \psi_{k-1}}{\Delta T} - \dot{\psi}_{k-1}\right) \tag{7.30}$$

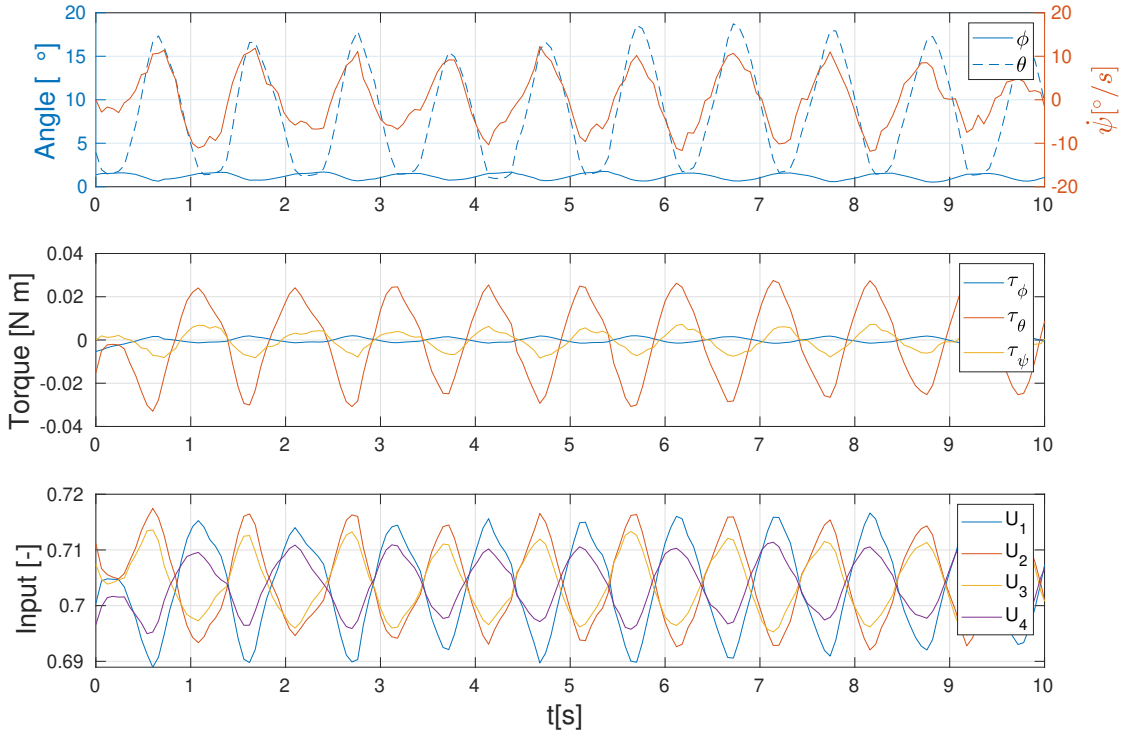

Figura 7.6: Respuesta del control a una entrada oscilatoria en θ

donde $RC = (2\pi f_c)^{-1}$, siendo $f_c = 0.7958$ Hz la frecuencia de corte del filtro.

A continuación, se calcula el error entre las lecturas y la referencia, (7.29) y se pasan las señales de error $\mathbf{e} = \mathbf{r} - \mathbf{y}$ al algoritmo de control. El resultado es la evolución temporal de las salidas del control $\mathbf{u} = \begin{bmatrix} \tau_\phi & \tau_\theta & \tau_\psi \end{bmatrix}$ y las señales de comando enviadas a los motores U_i con $i = 1, 2, 3, 4$. Para calcular las señales de comando se utiliza la ecuación eq. (7.13) y a continuación se toma la raíz cuadrada de cada elemento del vector solución. Los resultados se han representado en las figuras 7.5, 7.6 y 7.7. En cada figura, el panel superior representa la evolución temporal de la señal de error $\mathbf{e}(t)$, el cuadro intermedio muestra la salida del control en términos de par de actitud y, finalmente, el gráfico inferior representa la evolución temporal de las señales de comando enviadas a los motores.

