UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

DISEÑO DE UNA PLATAFORMA SOFTWARE PARA EL CONTROL DE ORIENTACIÓN 3D DE UN SISTEMA QUADROTOR BASADO EN RASPBERRY PI 2

AUTOR: RUI PEDRO MACHADO DA SILVA

TUTOR: PEDRO JOSÉ GARCÍA GIL

Curso Académico: 2015-16

"A mis compañeros de laboratorio, por haberme ayudado a hacer posible este trabajo

A mi familia, por la paciencia y apoyo mostrados durante la realización de este trabajo

A mi tutor, por darme la oportunidad de demostrar de lo que soy capaz"

RESUMEN

El objetivo del presente trabajo es el desarrollo de una plataforma software para el control de orientación y altura de un quadrotor. Dicho software debe presentar la suficiente versatilidad para el desarrollo de futuros trabajos.

El sistema quadrotor de estudio consta de un mini ordenador Raspberry Pi, encargado de ejecutar los algoritmos de control, que permiten un vuelo estable y robusto. Además, incluye dos sensores, una Unidad de Medida Inercial (IMU) y un barómetro, que permiten conocer la orientación espacial del quadrotor y su altura respecto al plano del suelo. Por último, cuenta con los motores, con sus correspondientes drivers, que serán los encargados de realizar las actuaciones pertinentes.

Para dotar al de una buena respuesta dinámica se ha implementado un Sistema Operativo de Tiempo Real (SOTR), que garantiza el cumplimiento de unos estrictos tiempos de ejecución y un respeto por las prioridades de las tareas.

Los elementos electrónicos se han conectado debidamente, permitiendo el flujo de información entre los distintos componentes, siguiendo unos protocolos de comunicación. Una vez realizadas las conexiones eléctricas, se han colocado sobre la estructura del quadrotor.

Por último, a través de un ordenador en tierra, se ha hecho uso del Interfaz Hombre-Máquina (HMI), capaz de comunicarse con el quadrotor de forma inalámbrica y de modificar los parámetros de control durante el vuelo, se han realizado una serie de ensayos para validar el software desarrollado. Dichos ensayos se han efectuado con el quadrotor acoplado a una plataforma de ensayo, que permite el movimiento de cuatro grados de libertad (los tres ángulos de orientación y el desplazamiento vertical).

Este trabajo forma parte de un proyecto conjunto, donde otro compañero ha desarrollado el quadrotor físico y una plataforma de testeo de vuelo, que permita la implementación del sistema de control expuesto anteriormente.

Palabras clave: Quadrotor, Raspberry Pi, IMU, SOTR

DOCUMENTOS CONTENIDOS EN ESTE TFG

- Memoria
- Presupuesto

ÍNDICE DE LA MEMORIA

1 INTRODUCCIÓN 2

1.1 Objetivos 2
1.2 Estructura del Documento 3

2 ESTADO DEL ARTE 4

2.1 Reseña Histórica 4
2.2 Contexto Actual y Proyección Futura 5
2.3 Motivaciones 6

3 FUNDAMENTOS DE UN QUADROTOR 8

3.1 Descripción de un Quadrotor 8
3.1.1 Definición 8
3.1.2 Elementos básicos 8
3.2 Modelo teórico de un Quadrotor 10
3.2.1 Modelo matemático 10
3.2.2 Control en cruz 11
3.2.3 Control en equis 12
3.2.4 Incertidumbre del modelo 13

4 SOPORTE HARDWARE 14

4.1 Justificación 14
4.2 Raspberry Pi 14
4.2.1 Raspberry Pi 2 Model B 14
4.2.2 Raspberry to Arduino Shield Connection Bridge 15
4.2.3 Realtek Wireless USB Adapter 16
4.3 Sensor IMU 10DOF V2 16
4.3.1 IMU MPU6050 17
4.3.2 Barómetro MS5611 18
4.4 Unidad Motriz 19
4.4.1 Motores Yellow Trainer 1500 19
4.4.2 Drivers Turnigy Basic 25A 20
4.5 Elementos Adicionales 21
4.5.1 Quadrotor 21
4.5.2 Plataforma de ensayo 21

4.5.3 Interfaz Hombre-Máquina 22

5 ASPECTOS GENERALES DEL SOFTWARE 23

5.1 Comunicación 23

5.1.1 Aspectos generales 23
5.1.2 Comunicación I²C 24
5.1.3 Comunicación PWM 25
5.1.4 Comunicación UDP 26

5.2 Sistema Operativo 27

5.2.1 Necesidad de un sistema operativo de tiempo real 27
5.2.2 Características de un SOTR 27
5.2.3 Solución adoptada 30

5.3 Lenguaje de Programación 31

5.3.1 Criterios de selección del lenguaje de programación 31
5.3.2 Características de la programación orientada por objetos 32
5.3.3 Ventajas respecto a otros paradigmas de programación 34
5.3.4 Solución adoptada 35

6 DESCRIPCIÓN DE LA FUNCIONALIDAD 36

6.1 Estructura del Programa 36

6.1.1 Hilos de ejecución 36
6.1.2 Jerarquía de objetos 37

6.2 Lectura de Sensores 38

6.2.1 Lecturas de la IMU 38
6.2.2 Lecturas del barómetro 40
6.2.3 Gestión del bus 41

6.3 Procesado de señal 42

6.3.1 Necesidad de procesar las mediciones 42
6.3.2 Fundamentos del filtro de Kalman 42
6.3.3 Corrección de la altura 44
6.3.4 Cálculo del pitch, roll y yaw 46

6.4 Algoritmo de Control 48

6.4.1 Fundamentos de un PID 48
6.4.2 Determinación de los parámetros de control 49
6.4.3 Control de orientación y altura 50

6.5 Actuación sobre los motores 50

6.5.1 Saturación de la acción de control 50
6.5.2 Generación de la señal PWM 51

6.6 Control desde Tierra 52

7 RESULTADOS EXPERIMENTALES 53

7.1 Introducción 53

7.2 Medida de Altura 53

7.3 Control del Pitch 54

7.4 Control del Roll 55
7.5 Control del Yaw 56

8 CONCLUSIONES Y TRABAJOS FUTUROS 58

8.1 Conclusiones 58
8.2 Trabajos Futuros 59

BIBLIOGRAFÍA Y REFERENCIAS 60

ÍNDICE DEL PRESUPUESTO

1 Introducción 2

2 Cuadro de Precios Descompuestos 3

2.1 Capítulo 1: Diseño 3
2.2 Capítulo 2: Construcción 3
2.3 Capítulo 3: Software 6
2.4 Capítulo 4: Ensayos 6

3 Presupuesto Parcial 6

4 Presupuesto de Ejecución Material, Presupuesto por Contrata y Presupuesto de Base de Licitación 7

ÍNDICE DE ILUSTRACIONES

Ilustración 1: Kattering Bug, en el Museo Nacional de las Fuerzas Armadas de Estados Unidos 4
Ilustración 2: MQ-1 Predator 5
Ilustración 3: DJI Phantom 4 6
Ilustración 4: Plataforma de estudio de diámicas de vuelo 3DOF Hover desarrollada por Quanser 7
Ilustración 5: IGEP V2, ordenador compacto usado en anteriores proyectos 9
Ilustración 6: Sensor óptico de posición Hokuyo URG-04LX 10
Ilustración 7: Ángulo de orientación del plano del quadrotor 10
Ilustración 8: Control en cruz 12
Ilustración 9: Control en equis 13
Ilustración 10: Raspberry Pi 2 Model B 15
Ilustración 11: Raspberry to Arduino Shield Connection Bridge 15
Ilustración 12: Realtek Wireless USB Adapter 16
Ilustración 13: IMU 10DOF V2 sobre una placa compatible 16
Ilustración 14: Conexión de la IMU a la Raspberry 17
Ilustración 15: IvenSense MPU 6050 18

Ilustración 16: MEAS MS5611 18
Ilustración 17: Cuatro motores Yellow Trainer 1500 19
Ilustración 18: 4 drivers Turnigy Basic 25A 20
Ilustración 19: Conexión de los drivers a la Raspberry 20
Ilustración 20: Quadrotor con (izquierda) y sin estructura de seguridad (derecha) 21
Ilustración 21: Antigua plataforma de ensayo (izquierda). Nueva plataforma desarrollada este año (derecha) 22
Ilustración 22: Interfaz Hombre Máquina 22
Ilustración 23: Esquema de comunicación 23
Ilustración 24: Selección de dispositivo y registro en I2C 24
Ilustración 25: Escritura de un byte en I2C 24
Ilustración 26: Lectura de un byte en I2C 25
Ilustración 27: Diferentes señales PWM para una frecuencia fija 26
Ilustración 28: Estados posibles de una tarea 28
Ilustración 29: Funcionamiento de un semáforo binario 29
Ilustración 30: Funcionamiento de un mutex 30
Ilustración 31: Comparativa de tiempos de ejecución con y sin parche de tiempo real 31
Ilustración 32: Comparativa de rendimientos con y sin parche de tiempo real 31
Ilustración 33: Estructura de encapsulamiento de un objeto 33
Ilustración 34: Ejemplo esquematizado de herencia de clases 34
Ilustración 35: Flujo de información entre los hilos del programa 37
Ilustración 36: Esquema jerárquico de los objetos del programa 38
Ilustración 37: Función de transferencia del giróscopo 40
Ilustración 38: Bucle de control PID 48
Ilustración 39: Estructura interna de un controlador PID 48
Ilustración 40: Acción de los motores en función de la acción de control 51
Ilustración 41: Función de trasferencia del pulso PWM 52
Ilustración 42: Medida de la altura 53
Ilustración 43: Respuesta del pitch ante cambio de referencia 54
Ilustración 44: Respuesta del pitch ante perturbación 54
Ilustración 45: Respuesta del roll ante cambio de referencia 55
Ilustración 46: Respuesta del roll ante perturbación 56
Ilustración 47: Respuesta del yaw ante cambios de referencia 57
Ilustración 48: Respuesta del yaw ante perturbación 57

DISEÑO DE UNA PLATAFORMA SOFTWARE PARA EL CONTROL DE ORIENTACIÓN 3D DE UN SISTEMAQUADROTOR BASADO EN RASPBERRY PI 2

MEMORIA

1 INTRODUCCIÓN

El presente documento constituye la memoria de un Trabajo de Final de Grado de la titulación Grado en Tecnologías Industriales, cursado en la Escuela Técnica Superior de Ingenieros Industriales, en la Universidad Politécnica de Valencia desde el año 2012 hasta el año 2016.

1.1 Objetivos

El objetivo del presente trabajo es el desarrollo de una plataforma software para el control de orientación y altura de un quadrotor. Este trabajo forma parte de un proyecto conjunto, donde otro compañero ha desarrollado el quadrotor físico y una plataforma de testeo de vuelo, que permita la validación del sistema de control expuesto.

Para lograr este objetivo global, se propone una serie de objetivos más específicos, que son los siguientes:

- Implementar un sistema operativo de tiempo real que permita la ejecución simultanea de varias tareas con prioridad definida y dote a al sistema de una buena respuesta y garantice la estabilidad del vehículo en todo momento.

- Proveer al conjunto del sistema de la capacidad de conocer su posición y orientación en el espacio en todo momento

- Desarrollar un del programa de control de orientación y altura que tenga una estructura modular del código, para posibilitar la creación de varios hilos independientes y facilitar la ampliación y mejoras futuras. El programa se enfocará para ser usado en el mencionado quadrotor, sobre la plataforma de pruebas.

- Integrar al máximo las funcionalidades del sistema en el menor número de elementos posibles para simplificar las comunicaciones y reducir los retardos y las fuentes de error.

- Estudiar la capacidad de programa para realizar el control del ángulo yaw y de la altura dentro del marco de sensores disponibles.

- Implementar el software desarrollado en el quadrotor en cuestión y analizar la aptitud del algoritmo de control.

- Analizar la bondad del mini ordenador Raspberry Pi 2 para realizar el control de orientación de un quadrotor, en base al desempeño obtenido du.

1.2 Estructura del Documento

A continuación se enumeran los capítulos que constituyen este documento y se describen brevemente los aspectos más importantes

1. Introducción

Se detallan los objetivos del trabajo y las circunstancias que han motivado el desarrollo de este trabajo, aparte de analizar la situación de los RPAS en la actualidad.

2. Fundamentos de un quadrotor

Estudio del quadrotor como un elemento teórico, las características más destacables, los componentes básicos que debe tener y el modelo matemático que rige su comportamiento

3. Soporte hardware

Exposición de los elementos que constituyen el quadrotor que se ha usado como referencia para el desarrollo del software.

4. Aspectos generales del software

Desglose de los puntos y criterios fundamentales que debe tener el software para garantizar el buen funcionamiento del mismo y las soluciones adoptadas para la satisfacción de los mismos.

5. Descripción de la funcionalidad

Descripción detallada de la estructura del programa y de los elementos que lo componen. Se explica la funcionalidad de cada uno de ellos y como se relacionan entre sí.

6. Resultados experimentales

Análisis de los resultados obtenidos en la puesta a prueba del quadrotor y software desarrollados.

7. Conclusiones y trabajos futuros

Valoración de los resultados obtenidos, análisis de la consecución de los objetivos planteado inicialmente y propuestas de mejora para futuros trabajos.

Se puede observar que el documento tiene dos partes diferenciadas: en los capítulos 2 y 3 se plantea el quadrotor como una entidad teórica y analiza los elementos que lo componen. Por otro lado, entre los capítulos 4 y 6, se desarrolla la solución software para el control del mismo y se experimentalmente la bondad del mismo.

2 ESTADO DEL ARTE

2.1 Reseña Histórica

La primera plataforma aérea no tripulada de la que se tiene constancia data de 1849, cuando el ejército austríaco usó globos cargados de explosivos, que liberaban a distancia gracias a un cable de cobre. Si bien no encaja en el concepto actual de UAV, es el primer precedente en este campo.

No sería hasta después de la Primera Guerra Mundial cuando se inicia el desarrollo de vehículos aéreos no tripulados (*Unmanned Aerial Vehicle*, UAV), cuando las primeras aeronaves, motorizadas, sin piloto como aviones blanco, para la práctica de tiro de armas antiaéreas y pilotos de combate, pero sin ningún otro objetivo más que ser destruidos por soldados en formación [1]. Más adelante, durante la Segunda Guerra Mundial y el auge de la tecnología de la Alemania Nazi, los ingenieros alemanes desarrollaron diversas armas guiadas por radiofrecuencia, también destinadas a su propia destrucción al impactar en el objetivo. Esta sería la tendencia que seguirían las dos grandes potencias, Estados Unidos y la Unión Soviética, al finalizar la guerra y la caída del dominio nazi. La amenaza de la bomba nuclear potencio durante los primeros años de la guerra fría el desarrollo de misiles balísticos y cabezas nucleares, dejando de lado el desarrollo de aeronaves no tripuladas.

Ilustración 1: Kattering Bug, en el Museo Nacional de las Fuerzas Armadas de Estados Unidos

El retorno de los UAV al plano de desarrollo no se produciría hasta varias décadas después, cuando a ambos lados del telón de acero empezaron a usar las aeronaves como aviones espía y de reconocimiento, dejando atrás la época donde su único objetivo era su propia destrucción. Esto permitió una notable ventaja sobre terrenos en conflicto, sobre todo para los estadounidenses, y acabarían por imponerse los sistemas UAV de reconocimiento sobre los satélites, en los años 80. En la última década del siglo aparece el mítico MQ-1 Predator, recuperando la

faceta ofensiva, pero esta vez gracias a sistema armamentístico incluido que no implica la destrucción de la aeronave [2].

Ilustración 2: MQ-1 Predator

2.2 Contexto Actual y Proyección Futura

Al igual que muchos otros avances tecnológicos, los RPAS (*Remotely Piloted Aircraft Systems*) tuvieron su fase de desarrollo en la industria para más tarde sufrir una adaptación al ámbito civil, como el nacimiento de la computación moderna a manos de Alan Turing con el propósito de descifrar la máquina de encriptado alemana *Enigma*, el desarrollo del radar para localizar vehículos enemigos o la investigación y estudio de los antibiótico, para paliar las infecciones en el campo de batalla [3].

Hoy en día vivimos la década de las aplicaciones civiles de los UAV (es difícil hablar de una era en un contexto donde los avances tecnológicos y sociales se producen tan rápidamente). Su versatilidad y la no necesidad de un piloto a bordo han permitido la proliferación en multitud de campos, desde la cartografía hasta la salvación marítima. Aunque la aplicación más extendida es para usos recreativos; existen cientos de drones (terminología popular de UAV) en el mercado a precios de usuario. Esta propagación en la sociedad abre las puertas de un nuevo debate: la seguridad y el control sobre los RPAS.

Ser propietario de un RPAS no requiere experiencia alguna en pilotaje y, si bien la mayoría de los que se comercializan son de bajo alcance, existen algunos modelos de gran tamaño y potencia. Esto pone en peligro la seguridad ciudadana y de algunas instituciones, como ya se ha visto en algunos incidentes ocurridos recientemente: aeronaves desconocidas sobrevolando centrales nucleares o dificultando el aterrizaje de aviones comerciales. Esto ha obligado a los países a iniciar trámites de normativas para regularizar el uso de RPAS. Como su proliferación se ha producido de una forma tan repentina, muchos de estos reglamentos son de carácter provisional o se encuentran todavía en fases de desarrollo, la propensión es hacia la prohibición de estos en zonas públicas.

Ilustración 3: DJI Phantom 4

Estas limitaciones de vuelo no impiden que su difusión por todos los ámbitos se detenga. Se prevé que dentro de diez años los RPAS acaparen el 10% del mercado aeronáutico mundial [4]. Esto no hace más que afianzar la necesidad de seguir mejorando las funcionalidades y la seguridad de estos, con el fin de garantizar que los vehículos aéreos no pilotados tengan un impacto beneficioso en la sociedad en la que vivimos.

2.3 Motivaciones

A la vista del potencial y la versatilidad que ofrecen los RPAS, tanto para aplicaciones civiles e industriales, tener la posibilidad de participar en un proyecto de diseño, desarrollo e implementación de uno resulta una oportunidad única. Permite un contacto más profundo con un campo en continua evolución y de creciente demanda, lo que puede suponer una ventaja futura en el mercado laboral.

Además, se presenta como un reto multidisciplinar donde se tendrá la posibilidad de aplicar conceptos asociados a diversas áreas de la ingeniería: electricidad, aerodinámica, electrónica, informática, control… lo que supone un reto y permitirá afianzar los conocimientos adquiridos durante estos últimos cuatro años.

Hay dos grandes aspectos tecnológicos que motivan este proyecto. Primero, aportar una prueba más de la capacidad de los mini ordenadores Raspberry de realizar el control de un quadrotor de forma autónoma y de las posibilidades que puede llegar a ofrecer este potente miniordenador

Por otro lado, el intento de desarrollar una plataforma basada en el Hover 3DOF de Quanser de bajo presupuesto, que pueda ser usada en un futuro con propósitos académicos y facilitar el estudio y desarrollo de otros quadrotores.

Ilustración 4: Plataforma de estudio de dinámicas de vuelo 3DOF Hover desarrollada por Quanser

3 FUNDAMENTOS DE UN QUADROTOR

3.1 Descripción de un Quadrotor

3.1.1 Definición

Un quadrotor, también denominado cuadricóptero, es un helicóptero con cuatro rotores, los cuales sirven para su propia sustentación y para la propulsión. A diferencia de las aeronaves de ala fija, Se suelen colocar los rotores, como en este caso, en una cruz simétrica. Los movimientos, los giros y la elevación se consiguen variando la velocidad de giro de los motores, a fin de cambiar el par motor que generan.

En la definición de quadrotor no se especifica cómo o quién realiza el control, tan solo la forma de sustentación, por lo que pueden ser tanto pilotados como controlados remotamente. No obstante, no existen desarrollos actuales de cuadricópteros pilotados por un motivo: el control de los tres giros se realiza variando el régimen de los motores individualmente, lo cual supone un control más sencillo que en helicópteros convencionales de dos rotores. Sin embargo, esto solo resulta factible en modelos de pequeñas dimensiones, con motores de corriente continua pequeños, mientras que para realizar un vehículo a gran escala capaz de transportar al piloto supondría el uso de motores adicionales o largas cadenas de transmisión, lo que aumentaría el peso global.

Por tanto, en ausencia de una tecnología que haga posibles modelos de mayor tamaño, capaces de transportar a una persona, el quadrotor queda relegado al plano de los UAV.

3.1.2 Elementos básicos

Los componentes varían de un modelo a otro, pero todos constan de unos básicos, los cuales son imprescindibles para el funcionamiento de la aeronave:

- Motores: en el caso de un quadrotor, son necesarios cuatro, con sus respectivas hélices. Cada motor tiene que tener la posibilidad de ser controlado de forma independiente a los demás. Para que sea posible la estabilidad, deben girar en sentido opuesto a pares y con las hélices apropiadas para generar un impulso ascendente.

- Sensor de orientación: ha de ser posible conocer la orientación de la aeronave en todo momento. La unidad de medida inercial aporta las velocidades angulares y las aceleraciones lineales en los tres ejes, que pueden ser usadas para calcular los ángulos de orientación

- Controlador: capaz de procesar las medidas de orientación y determinar el régimen de cada motor, con el fin de garantizar la estabilidad, por lo que debe contar con la suficiente potencia de cálculo. Puede trabajar de forma completamente autónoma o recibiendo ordenes externas.

Ilustración 5: IGEP V2, ordenador compacto usado en anteriores proyectos

A parte de los componentes básicos que garantizan el funcionamiento de RPAS, existen otros elementos también importantes pero que no resultan fundamentales para la estabilidad del quadrotor:

- Seguridad: el quadrotor debe contar con los elementos suficientes que garanticen la seguridad de las personas y su propia integridad. Pueden ser elementos de seguridad activos, que evitan los accidentes, como un detector de obstáculos choques o un medidor de la carga de la batería que avisa cuando se alcanza un cierto límite inferior; o pasivos, que palian los efectos cuando se produce un fallo, como pueden ser la estructura externa del quadrotor o incluso un paracaídas que se despliega cuando este entra en caída.

- Control de posición: sensores capaces de determinar la posición en los tres ejes. El control x-y se puede realizar con GPS de forma separada al control de altura (z), que se puede determinar con un barómetro (grandes alturas) o un ultrasonidos (rango pequeño de funcionamiento), aunque existen sensores ópticos capaces de medir los tres al mismo tiempo.

Por último, existen componentes no tan relevantes para el control pero que también presentan utilidad en el vuelo del RPAS:

- Cámara: permiten observar el vuelo desde el punto de vista del quadrotor, ya sea en tiempo real o a posteriori

- Luces: pueden cumplir una función tanto estética como de seguridad, permitiendo vislumbrar la aeronave en situaciones de baja visibilidad.

- Carga: dotar al quadrotor de la capacidad de transportar objetos, de pequeño y mediano tamaño permite usarlo para realizar entregas en zonas de difícil

Ilustración 6: Sensor óptico de posición Hokuyo URG-04LX

3.2 Modelo Teórico de un Quadrotor

3.2.1 Modelo matemático

Existen seis variables independientes que permiten determinar el quadrotor en el espacio: tres para posicionar su centro de masas y tres para determinar la orientación (conocidos como los ángulos de Euler, son pitch, roll y yaw).