La figura 7.5 corresponde a una entrada oscilatoria en ϕ. En este caso, debido a la configuración de los motores escogida, los motores 1 y 2 se encuentran en oposición de fase con los motores 3 y 4. La figura 7.6 representa la salida del control para una entrada oscilatoria en θ. Para corregir perturbaciones en el ángulo de cabeceo, los motores 1 y 4 trabajan en oposición de fase a los motores 2 y 3. Finalmente, la figura 7.7 muestra la evolución de las señales para una entrada oscilatoria en guiñada, ψ. Para esta entrada, los motores 1 y 3 responden de forma conjunta y en oposición de fase a los motores 2 y 4.

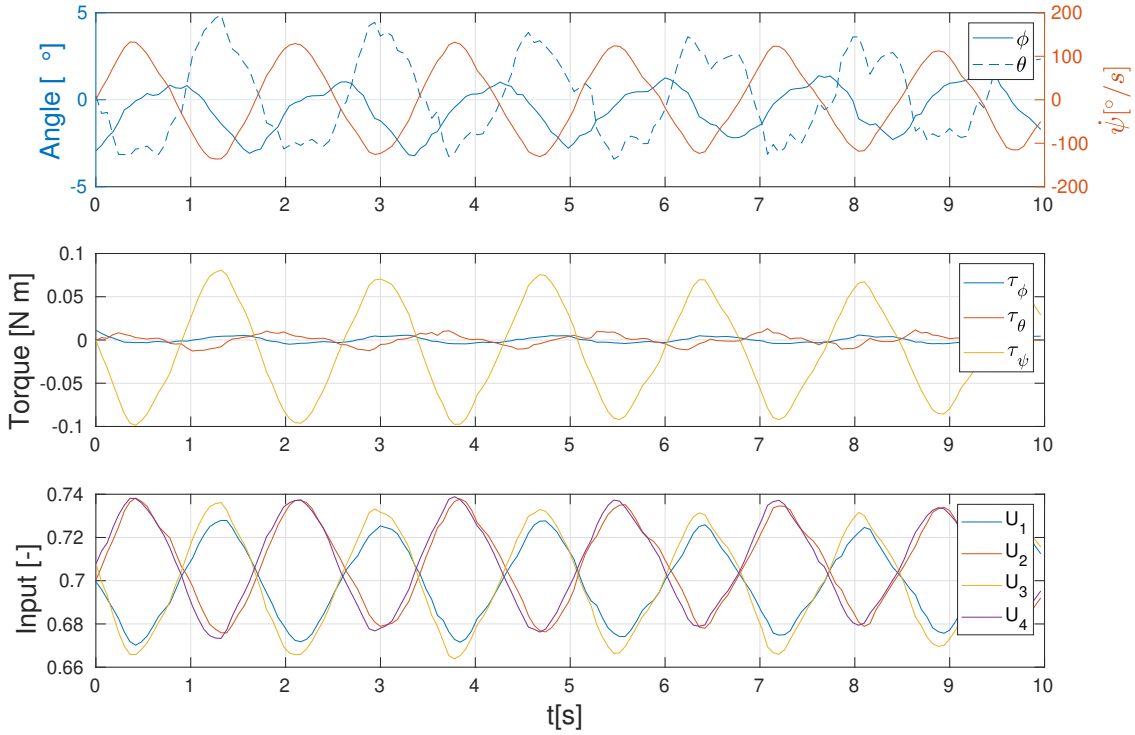

Figura 7.7: Respuesta del control a una entrada oscilatoria en ψ

La numeración de los motores sigue un criterio arbitrario por lo que no es relevante en sí misma. Sin embargo, sí es crucial comprobar que existe una correspondencia entre el diseño y el prototipo real a la hora de evaluar el algoritmo de control en un dispositivo real. Un aspecto que si que tiene consecuencias directas en la salida es la configuración. En la configuración en $\times$, los motores responden por pares, mientras que en la configuración en $+$, solo un par de motores que responden de forma opuesta es responsable del equilibrio en cada eje.

En general, los resultados presentados en 7.5, 7.6 y 7.7 muestran que las salidas del algoritmo de control están dentro de unos rangos razonables y presentan unos niveles de ruido aceptables. Todo ello permite concluir la fase de pre-diseño del control de forma exitosa. El siguiente paso consistiría en implementar este algoritmo de control en una planta real y reajustar los parámetros de los PIDs para corregir las desviaciones por errores de modelización.

Capítulo 8

Conclusiones y trabajo futuro

El presente trabajo representa la fase de desarrollo preliminar de un vehículo autónomo basado en el microcontrolador Raspberry pi Pico. El trabajo se orienta hacia los aspectos prácticos de la sensorización y, en particular, en la estimación de la actitud del vehículo a partir de las lecturas de una unidad de medida inercial. Los principales hitos del trabajo son,

1. Montaje de un circuito prototipo utilizando dos microcontroladores Raspberry pi Pico y una IMU MPU9250 y puesta a punto del entorno de trabajo para el *bootloading* del microcontrolador. La Raspberry pi Pico es menos *user friendly* que otros microcontroladores tales como Arduino y requiere un mayor esfuerzo inicial para desarrollar un entorno de trabajo que permita compilar el código, cargarlo en el controlador y acceder a la salida del programa de manera eficiente.

2. Análisis detallado de las fuentes de error en una IMU de nueve ejes (giróscopo, acelerómetro y magnetómetro) y calibración de los diferentes sensores. Aplicación de una calibración 3D estática para el acelerómetro y de una calibración 3D dinámica para el magnetómetro. Desarrollo de una librería de C/C++ para la Raspberry pi Pico que permite acceder a los registros de memoria del sensor MPU9250 y aplicar la calibración a las lecturas de los sensores.

3. Implementación de un filtro de Kalman extendido para la estimación de la actitud utilizando las lecturas de la IMU. Desarrollo de una librería de C/C++ para la Raspberry pi Pico que ejecuta el algoritmo de fusión de datos a partir de las lecturas de la IMU. La librería del filtro de Kalman requiere de una biblioteca suplementaria – también desarrollada en el contexto del presente trabajo – para la realización de operaciones con matrices y vectores.

4. Desarrollo de un algoritmo de estimación de actitud que combina mediante un filtro complementario la actitud estimada por el algoritmo TRIAD con la integración de

la velocidad angular del sistema. El resultado es un filtro con prestaciones comparables al filtro de Kalman extendido pero que involucra muchas menos operaciones por iteración. Para la implementación de este algoritmo de fusión de datos se ha desarrollado una librería de C/C++ para la Raspberry pi Pico.

5. Desarrollo preliminar de un controlador desacoplado para el control de actitud en un quadrotor. Para ello, se ha construido un modelo simple de la dinámica de la actitud del drone, se ha linealizado el modelo en torno a una condición de vuelo a punto fijo y se ha ajustado un controlador PID para cada uno de los bucles de control resultantes. Se ha analizado la estabilidad del controlador en lazo cerrado, su respuesta en frecuencia y su respuesta temporal. Finalmente, se ha evaluado la respuesta del controlador a señales de entrada generadas por la planta real utilizando el dispositivo experimental construido.

El trabajo sienta las bases para el futuro desarrollo de un vehículo autónomo basado en el controlador Raspberry pi Pico y presenta un nuevo método alternativo de estimación de la actitud para microcontroladores con baja capacidad de procesado. De cara a una posible continuación, se proponen las siguientes líneas de trabajo futuro,

1. Construcción de un dispositivo experimental que permita una validación cuantitativa y precisa de los algoritmos de estimación de actitud. El dispositivo debe construirse a base de encoders angulares y debe permitir la medición simultánea de los tres ángulos de Euler. En una primera fase, puede construirse un montaje pasivo que se limite a medir la orientación del sistema. A continuación, podrían añadirse al sistema motores paso a paso para generar oscilaciones de los ángulos de Euler de amplitud y frecuencia controladas. Esto permitiría caracterizar la respuesta frecuencial del algoritmo de estimación de actitud mediante una *Describing function* [63].

2. Implementación de otras versiones del filtro de Kalman tales como el filtro de Kalman borroso [64]. Evaluación y comparación de sus prestaciones con los algoritmos de estimación de actitud ya desarrollados.