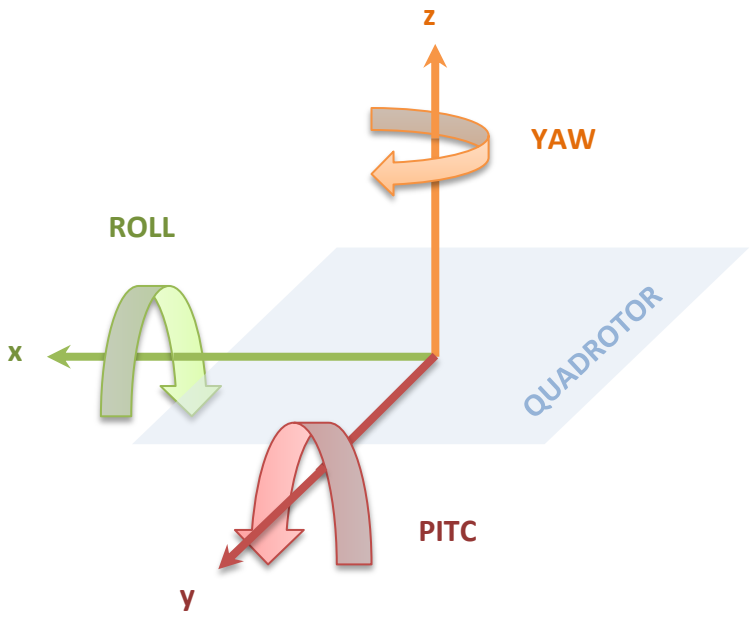

Ilustración 7: Ángulo de orientación del plano del quadrotor

Suponiendo el plano del quadrotor horizontal, el pitch y el roll supondrían los dos ángulos de inclinación del plano respectivamente (cabeceo y alabeo) y el de rotación (deriva), como se observa en la Ilustración 7. Las ecuaciones que rigen el movimiento del quadrotor (ecuaciones 3.1 – 3.6) se deducen a través de la física clásica o mediante el enfoque de Euler-LaGrange, pero cuyo desarrollo queda fuera del alcance de este trabajo [5].

$$m \cdot \ddot{x} = F \cdot (-\sin(\theta) \cdot \cos(\phi) \cdot \cos(\psi) - sen(\phi) \cdot \cos(\psi)) \quad (3.1)$$

$$m \cdot \ddot{y} = F \cdot (-\sin(\theta) \cdot \cos(\phi) \cdot \cos(\psi) + sen(\phi) \cdot \cos(\psi)) \quad (3.2)$$

$$m \cdot \ddot{z} = F \cdot cos(\theta) \cdot cos(\phi) \quad (3.3)$$

$$J_{xx} \cdot \ddot{\phi} = \tau_{\phi} \quad (3.4)$$

$$J_{yy} \cdot \ddot{\theta} = \tau_{\theta} \quad (3.5)$$

$$J_{zz} \cdot \ddot{\psi} = \tau_{\psi} \quad (3.6)$$

Siendo φ, θ y ψ el roll, pitch y yaw respectivamente; z la distancia del plano del quadrotor al plano del suelo; x-y las coordenadas del centro de masas de quadrotor sobre el plano del mismo; y J el momento de inercia del respectivo eje. F representa el empuje total, suma de los empujes de cada motor (ecuación 3.7), y τ los pares en cada uno de los ángulos

$$F = F_1 + F_2 + F_3 + F_4 \quad (3.7)$$

El par en el eje del yaw se produce cuando la suma de los pares de giro de los motores es distinta de cero, suponiendo los motores enfrentados con el mismo sentido de giro (ecuación 3.8).

$$\tau_{\psi} = \tau_1 - \tau_2 + \tau_3 - \tau_4 \quad (3.8)$$

Para poder determinar cómo se distribuyen los pares del pitch y el roll entre los motores, primero se debe determinar qué tipo de control se efectuará sobre el quadrotor.

3.2.2 Control en cruz

En este formato de control, los ejes del quadrotor coinciden con la alineación de los rotores. De esta forma, el control de un cierto ángulo se realiza con dos motores enfrentados. Los pares resultantes pueden verse en las ecuaciones 3.9 y 3.10, dada la distribución de motores de la Ilustración 8

$$\tau_{\phi} = (F_2 - F_4) \cdot l_b \quad (3.9)$$

$$\tau_{\theta} = (F_3 - F_1) \cdot l_b \quad (3.10)$$

Donde l_b es la longitud de los brazos de los motores. Este tipo de control es poco habitual, pues solo dos de los motores trabajan sobre cada ángulo.

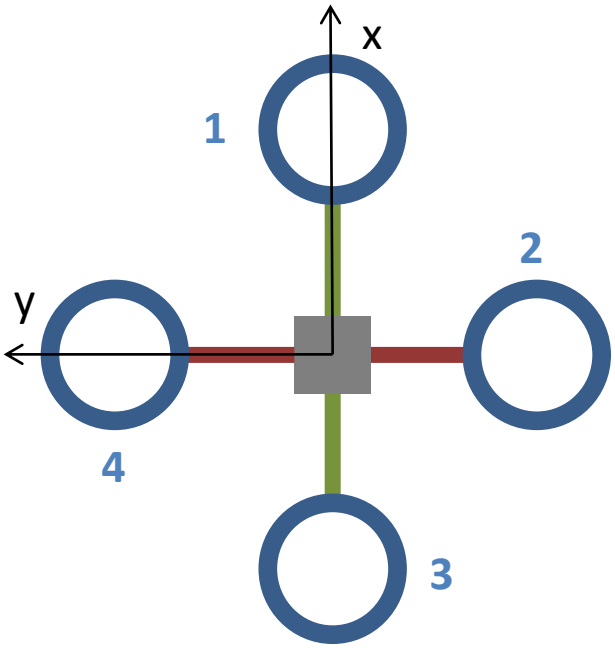

Ilustración 8: Control en cruz

3.2.3 Control en equis

Los ejes de referencia se sitúan a 45º de los ejes de los motores, dejando a ambos lados un motor de cada sentido de giro, permitiendo un control más estable. Este sistema obliga a los cuatro motores a actuar sobre ambos giros, repartiendo así las fuerzas necesarias y evitando que saturen.

Lo pares aplicados en función de los empujes aplicados por cada motor se pueden ver en las ecuaciones 3.11 y 3.12.

$$\tau_\phi = (-F_1 - F_2 + F_3 + F_4) \cdot {l_b}/{\sqrt{2}} \quad (3.11)$$

$$\tau_\theta = (-F_1 + F_2 + F_3 - F_4) \cdot {l_b}/{\sqrt{2}} \quad (3.12)$$

Como se puede observar, la distancia de aplicación de la fuerza se reduce, y por consecuente el par que produce, lo que permite un control más preciso de los ángulos. Esta configuración es ampliamente utilizada, dejando el control en cruz en un plano teórico.

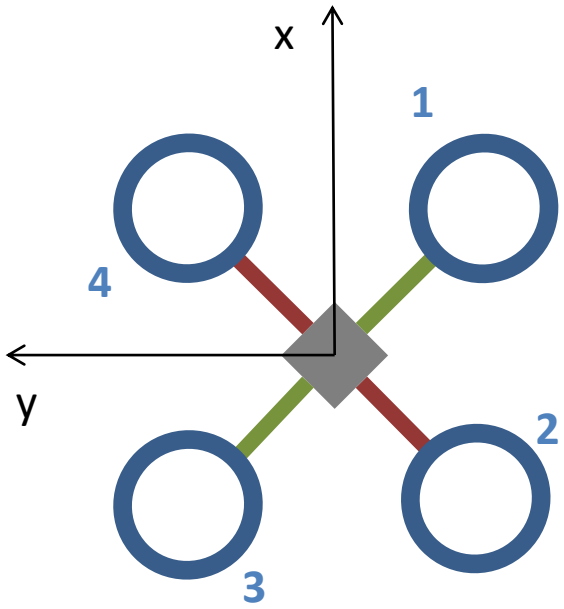

Ilustración 9: Control en equis

3.2.4 Incertidumbre del modelo

El modelo planteado en los puntos anteriores se trata de un modelo matemático simplificado que, si bien en la teoría puede servir para comprender el funcionamiento y comportamiento de un quadrotor, en la práctica resulta poco útil. Esto se debe principalmente a dos factores: el desconocimiento de los parámetros y la ausencia de variables que afectan la conducta.

Tal y como se puede apreciar en las ecuaciones 3.1-3.12, la dinámica del quadrotor depende principalmente de los empujes de los motores. Pero este empuje no se controla directamente, sino que se hace a través de la tensión de los motores, lo que obliga a conocer la relación entre estos, que posiblemente será no lineal. Además, también es necesario determinar los momentos de inercia del quadrotor, cosa que en ningún momento será fácil de determinar.

A todo esto, hay que sumar la ausencia en el modelo de la aerodinámica. En ningún momento se tiene en cuenta efectos externos al propio quadrotor, como viento u otras perturbaciones, que si se darán en un vuelo real.

Por tanto, usar el modelo matemático para realizar el control no es factible, si bien se puede usar como base o punto de partida para obtener una aproximación del comportamiento real. Esto obliga a pasar por una fase experimental, lo cual ratifica la necesidad de un banco de pruebas.

4 SOPORTE HARDWARE

4.1 Justificación

Si bien el objetivo es la creación de un software de control, es necesario tener un RPAS asociado. Esto no significa que dicho quadrotor deba existir físicamente, tan solo saber que partes lo componen para desarrollar un sistema de control acorde a sus características.

Por tanto, es importante determinar tres cosas fundamentalmente: qué tipo de controlador u ordenador ejecutará el algoritmo, a fin de saber el desempeño que se le puede exigir y conocer las peculiaridades de su programación (librerías necesarias, tipos de comunicación incluidos); sensores de los que dispone, para poder conectarlos y realizar la lectura correctamente y además saber que grados de libertad nos permitirán controlar; y por último, los motores (y sus correspondientes drivers), para conocer el rango de funcionamiento y su respuesta en carga.

4.2 Raspberry Pi

4.2.1 Raspberry Pi 2 Model B

Raspberry Pi 2 es la segunda versión de un mini ordenador creado por la Fundación Raspberry Pi, con el fin de hacer más accesibles los ordenadores en el ámbito de la enseñanza. Por ese motivo, se caracteriza por su bajo precio y unas prestaciones aceptables que le confieren la mejor relación calidad/precio del mercado en esta gama de productos [6].

El reducido coste de la Raspberry Pi (35$) ha permitido su proliferación en el mundo de la informática y actualmente cuenta con una comunidad que facilita el desarrollo de nuevos proyectos. Esto no hace más que afianzar su superioridad respecto a sus rivales y decantar la balanza a la hora de escoger en el momento de la adquisición.

El modelo usado consta de cuatro puertos USB, un puerto Ethernet y salida HDMI, por lo que se podría conectar un monitor, un ratón y un teclado y trabajar como un ordenador convencional. Su procesador de cuatro núcleos y un 1GB de memoria RAM le confieren mucha más potencia que la versión anterior. Además, cuenta con pines de comunicación I^2C y serie, a parte de varios puertos de propósito general (GPIO).

Desafortunadamente, este modelo (a diferencia de versiones anteriores), no cuenta con un puerto de salida PWM, lo que supone tener que generar la señal por software o incluir algún dispositivo de soporte que realice la generación de pulsos PWM.

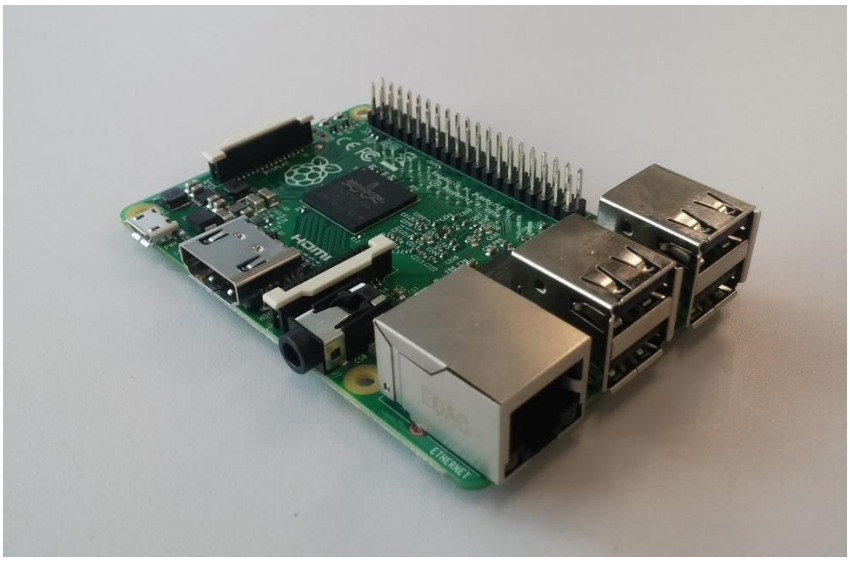

Ilustración 10: Raspberry Pi 2 Model B

4.2.2 Raspberry to Arduino Shield Connection Bridge

Placa diseñada para permitir la conexión con Raspberry de módulos diseñados para Arduino. Incluye además varios conversores analógico-digital que periten la conexión de varios sensores.