3. Abordar otros aspectos de la sensorización del drone tales como los sensores de altitud, la medición de la velocidad de avance y la detección de obstáculos. Con respecto a la medición de la altura y la velocidad vertical, los sensores basados en ultrasonidos son adecuados cuando el drone se encuentra cerca del suelo y resultan útiles en operaciones de despegue y aterrizaje. Por otro lado, los sensores barométricos están mejor adaptados para grandes altitudes y en condiciones de vuelo libre. Para la medición de la velocidad de avance pueden utilizarse cámaras de baja resolución y algoritmos de flujo óptico. No obstante, el procesado de imágenes requiere una gran capacidad de cálculo, lo que representará un reto para microcontroladores de bajo coste como la Raspberry pi Pico.

Bibliografía

[1] Mohamed Nadir Boukoberine, Zhibin Zhou y Mohamed Benbouzid. "A critical review on unmanned aerial vehicles power supply and energy management: Solutions, strategies, and prospects". En: *Applied Energy* 255 (2019), pág. 113823.

[2] Georgi Valentinov HrISToV, Plamen Zlatkov ZAHArIEV e Ivan Hristov BELoEV. "A review of the characteristics of modern unmanned aerial vehicles". En: *Acta technologica agriculturae* 19.2 (2016), págs. 33-38.

[3] Michał Mazur, Adam Wisniewski, Jeffery McMillan et al. "Clarity from above: PwC global report on the commercial applications of drone technology". En: *Warsaw: Drone Powered Solutions, PriceWater house Coopers* (2016).

[4] Pifu Zhang et al. "Navigation with IMU/GPS/digital compass with unscented Kalman filter". En: *IEEE International Conference Mechatronics and Automation, 2005.* Vol. 3. IEEE. 2005, págs. 1497-1502.

[5] Martin Pettersson. *Extended kalman filter for robust uav attitude estimation.* 2015.

[6] Nikolas Trawny y Stergios I Roumeliotis. "Indirect Kalman filter for 3D attitude estimation". En: *University of Minnesota, Dept. of Comp. Sci. & Eng., Tech. Rep* 2 (2005), pág. 2005.

[7] Mohinder S Grewal, Angus P Andrews y Chris G Bartone. "Kalman filtering". En: (2020).

[8] Charles K Chui, Guanrong Chen et al. *Kalman filtering.* Springer, 2017.

[9] Rudolph Emil Kalman. "A new approach to linear filtering and prediction problems". En: (1960).

[10] François Auger et al. "Industrial applications of the Kalman filter: A review". En: *IEEE Transactions on Industrial Electronics* 60.12 (2013), págs. 5458-5471.

[11] SY Chen. "Kalman filter for robot vision: a survey". En: *IEEE Transactions on industrial electronics* 59.11 (2011), págs. 4409-4420.

[12] Mohinder S Grewal y Angus P Andrews. *Kalman filtering: Theory and Practice with MATLAB.* John Wiley & Sons, 2014.

[13] N Emer y N Özbek. "A survey on Kalman filtering for unmanned aerial vehicles: Recent trends, applications, and challenges". En: *Proceedings of the International Conference on Engineering Technologies (ICENTE'20), Konya, Turkey*. 2020, págs. 19-21.

[14] Walter T Higgins. "A comparison of complementary and Kalman filtering". En: *IEEE Transactions on Aerospace and Electronic Systems* 3 (1975), págs. 321-325.

[15] Mark Euston et al. "A complementary filter for attitude estimation of a fixed-wing UAV". En: *2008 IEEE/RSJ international conference on intelligent robots and systems*. IEEE. 2008, págs. 340-345.

[16] Pengfei Gui, Liqiong Tang y Subhas Mukhopadhyay. "MEMS based IMU for tilting measurement: Comparison of complementary and kalman filter based data fusion". En: *2015 IEEE 10th conference on Industrial Electronics and Applications (ICIEA)*. IEEE. 2015, págs. 2004-2009.

[17] Peter Marwedel. *Embedded system design: embedded systems foundations of cyber-physical systems, and the internet of things*. Springer Nature, 2021.

[18] Dimosthenis E Bolanakis. "A survey of research in microcontroller education". En: *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje* 14.2 (2019), págs. 50-57.

[19] Hari Kishan Kondaveeti et al. "A systematic literature review on prototyping with Arduino: Applications, challenges, advantages, and limitations". En: *Computer Science Review* 40 (2021), pág. 100364.

[20] Eben Upton y Gareth Halfacree. *Raspberry Pi user guide*. John Wiley & Sons, 2016.

[21] Wojciech Kunikowski et al. "An overview of ATmega AVR microcontrollers used in scientific research and industrial applications". En: *Pomiary Automatyka Robotyka* 19.1 (2015), págs. 15-19.

[22] Jon Mullen, Sean CC Bailey y Jesse B Hoagg. "Filtered dynamic inversion for altitude control of fixed-wing unmanned air vehicles". En: *Aerospace Science and Technology* 54 (2016), págs. 241-252.

[23] Raspberry. "Raspberry pi Pico datasheet". En: *Copyright ©2020-2022 Raspberry Pi Ltd* (2022).

[24] Raspberry. "Raspberry pi Pico Python SDK: A MicroPython environment for RP2040 microcontrollers". En: *Copyright ©2020-2022 Raspberry Pi Ltd* (2022).

[25] Raspberry. "Getting started with Raspberry pi Pico: C/C++ development with Raspberry pi Pico and other RP2040-based microcontroller boards". En: *Copyright ©2020-2022 Raspberry Pi Ltd* (2022).

[26] Raspberry. "Raspberry pi Pico C/C++ SDK: Libraries and tools for C++ development on RP2040 microcontrollers". En: *Copyright ©2020-2022 Raspberry Pi Ltd* (2022).

[27] InvenSense. "MPU 9250 Product Specification". En: *PS-MPU-9250A-01* (2014).

[28] Dominique Paret y Carl Fenger. *The I2C bus: from theory to practice*. John Wiley & Sons, Inc., 1997.

[29] Lorenz Meier, Dominik Honegger y Marc Pollefeys. "PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms". En: *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2015, págs. 6235-6240.

[30] ArduPilot Dev Team. *Ardupilot Project*. 2021.

[31] Wojciech Giernacki et al. "Crazyflie 2.0 quadrotor as a platform for research and education in robotics and control engineering". En: *2017 22nd International Conference on Methods and Models in Automation and Robotics (MMAR)*. IEEE. 2017, págs. 37-42.

[32] Pascal Brisset et al. "The paparazzi solution". En: *MAV 2006, 2nd US-European competition and workshop on micro air vehicles*. 2006, pp-xxxx.

[33] Robert Bogue. "MEMS sensors: past, present and future". En: *Sensor Review* (2007).

[34] Stephen Beeby. *MEMS mechanical sensors*. Artech House, 2004.

[35] Yuanxin Wu y Ling Pei. "Gyroscope calibration via magnetometer". En: *IEEE Sensors Journal* 17.16 (2017), págs. 5269-5275.

[36] Haotian Yang et al. "A novel tri-axial MEMS gyroscope calibration method over a full temperature range". En: *Sensors* 18.9 (2018), pág. 3004.

[37] Kian Sek Tee et al. "Triaxial accelerometer static calibration". En: *Proceedings of the World Congress on Engineering*. Vol. 3. 2011, págs. 25-27.

[38] Isaac Skog y Peter Händel. "Calibration of a MEMS inertial measurement unit". En: *XVII IMEKO world congress*. 2006, págs. 1-6.

[39] Valérie Renaudin, Muhammad Haris Afzal y Gérard Lachapelle. "Complete triaxis magnetometer calibration in the magnetic domain". En: *Journal of sensors* 2010 (2010).

[40] PP Freitas et al. "Magnetoresistive sensors". En: *Journal of Physics: Condensed Matter* 19.16 (2007), pág. 165221.

[41] Edward Ramsden. *Hall-effect sensors: theory and application*. Elsevier, 2011.

[42] Manon Kok et al. "Calibration of a magnetometer in combination with inertial sensors". En: *2012 15th International Conference on Information Fusion*. IEEE. 2012, págs. 787-793.