En este proyecto no se ha usado la conectividad con Arduino porque no había necesidad de ello. No obstante, la conexión de sensores analógicos puede ser interesante para trabajos futuros. El principal propósito de usar este *shield* es para proteger la Raspberry y evitar durante la fase de desarrollo realizar constantes conexiones y desconexiones sobre los pines de la Raspberry.

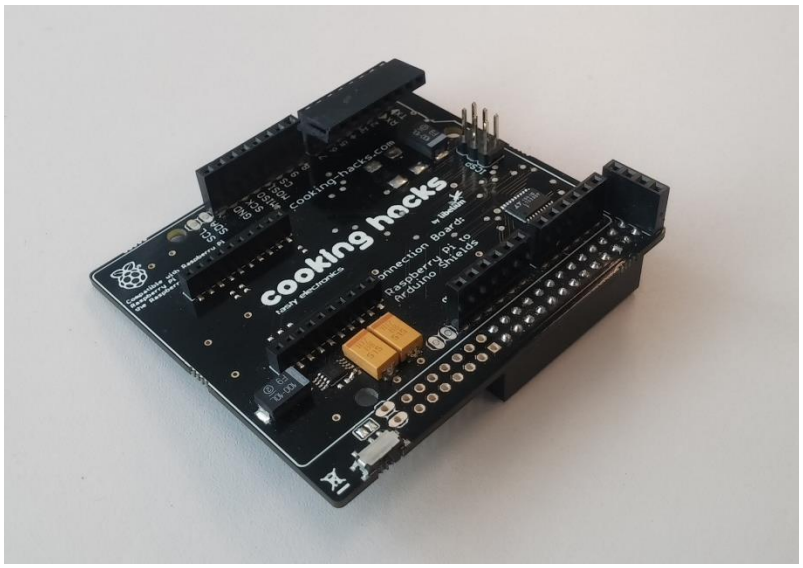

Ilustración 11: Raspberry to Arduino Shield Connection Bridge

4.2.3 Realtek Wireless USB Adapter

Pequeño adaptador que permite a la Raspberry realizar conexiones Wifi, que, por defecto, no están disponibles en la misma. A través de este puerto se podrán realizar conexiones inalámbricas con otros ordenadores y comunicar con el controlador

Ilustración 12: Realtek Wireless USB Adapter

4.3 Sensor IMU 10DOF V2

El quadrotor consta únicamente de un único sensor, la IMU 1DOF V2 de Drotek. Dicho sensor se compone en realidad de tres sensores diferentes: Una unidad de medida inercial (MPU6050), un barómetro (MS5611) y un magnetómetro (HMC5883), que le confieren esos 10 grados de libertad [7]. En este trabajo solo se usarán los dos primeros, pero pudiendo incluir en un futuro el último

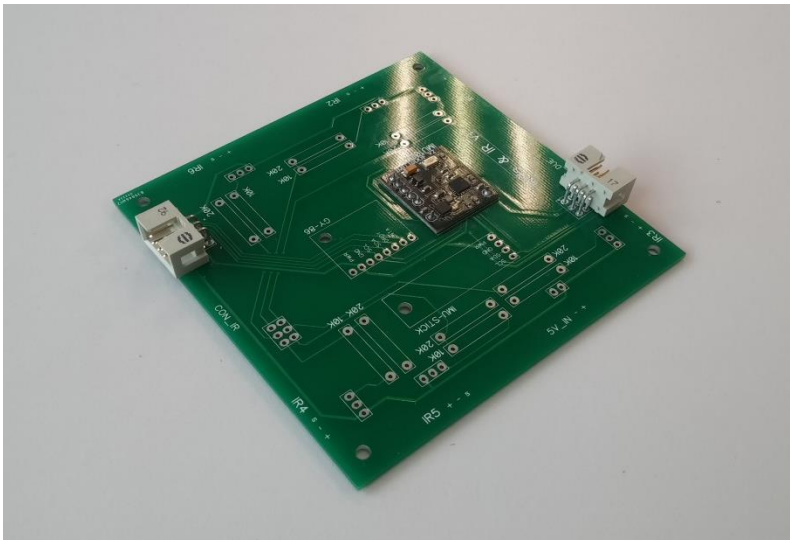

Ilustración 13: IMU 10DOF V2 sobre una placa compatible

Este sensor presente una buena respuesta y precisión a un precio asequible (18,90€). Además, su disponibilidad en el laboratorio y los buenos resultados ofrecidos en proyectos anteriores acabaron por determinarla como la mejor opción.

La lectura y escritura de los registros de estos sensores se realiza mediante el protocolo de comunicación I^2C o, alternativamente, por SPI.

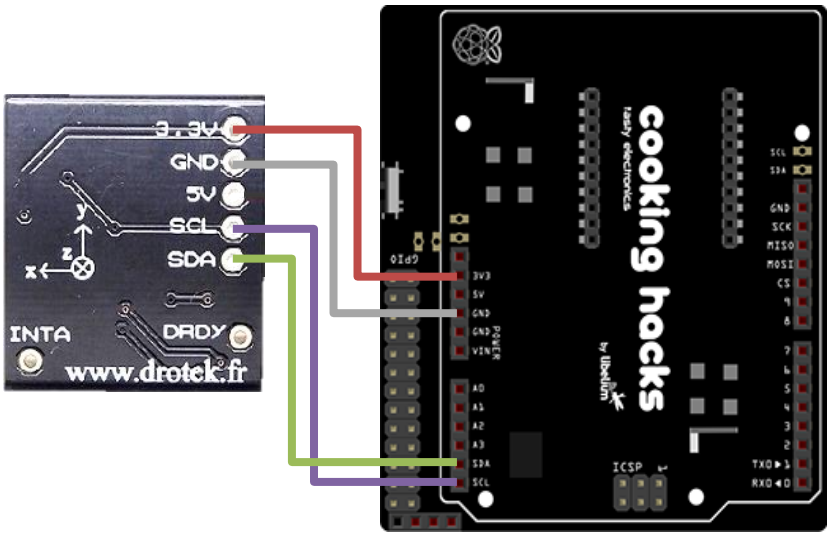

Ilustración 14: Conexión de la IMU a la Raspberry

4.3.1 IMU MPU6050

Contiene un giróscopo y un acelerómetro, que permite obtener las velocidades angulares y las aceleraciones lineales en los tres ejes.

Cuenta con un pequeño procesador que interpreta los valores obtenidos directamente de los seis sensores y los procesa aplicando algoritmos de fusión. También permite configurar aspectos del funcionamiento del mismo, tales como la frecuencia de muestreo o los rangos de medida.

Adicionalmente, permite conectar a través de un bus auxiliar de I^2C a un magnetómetro y, actuando como maestro-esclavo, obtener las medidas de este directamente, sin tener que acceder por otro canal.

Permite operar a distintas frecuencias (valores comprendidos entre 125 y 1000Hz), y ajustar la precisión de la medida, a costa de reducir el rango [8].

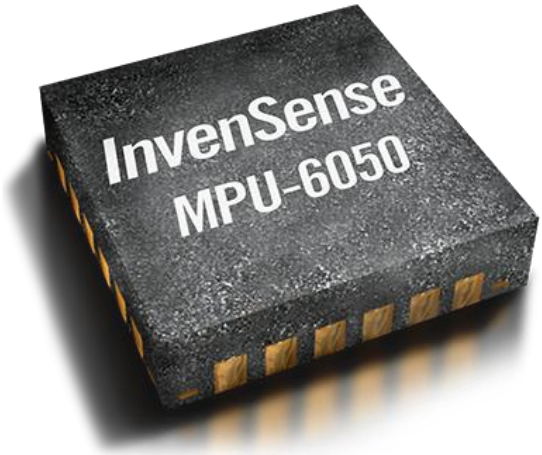

Ilustración 15: IvenSense MPU 6050

4.3.2 Barómetro MS5611

Altímetro barométrico de alta precisión, permite obtener la presión y la temperatura local. Con los apropiados cálculos y conocida la presión a nivel del mar, se puede obtener la altitud absoluta.

A diferencia de la IMU, no cuenta con un procesador interno, por lo que la configuración y la lectura de medidas se hacen de una forma más directa, sin uso de registros. No obstante, la comunicación se sigue realizando mediante I^2C

El tiempo de muestreo (tiempo en el que tarde en realizar una medida y hacer la conversión) varía entre 1 y 10 ms, en función de la precisión deseada. La medición presenta un problema: al no haber registros de almacenamiento, tan solo se puede medir una variable por ciclo (temperatura o presión), lo que reduce a la mitad la frecuencia real de ejecución [9].

Ilustración 16: MEAS MS5611

4.4 Unidad Motriz

Los únicos actuadores en un quadrotor son sus motores. No obstante, estos por si solos no son capaces de hacer funcionar la aeronave; necesitan hélices capaces de generar el empuje (aunque para el desarrollo que aquí se busca no son relevantes) y drivers, capaces de entregar la tensión correspondiente a los motores dada una especificación de régimen de funcionamiento.

4.4.1 Motores Yellow Trainer 1500

Los motores son el único actuador presente en el quadrotor y, junto con las hélices, son los encargados de producir la sustentación del mismo. Es importante que los cuatro motores sean iguales, para garantizar la estabilidad de la aeronave

Estos motores llegan a generar 180W cada uno y son capaces de sustentar entre los cuatro aeronaves de más de dos kilos, lo que los hace más que suficientes para el quadrotor desarrollado. Tienen una eficiencia máxima del 78% y producen 1400 kV (1400rpm por cada voltio suministrado).

Su corriente de trabajo se sitúa entre los 7 y los 12 amperios (zona de máxima eficiencia), no obstante pueden llegar a solicitar 16 amperios en ciertas ocasiones (arranque), lo que obliga a tener unos dispositivos de alimentación capaces de soportar estas solicitaciones [10].

Ilustración 17: Cuatro motores Yellow Trainer 1500

4.4.2 Drivers Turnigy Basic 25A

Los drivers tienen dos entradas: una de alimentación, de 11 voltios, y otra de control de PWM. Ajustando el ancho de pulso de la señal de PWM entre unos valores predeterminados, los drivers entregan una corriente entre 7 y 11 voltios trifásicos a la salida.

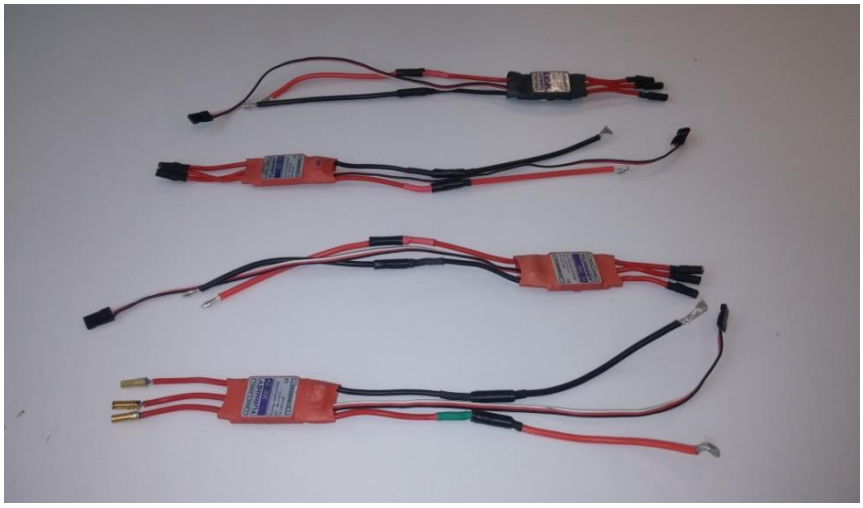

Ilustración 18: 4 drivers Turnigy Basic 25A

El ancho de pulso que requieren los drivers es indiferente del periodo de dicha señal, y suele situarse entre 1000 µs de valor mínimo y 2000 µs de valor máximo.

Los drivers usados son tipo BEC (permiten alimentar el receptor de radio control, si lo hay, sin necesidad de baterías externas) y con una intensidad base de 25A (la mínima recomendada para los motores usados es de 20A), pero son capaces de soportar hasta 28 amperios [11]. Tras la configuración, su rango de lectura de señales PWM es de 1145 a 1800µs, con una frecuencia de 100Hz.

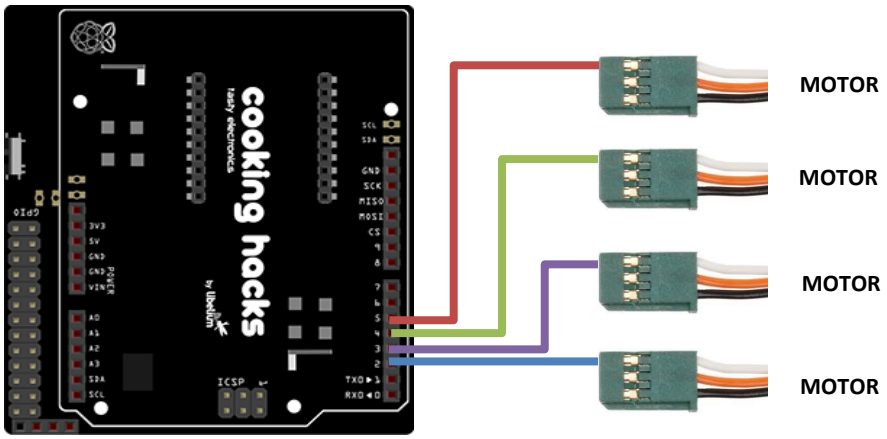

Ilustración 19: Conexión de los drivers a la Raspberry

4.5 Elementos Adicionales

Aunque el diseño y desarrollo de los elementos que a continuación se describen queda fuera del alcance de este trabajo, constituyen una parte fundamental para el diseño y validación del software de control

4.5.1 Quadrotor

Concepción física del quadrotor de referencia tomado a lo largo del desarrollo. Su diseño y construcción son fruto del trabajo de fin de grado de otro compañero, Javier Reina.

El quadrotor incluye todos los componentes anteriormente comentados, a parte de otros en los que no se ha entrado en detalle porque no condicionan el funcionamiento del programa pero que resultan relevantes para el correcto funcionamiento del conjunto.

Ilustración 20: Quadrotor con (izquierda) y sin estructura de seguridad (derecha)

4.5.2 Plataforma de ensayo

También fabricada por Javier, la plataforma permite el acoplamiento del quadrotor para estudiar su desempeño de forma segura. El soporte permite a la aeronave enlazada cuatro grados de libertad: los tres ángulos de orientación (pitch, roll y yaw), gracias a una rótula mecánica, y el desplazamiento vertical, mediante un tubo telescópico, el cual puede ser bloqueado si se desea.

Esto supone una ventaja respecto a la plataforma desarrollada el año pasado que solo permitía el giro pitch y roll (no simultáneos), pues permite ensayar el quadrotor en unas condiciones más

No obstante, esta mayor libertad de movimiento también supone una desventaja al mismo tiempo, obliga a controlar los tres giros al mismo tiempo, lo que dificulta el sintonizado y la búsqueda de errores.

Ilustración 21: Antigua plataforma de ensayo (izquierda). Nueva plataforma desarrollada este año (derecha)

4.5.3 Interfaz Hombre-Máquina

Programa desarrollado por otro compañero en un trabajo anterior, permite el control desde tierra del quadrotor. Conectados de forma inalámbrica controlador y ordenador de tierra, se pueden modificar los parámetros de control, referencias y otros aspectos durante el vuelo.

Esto obliga a tener en cuenta dentro del programa está conectividad y la posibilidad de recibir estos paquetes de información.

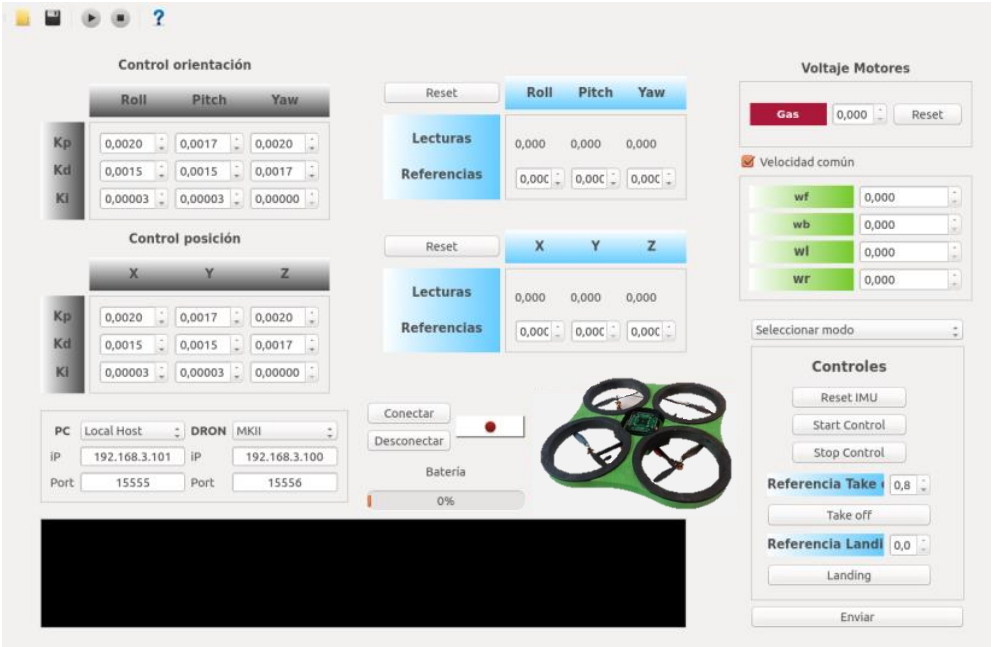

Ilustración 22: Interfaz Hombre Máquina

5 ASPECTOS GENERALES DEL SOFTWARE

5.1 Comunicación

5.1.1 Aspectos generales

Como se ha visto en el capítulo anterior (ver 4. SOPORTE HARDWARE), el quadrotor se compone de diversas partes, las cuales desempeñan funciones específicas pero que requieres estar comunicadas.

Esto torna la comunicación un aspecto crítico a tener en cuenta en el desarrollo del programa y debe tener en cuenta las peculiaridades de cada uno de ellos

El esquema general de comunicación es el siguiente:

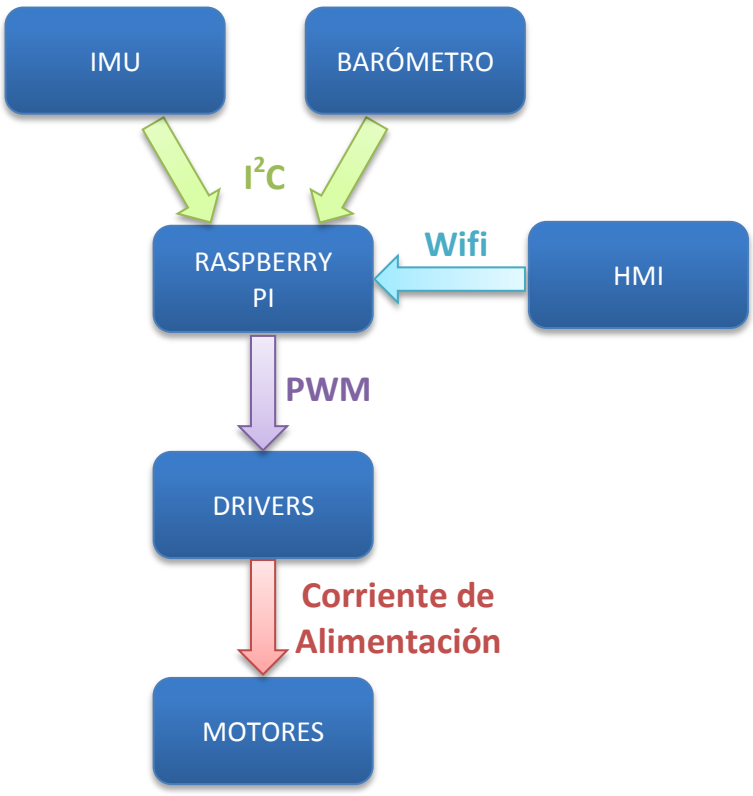

Ilustración 23: Esquema de comunicación

La última etapa no se trata de una fase de comunicación sino de potencia, pues no hay transferencia de información. Sin embargo, se ha representado en este diagrama pues esta conexión estará presente en los esquemas eléctricos y representan la última etapa del proceso de control.

5.1.2 Comunicación I²C

El I^2C (*Inter-integrated circuit*) es un protocolo de comunicación estandarizado para la comunicación entre dispositivos de cierto nivel de inteligencia de forma síncrona. Se basa en un bus serie, con dos cables (SDA y SCL) y en una referencia de masa (GND). Permite una comunicación entre 100 y 400 kbits/s, aunque en ciertos modos (*UltraFast-mode* o *HighSpeed-mode*) logra superar los 3,4 Mbits/s e incluso alcanzar los 5Mbits/s.

Permite la comunicación entre varios dispositivos, donde todos están conectados a la misma red de comunicación, y unos actúan como maestros (tienen la potestad de iniciar la comunicación) y los demás como esclavos (tan solo pueden responder a las solicitudes de comunicación). Cuando el bus está libre (ambos canales a nivel lógico alto, gracias a las resistencias *pull-up*, cualquier dispositivo maestro puede ocuparlo.

El dispositivo maestro inicia la secuencia de inicio. Después, envía la dirección del dispositivo esclavo con el que quiere comunicarse. Si dicha dirección existe en el bus, el esclavo envía la secuencia de confirmación. A continuación, el maestro, de nuevo, envía la dirección del registro que quiere leer o escribir. Si se selecciona escritura, se deberá enviar el dato (o datos) y una vez finalizada dicha transmisión, enviar la secuencia de Stop. Por otro lado, si se selecciona lectura, se envía de nuevo la secuencia de inicio y la de direccionamiento del dispositivo (con un bit extra, el de lectura/escritura, en nivel alto), lo que inicia la transmisión del esclavo del dato almacenado en el registro. Cada secuencia incluye un bit extra de conformación, a nivel bajo si todo está correcto [6].

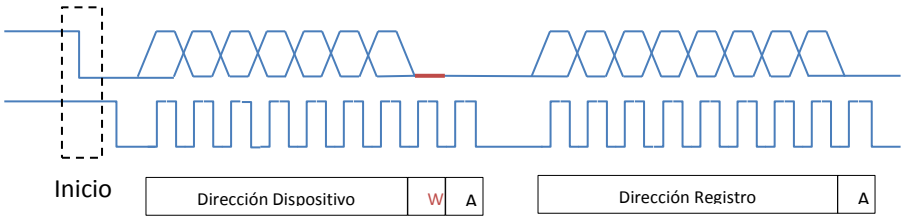

Ilustración 24: Selección de dispositivo y registro en I2C

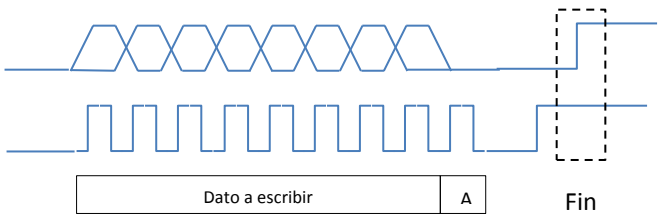

Ilustración 25: Escritura de un byte en I2C

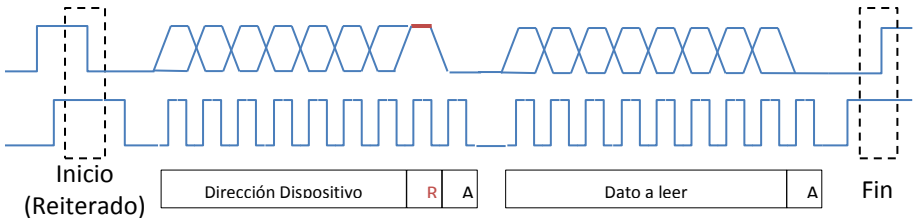

Ilustración 26: Lectura de un byte en I2C

La Raspberry ya incluye la comunicación I^2C, dedicando dos pines exclusivamente a este protocolo. Por tanto, con la versión oficial del software (Raspbian) ya se pueden realizar comunicaciones mediante este bus, sin necesidad de librerías adicionales. Bastará únicamente con habilitar la comunicación I^2C y conocer las direcciones de los esclavos.