[43] Thom Magnusson. *State estimation of uav using extended kalman filter*. 2013.

[44] Qingde Li y John G Griffiths. "Least squares ellipsoid specific fitting". En: *Geometric modeling and processing, 2004. proceedings*. IEEE. 2004, págs. 335-340.

[45] Andrew Fitzgibbon, Maurizio Pilu y Robert B Fisher. "Direct least square fitting of ellipses". En: *IEEE Transactions on pattern analysis and machine intelligence* 21.5 (1999), págs. 476-480.

[46] Chi-Lun Cheng y John W Van Ness. "Statistical regression with measurement error". En: *(No Title)* (1999).

[47] Ivan Markovsky, Alexander Kukush y S Van Huffel. "Consistent least squares fitting of ellipsoids". En: *Numerische Mathematik* 98 (2004), págs. 177-194.

[48] Michael J Caruso. "Applications of magnetoresistive sensors in navigation systems". En: *Progress in technology* 72 (1998), págs. 159-168.

[49] Edward Allen. *Modeling with Itô stochastic differential equations*. Vol. 22. Springer Science & Business Media, 2007.

[50] Bernt Oksendal. *Stochastic differential equations: an introduction with applications*. Springer Science & Business Media, 2013.

[51] Alex Becker. "Kalman Filter". En: *Dipetik Juli* 1 (2018), pág. 2020.

[52] Sebastian Madgwick et al. "An efficient orientation filter for inertial and inertial/-magnetic sensor arrays". En: *Report x-io and University of Bristol (UK)* 25 (2010), págs. 113-118.

[53] Harold D Black. "A passive system for determining the attitude of a satellite". En: *AIAA journal* 2.7 (1964), págs. 1350-1351.

[54] Itzhack Y Bar-Itzhack y Richard R Harman. "Optimized TRIAD algorithm for attitude determination". En: *Journal of guidance, control, and dynamics* 20.1 (1997), págs. 208-211.

[55] Sergei Tanygin y Malcolm D Shuster. "The many triad algorithms". En: *Adv. Astronaut. Sci* 127 (2007), págs. 81-99.

[56] Samir Bouabdallah, Pierpaolo Murrieri y Roland Siegwart. "Design and control of an indoor micro quadrotor". En: *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*. Vol. 5. IEEE. 2004, págs. 4393-4398.

[57] Randal W Beard. "Quadrotor dynamics and control". En: *Brigham Young University* 19.3 (2008), págs. 46-56.

[58] Jun Li y Yuntang Li. "Dynamic analysis and PID control for a quadrotor". En: *2011 IEEE International Conference on Mechatronics and Automation*. IEEE. 2011, págs. 573-578.

[59] Taeyoung Lee, Melvin Leok y N Harris McClamroch. "Geometric tracking control of a quadrotor UAV on SE (3)". En: *49th IEEE conference on decision and control (CDC)*. IEEE. 2010, págs. 5420-5425.

[60] Amir Hussein y Rayyan Abdallah. "Autopilot Design for a Quadcopter". En: *Khartoum: University Of Khartoum* (2017).

[61] Abbas Emami-Naeini y Robert L Kosut. "The generalized Nyquist criterion and robustness margins with applications". En: *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*. IEEE. 2012, págs. 226-231.

[62] Pedro Albertos y Sala Antonio. *Multivariable control systems: an engineering approach*. Springer Science & Business Media, 2006.

[63] Wallace E Vander Velde et al. "Multiple-input describing functions and nonlinear system design". En: *McGraw lill* (1968).

[64] R Senthil, K Janarthanan y J Prakash. "Nonlinear state estimation using fuzzy Kalman filter". En: *Industrial & engineering chemistry research* 45.25 (2006), págs. 8678-8688.

[65] Hector Garcia De Marina et al. "UAV attitude estimation using unscented Kalman filter and TRIAD". En: *IEEE Transactions on Industrial Electronics* 59.11 (2011), págs. 4465-4474.

[66] Ern J Lefferts, F Landis Markley y Malcolm D Shuster. "Kalman filtering for spacecraft attitude estimation". En: *Journal of Guidance, Control, and Dynamics* 5.5 (1982), págs. 417-429.