5.1.3 Comunicación PWM

La modulación por ancho de pulso (o *Pulse Wave Modulation*, en inglés, dando origen a las siglas PWM), es una señal de onda cuadrada usada para transmitir información o para regular el índice de carga.

Ajustando el ancho del semipulso activo, se logra una señal con un valor eficaz variable. Por ese motivo se suele utilizar en conversores digital-analógicos de poca calidad. También se puede usar para regular el régimen de funcionamiento de un motor de corriente continua, simulando una tensión continua ajustable.

Aunque normalmente no se usa como protocolo de comunicación, en este caso se usan para enviar las referencias a los motores. Variando el ancho de pulso, se ajusta la tensión trifásica que entregan los drivers a los motores. Esto supone un problema, dado que la placa no incluye ningún pin por defecto para señales PWM. Surgen, por tanto, dos alternativas posibles: simulación vía software de dicha señal o uso de un dispositivo adicional que se encargue de la generación.

El uso de una etapa intermedia entre el mini-PC y los drivers permitiría generar las señales deseadas. No obstante, habría retrasos en la comunicación y se incrementarían las fuentes de errores, tal y como ocurría en proyectos anteriores donde se usaba esta etapa. Si bien no eran errores críticos para el funcionamiento ya estabilidad del vehículo, impedían un desempeño óptimo. Por ese motivo, se ha intentado eliminar esta capa.

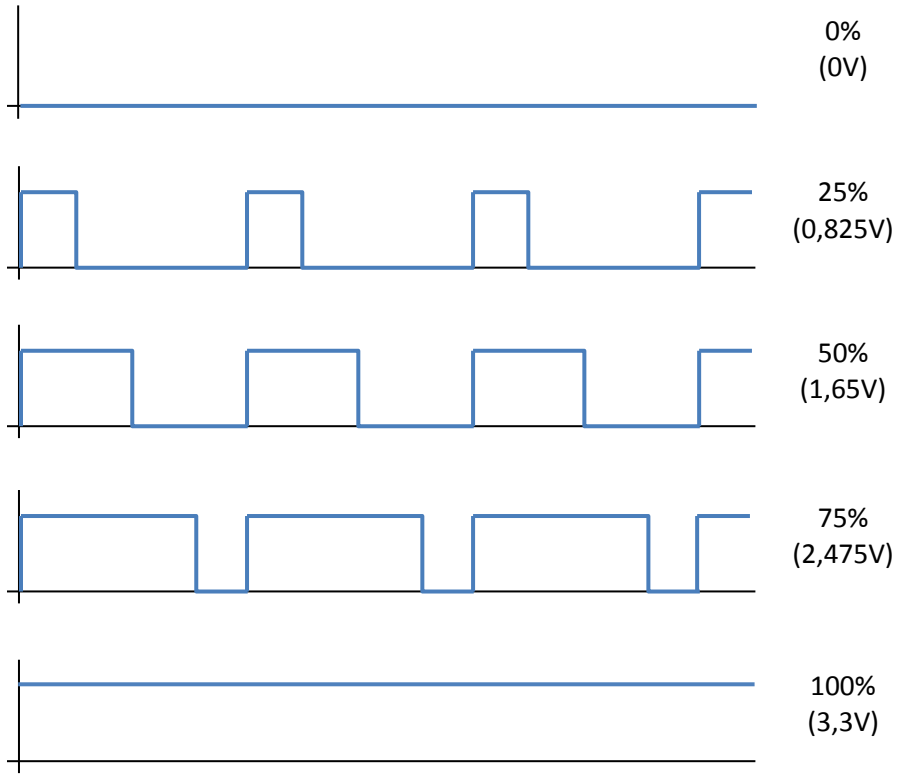

Ilustración 27: Diferentes señales PWM para una frecuencia fija

La otra alternativa es la generación mediante software, lo que supone dos problemas. Primero, la velocidad a la que se pueden generar los pulsos no tiene la suficiente resolución dentro del rango de funcionamiento (el pulso activo debe situarse entre 1 y 2 ms). Además, emular estas ondas supone el uso del procesador, lo que puede perjudicar a la ejecución del hilo de control.

La solución ideal sería generar estas señales sin que suponga gasto de tiempo de ejecución. Esto se puede lograr gracias a unas librerías (PIGPIO), que permiten actuar sobre los GPIO mediante Acceso Directo a Memoria (DMA). De este modo, se ha logrado generar pulsos PWM con buena resolución que permite el control de los cuatro motores

5.1.4 Comunicación UDP

El Protocolo de Datagramas de Usuario (UDP en sus siglas en inglés) en un protocolo de comunicación sin conexión. Por tanto, no se asegura de que existe una conexión antes de iniciar la transmisión. Es por eso que se usa en casos donde prima la velocidad de transferencia sobre la garantía de la transmisión integra de toda la información. Una vez enviado un paquete (o datagrama) en la red, el servidor se olvida y no se certifica de si ha llegado o no, por lo que puede ocurrir que un paquete se pierda o llegue por duplicado, o incluso que se adelanten unos a otros (carece de control de flujo).

En el protocolo existen dos entidades: el servidor (emisor) y el cliente (receptor). El servidor establece la configuración de la transmisión, por lo que el cliente debe saber la IP y el puerto que debe escuchar. Aparte del dato o mensaje (de longitud variable), el paquete incluye un puerto de origen, al cual debe dirigirse si el cliente quiere contestar.

Este tipo de protocolo es muy usado en sistemas donde las conexiones/desconexiones son frecuentes o en el envío de grandes cantidades de paquetes (audio o video), donde los retardos provocados por las constantes verificaciones de conexión producirían considerables retrasos en la comunicación perjudicando la transmisión en tiempo real. También se usa en redes de ordenadores donde las comunicaciones son poco frecuentes y de poca cantidad de paquetes, como en este caso [12].

5.2 Sistema Operativo

5.2.1 Necesidad de un sistema operativo de tiempo real

Un sistema operativo es software encargado de gestionar los recursos hardware del computador. Permite crear una capa de abstracción, un nivel intermedio entre el computador y el usuario. Crea las tareas y determina el tiempo de ejecución (procesador), controla el acceso y almacenamiento de información (memoria) y permite el envío y recepción con los recursos externos (periféricos). Es este primer aspecto, el de creación y gestión de tareas, por lo que resulta necesario un sistema operativo [13].

Un sistema operativo otorga al ordenador de un carácter multitarea. Sin embargo, eso no es suficiente para garantizar el correcto funcionamiento del programa de control. Para poder garantizar la estabilidad del quadrotor se requiere que el sistema tenga una rápida respuesta. Y eso se logra mediante un Sistema Operativo de Tiempo Real.

5.2.2 Características de un SOTR

La característica general de un SOTR es que el tiempo de ejecución de una tarea es predecible (determinístico) y certifica que las operaciones se ejecutan dentro de unos tiempos fijados. De esta manera, se pueden fijar prioridades a cada tarea de manera que, si en un momento dado, hay dos tareas pendientes de ejecución, se dé paso a la de mayor prioridad. Esto asegura que las tareas más importantes se ejecutarán con la frecuencia correspondiente. Si algunas de las tareas que solicitan tiempo de ejecución tienen la misma prioridad, el planificador del sistema operativo irá alternando la ejecución entre unas y otras, simulando una ejecución simultánea [14] .

Las tareas que gestiona el sistema operativo pueden encontrarse en tres estados diferentes:

- Ejecución: la tarea correspondiente está haciendo uso en este momento del procesador. Puede abandonar este estado si finaliza su rutina o si el planificador decide relegarla para dar paso a otra tarea.

- Pendiente: una tarea que está preparada para ser ejecutada y está a la espera que el planificador autorice el acceso al procesador. Se puede acceder a este estado si el planificador activa una tarea bloqueada o si bien alterna el tiempo de ejecución

- Bloqueada: una tarea inactiva, que no requiere ser ejecutada. Una tarea bloqueada se mantendrá en este estado hasta que una señal exterior, normalmente el planificador, la active, en cuyo caso pasará a bloqueada. Normalmente, las tareas se desbloquean con una determinada frecuencia [15].

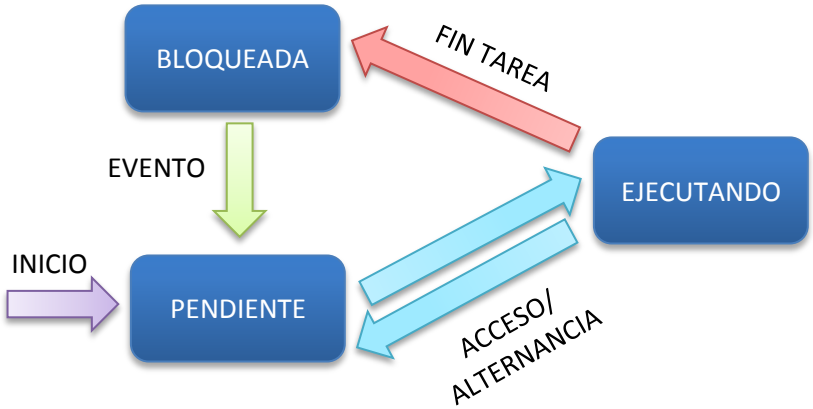

Ilustración 28: Estados posibles de una tarea

Aparte de le gestión del tiempo de ejecución y de la asignación de las respectivas prioridades, aún quedan dos problemas por resolver: la coherencia de información y la secuencia de ejecución. La ejecución simultanea de varias tareas o hilos que comparten información o funcionalidad puede ser crítica: secuencia de tareas que exigen un cierto orden temporal o hilos que acceden o modifican unos mismos datos globales. Para evitar este tipo de problemas, el SOTR integra varias herramientas: los semáforos binarios y los mutex.

Si bien ambos recursos tienen un funcionamiento similar, se usan para propósitos completamente diferentes. La idea que hay detrás es la de una tarea bloquea una parte del código que queda inaccesible para los demás hilos hasta que no se desbloquea. Pero el punto clave para diferenciarlos es quién tiene la "llave" que permite bloquear/desbloquear. Mientras que en los semáforos binarios, cualquier tarea que lo quiera puede desbloquear un semáforo bloqueado, en el caso de los mutex tan solo podrá hacerlo la tarea que lo bloqueó.

Los semáforos se usan para garantizar que una tarea no se ejecuta hasta que otra haya finalizado (total o parcialmente). La tarea que debe ejecutarse en segundo lugar inicia su rutina, pero se encuentra con el semáforo bloqueado, por lo que detiene su ejecución. Cuando el otro hilo finaliza su rutina, desbloquea el semáforo (el cual no había bloqueado), lo que permite a la primera realizar su ejecución y, al finalizar, volver a bloquear el semáforo. Hay por tanto, una tarea que desbloquea y la otra que bloquea con el fin de asegurar una secuencia de ejecución [16].

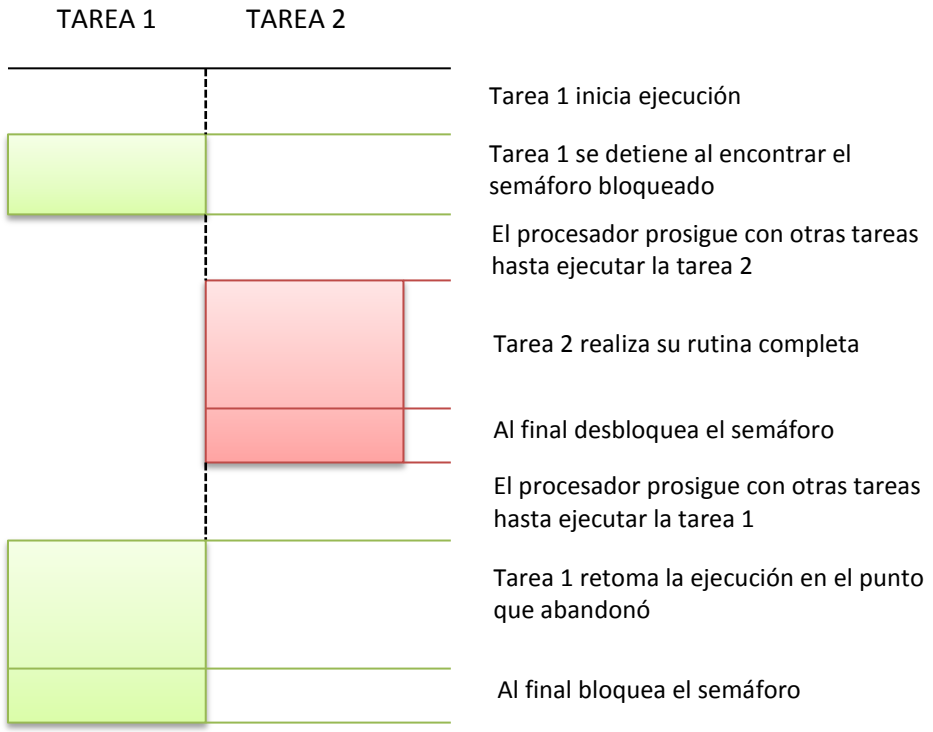

Ilustración 29: Funcionamiento de un semáforo binario

Por otro lado los mutex (*MUTual EXclusion*, de su origen en inglés), controlan se utilizan para controlar el acceso a lectura y escritura por varias tareas a un mismo dato o conjunto de datos. Cuando una tarea inicia el uso del recurso compartido, bloquea el mutex, de tal manera que si su ejecución se detiene (para dar prioridad a una tarea de mayor prioridad), este hilo no podrá hacer uso de este recurso en su rutina. Al retomar la ejecución de la primera y esta finalizar las operaciones de los datos en cuestión, desbloquea el mutex permitiendo el acceso a otras tareas.

Con este recurso programático, se pueden evitar serios problemas de coherencia de datos. El caso más representativo es el de lectura/escritura: un hilo está leyendo unas variables (por ejemplo, el hilo de control lee la orientación actual), cuando un hilo de mayor prioridad solicita ejecución (el hilo de medida de la IMU). Instantemente, el planificador da paso a esta nueva tarea, que escribe en las variables conflictivas, destrozando los valores anteriores. Cuando el primer hilo retoma la ejecución y prosigue con la lectura, almacenará una mezcla de valores anteriores y actuales, lo que puede producir errores [17].

Ilustración 30: Funcionamiento de un mutex

5.2.3 Solución adoptada

Antes esta necesidad de un sistema operativo de tiempo real, se deben buscar alternativas que satisfagan estos criterios. Este problema ya apareció en un trabajo del año pasado, donde se trabajó con la Raspberry Pi 1, se barajaron dos posibles opciones: Xenomai y PreemptRT. Ninguno de los dos constituye un sistema operativo por sí solo, sino que se tratan de parches para el kernel que le otorgan este carácter de tiempo real. El sistema operativo usado es el Raspbian, desarrollado en base Linux, especialmente diseñado para Raspberry.

Xenomai se barajó en un primer momento por haber sido un parche usado en proyectos anteriores. Su desempeño fue satisfactorio y cumplía los tiempos de ejecución. No obstante, perjudicaba el funcionamiento de los puertos, lo que impedía actuar sobre los motores o leer de los sensores. Por tanto, se descartó está opción.

Por otra parte, el parche PreemptRT, permitió la ejecución sin problemas de funcionamiento, tanto en los aspectos de tiempo real como del hardware, aunque con unos tiempos algo superiores al Xenomai. Es por eso que en este trabajo se optó por probar este parche para el nuevo modelo y con la versión más reciente de Raspbian. Los resultados durante la fase experimental fueron satisfactorios, cumpliendo las características de tiempo real y sin perjudicar al desempeño de los puertos.

```
PREEMPT    #Min: 00013µs #Avg: 00021µs #Max: 0307µs
PREEMPT_RT #Min: 00011µs #Avg: 00018µs #Max: 0058µs
```

Ilustración 31: Comparativa de tiempos de ejecución con y sin parche de tiempo real

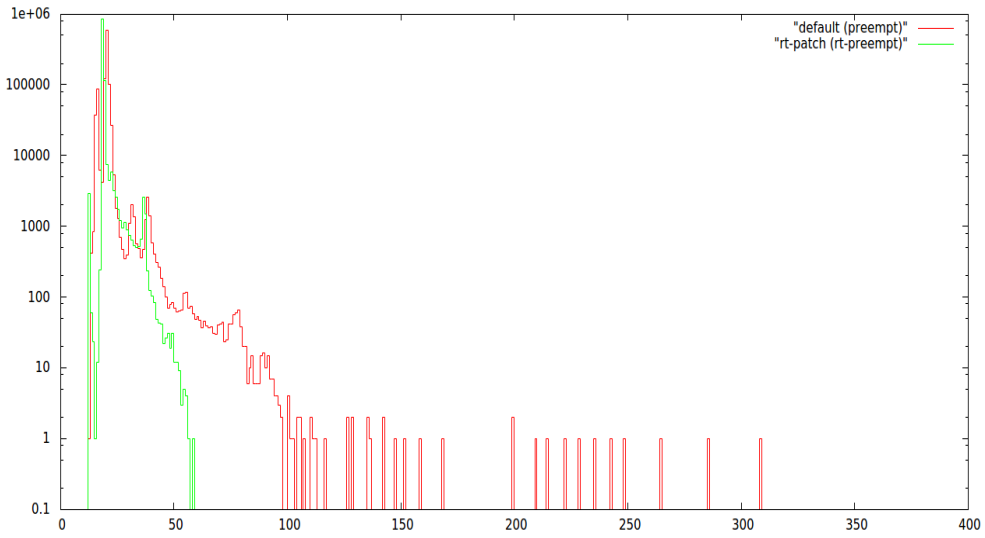

Ilustración 32: Comparativa de rendimientos con y sin parche de tiempo real

5.3 Lenguaje de Programación

5.3.1 Criterios de selección del lenguaje de programación

La elección de lenguaje de programación se suele guiar por unos criterios generales: entorno de ejecución, el tipo de programa a desarrollar y el paradigma de programación deseado.

El entorno de ejecución hace referencia a donde se ejecuta el programa. Existen dos tipos: lenguaje compilado y lenguaje interpretado. En el primero caen los programas que se ejecutan directamente sobre el sistema operativo, por tanto, se tratan de ficheros específicos para cada dispositivo, capaces de ser leídos por la máquina que lo ejecuta. Por otro lado, un programa escrito en lenguaje interpretado tiene carácter general para cualquier dispositivo, lo que requiere la existencia de un

intérprete (máquina virtual), que "traduzca" el lenguaje general del programa al lenguaje del computador que lo ejecuta. Esto obliga a la existencia de un nivel intermedio, lo que puede retrasar los tiempos de ejecución. Es por eso que se optará por una solución de lenguaje compilado.

Cuando se tratan de aplicaciones concretas, el criterio de un lenguaje específico suele tener más peso. Para aplicaciones de fuerte carácter matemático se suelen emplear lenguajes ideados para trabajar de forma eficiente con las operaciones matemáticas de alto nivel (ejemplos son el lenguaje M de Matlab o lenguaje Wolfram del Mathematica); para aplicaciones web se usan leguajes específicos orientados al desarrollo de aplicación en la red (JavaScript y HTML son ejemplos de lenguajes para aplicaciones web).

No obstante, el programa que se pretende desarrollar no encaja dentro de un carácter específico, por lo que se optará por la selección de un lenguaje de ámbito general. Para seleccionar entre la amplia gama de posibles lenguajes, habrá que determinar que paradigma de programación se pretende usar. Generalmente, existen dos tipos de paradigmas (realmente existen más, pero estos son los más extendidos): programación imperativa y programación orientada por objetos.

La programación imperativa (o por procedimiento) es el paradigma "convencional": un conjunto de órdenes e instrucciones que se ejecutan de manera secuencial. Esto permite la implementación de programas sencillos, pero se tornan complicados de desarrollar cuando la complejidad del programa crece. Es por este motivo por el que aparece la programación orientada por objetos, con el objetivo de simplificar el desarrollo.

5.3.2 Características de la programación orientada por objetos

El elemento clave en la Programación orientada por Objetos (POO) es, tal y como el propio nombre indica, los objetos. El objeto se entiende como una entidad que tiene unas determinadas características (atributos) y desempeña una serie de servicios (métodos). Los objetos constituyen toda la base de programación en este tipo de lenguajes, y el código resulta en una interacción entre unos objetos y otros.

Este formato de programación encaja con la imagen que crea la mente humana del mundo que la rodea, clasificando los elementos de la realidad dentro de grupos, con sus características y su interacción con el resto del mundo. Debido a esta analogía con la mente la POO resulta más sencilla de entender y de implementar.

La traducción de estos conceptos a elementos de programación sería la siguiente: los objetos son instancias de tipo clase, siendo el objeto el elemento en sí y la clase la categoría/concepto a la que pertenece.

Los atributos asociados a un objeto, dentro de la programación, se convierten en variables, que almacenan una cierta información sobre el objeto. Por otro lado, lo métodos se tratan como funciones, fragmentos de código que realizan un cierto servicio o trabajo. Bajo estas ideas surgen los principios en los que se basa cualquier lenguaje de programación orientada por objetos:

- Encapsulamiento: la idea de tener un objeto que contenga una serie de variables y funciones, relacionadas entre sí, que le otorguen una carácter de unidad. De este modo surgen las clases, emulando conceptos, y trasformar el programa en una estructura modular. El encapsulamiento también permite determinar que es y que no es accesible desde el exterior del propio objeto.

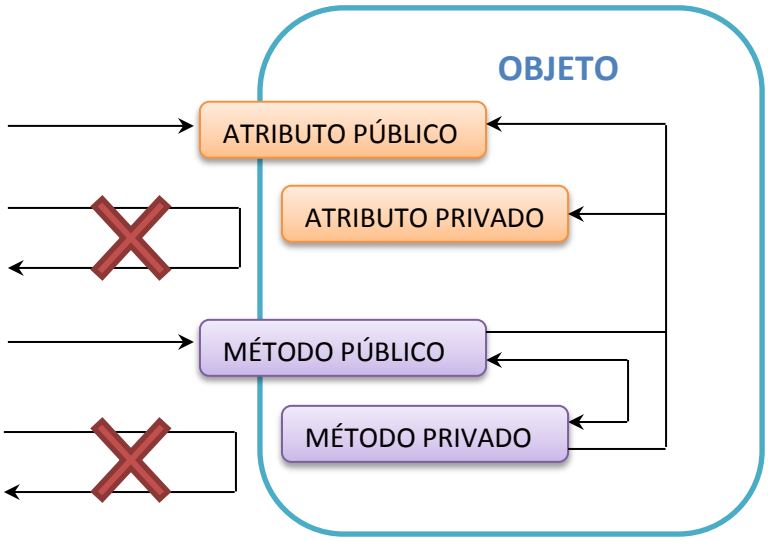

Ilustración 33: Estructura de encapsulamiento de un objeto

- Herencia: la herencia aparece por la manera que el cerbero estructura los conceptos. Primero, categorías (clases) más globales donde encajan muchas entidades diferentes. Entro de cada categoría, aparecen sub-categorías, con aspectos más concretos y restrictivos, pero siempre conservando las características de las categorías superiores. De este modo, se pueden crear clases y sub-clases, que heredan todos los métodos y atributos y agregan nuevos, sin tener que reescribir los aspectos más generales. Esto permite el ahorro de código que de otra manera sería repetido innecesariamente entre clases muy similares.

- Polimorfismo: el polimorfismo es la idea de un objeto perteneciente una clase en concreto (siendo esta heredera de otras clases más generales), puede, en un momento dado, comportarse como un objeto de una clase madre. Esto resulta lógico, dado que en ningún momento ha abandonado esa clase, tan solo se ha especificado, por lo que conserva todas las características originales.

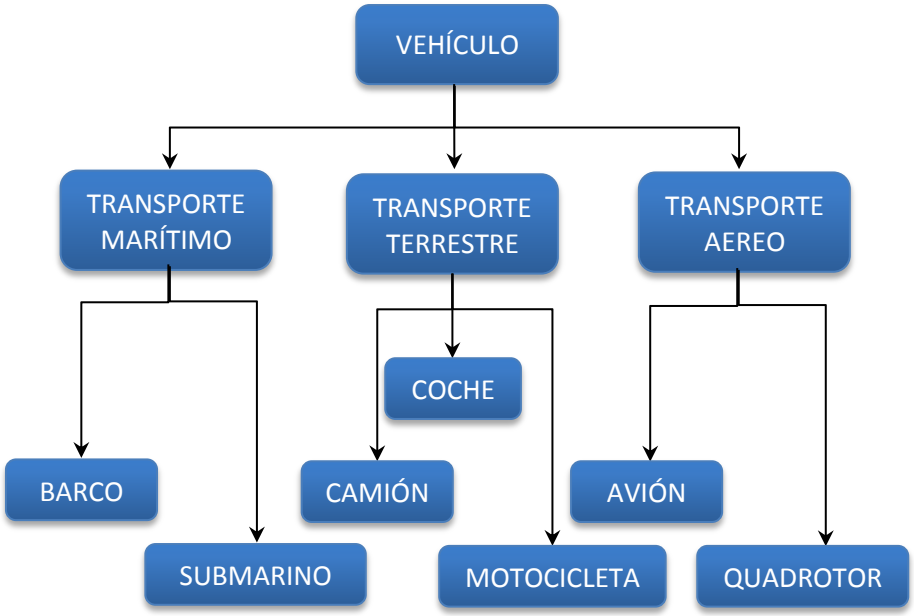

Ilustración 34: Ejemplo esquematizado de herencia de clases

5.3.3 Ventajas respecto a otros paradigmas de programación

Como se ha visto en el desarrollo de los últimos puntos, la POO presenta unas características que resultan muy beneficiosas para el software que se pretende desarrollar:

- Analogía estructural: la forma con la que se organizan los objetos dentro de un programa emula la forma con la que la mente humana estructura la realidad. Esto facilita el desarrollo y comprensión del código.

- Carácter modular: los objetos otorgan al programa de una estructura modular, donde cada objeto desempeña un papel en el conjunto, como si fueran distintos módulos con un objetivo concreto. Además, simplifica el código reduciéndolo a un conjunto de programas más reducidos, cada uno con una función específica.

- Protección de información: el encapsulamiento permite proteger las variables y funciones a las que no es deseable que un elemento externo acceda. De esta manera se puede garantizar la integridad del código

- Herencia: permite ahorrar grandes cantidades de código dado que muchos de los elementos que tiene el programa se repiten entre unos objetos y otros

- Jerarquía: permite crear con más facilidad una jerarquía entre los distintos objetos y cómo interactúan estos entre sí.

La mejor opción es, por tanto, un lenguaje que permita la programación orientado por objetos.

5.3.4 Solución adoptada

El lenguaje escogido para el desarrollo del programa ha sido C++. No se trata de un lenguaje POO puro, dado que permite programar en lenguaje C (imperativo). Además, es un lenguaje compilado, por lo que el resultado del programa es un ejecutable para un dispositivo concreto.

La estructura de los documentos es similar a un proyecto en C, donde los ficheros .h almacenan la declaración de la clase y los .cc (archivos C++) la funcionalidad de la misma. Se puede trabajar sin problemas con archivos .c (archivos C) y complementar la funcionalidad de ambos paradigmas, lo que resulta beneficios dado que ciertos elementos del código no tienen un comportamiento como objeto.

6 DESCRIPCIÓN DE LA FUNCIONALIDAD

6.1 Estructura del Programa

6.1.1 Hilos de ejecución

Como se ha indicado a lo largo del trabajo, el programa consta de diversos hilos asociados a distintas funcionalidades. Estos hilos (o tareas) se ejecutan simultáneamente, compartiendo información entre ellos.

Los 5 hilos presentes en el programa son:

- Hilo de Medida: encargado de efectuar las lecturas del correspondiente sensor y las interpreta. Es el hilo de máxima prioridad (49) al depender todo el proceso de él y se ejecuta a una frecuencia de 333Hz. En la práctica, se compone de dos hilos similares, uno para cada sensor.

- Hilo de Procesado: tarea que toma los valores obtenidos en el hilo de sensor y las procesa para obtener una medida corregida. El segundo hilo más importante en tema de prioridad (45) y se ejecuta a menor frecuencia que el de Medida, 200Hz.

- Hilo de Control: con los valores de posición calculados en el hilo de Kalman y las referencias deseadas entregadas por el hilo de comunicación, calcula la acción correctora para los motores. El hilo más crítico y el que se ejecuta con mayor frecuencia, 500Hz, y el tercero en la escala de prioridad (40)

- Hilo de Actuación: interpreta la medida de corrección calculada por el hilo de control y envía la acción correspondiente a los motores. Al ser el último de la cadena de control, su prioridad es más baja (35), y se ejecuta a una frecuencia de 100Hz, que corresponde a la frecuencia de la señal PWM.

- Hilo de Comunicación: recibe las órdenes del PC de tierra y se encarga de transmitirlas al hilo de control. El hilo con menor prioridad (25) al no ser crítico en el desempeño del control., por lo que se ejecuta a 50Hz

Si bien los hilos se ejecutan de manera independiente unos de otros, existen unas conexiones y un flujo de información entre ellos, como puede observarse en la imagen (Ilustración 35). El hecho que los hilos se ejecuten sin seguir esta secuencia se debe a dos motivos: Primero, para poder asignar distintas prioridades a cada uno de los hilos. Y segundo, para ajustar la frecuencia de ejecución de cada uno de los procesos. De esta manera, los hilos trabajan sin que las frecuencias de cada uno de ellos dependan de la frecuencia del hilo previo en la secuencia.

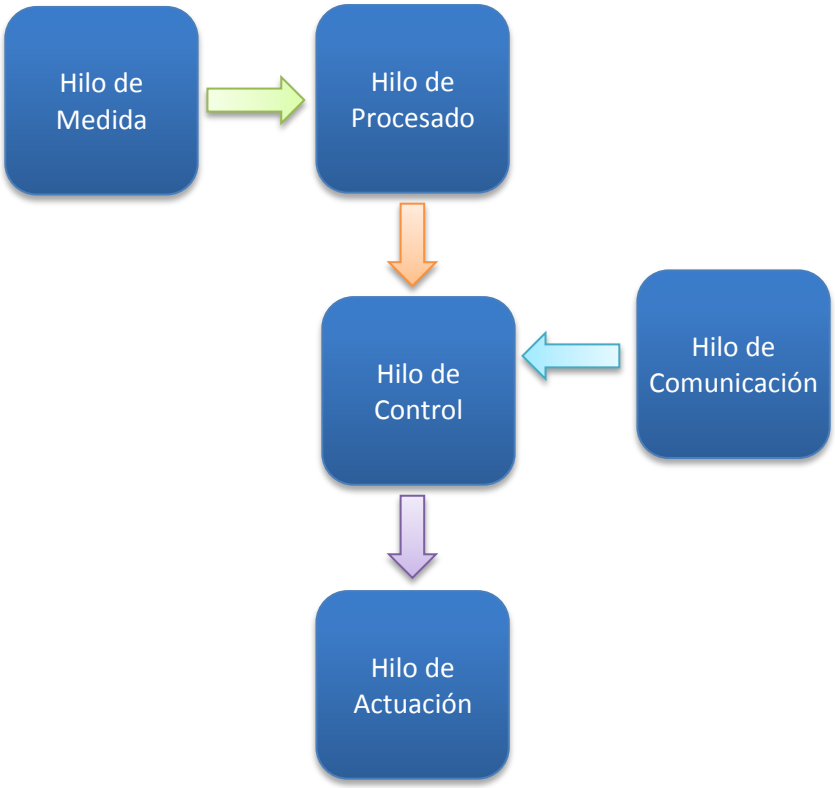

Ilustración 35: Flujo de información entre los hilos del programa

6.1.2 Jerarquía de objetos

Homólogamente a la estructura de hilos de programa, se crea la estructura de objetos. Los objetos de orden superior incluyen en su interior a los objetos de orden inferior, lo que otorga al esquema de organización de una jerarquía. Las clases no dejan de ser un tipo de variable, por lo que los objetos se almacenan entre sí como atributos.

Primeramente, se encuentra el objeto RPAS, el cual incluye en su interior todos los demás elementos. En su interior se encuentra el objeto RaspberryPi, encargado de ejecutar el algoritmo de control, y el Servidor, que permite la comunicación con el ordenador de tierra. Siguiendo la organización lógica, el objeto Raspberry contiene el objeto Motores, encargado de generar las pertinentes señales de PWM, y los objetos de los Sensores. Sin embargo, como las medidas de los sensores necesitan un filtrado previo, entre estas dos etapas aparece el objeto Filtro. Por debajo de cada objeto sensor aparece otro, i2cdevice, encargado de realizar la comunicación I^2C. Esta jerarquía permite fácilmente apreciar el flujo de información del programa

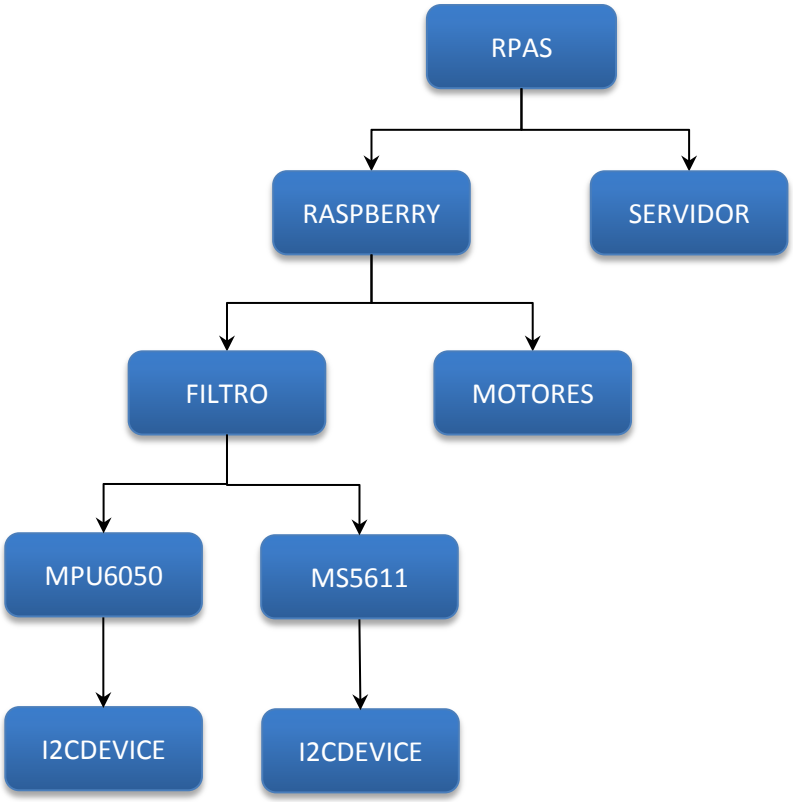

Ilustración 36: Esquema jerárquico de los objetos del programa

6.2 Lectura de Sensores

6.2.1 Lecturas de la IMU

Antes de empezar a realizar lecturas de la IMU, se debe primero configurar bajo qué condiciones trabajará. Existen tres aspectos importantes que deben ser configurados, para poder interpretar correctamente las medidas más adelante:

- Frecuencia de muestreo: como la mayoría de los sensores, la frecuencia con la que efectúa una muestra es configurable. Normalmente, la disminución del tiempo entre medidas supone un incremento del error de la misma, por lo que se debe encontrar un equilibro velocidad (para no comprometer la respuesta global del sistema) y precisión (con el fin de asegurar que el error sea mínimo). La IMU que incorpora el RPAS es de buena calidad, lo que significa que puede trabajar a frecuencias relativamente altas sin perjudicar demasiado las medidas. Al igual que el hilo asociado a las medidas (ver 6.1.1 Hilos de ejecución), se configura el sensor para que trabaje a 333Hz.

- Rango del giróscopo: aunque realmente la IMU incluye tres giróscopos (uno en cada uno de los tres ejes), no se puede trabajar directamente sobre cada uno de ellos, por lo que las configuraciones son generales para los tres. Es importante establecer el rango de trabajo porque afectará directamente a la precisión de las medidas. Un rango innecesariamente grande (capaz de medir velocidades angulares mayores), supone una notable pérdida de resolución, mientras que un excesivamente pequeño supondría no percibir movimientos rápidos del quadrotor. Como las medidas se toman dentro de un sistema controlado que evita las variaciones bruscas de la posición, el rango no tiene que ser muy grande. Por seguridad, se establece un rango de ±500º/s.

- Rango del acelerómetro: al igual que en el caso de los giróscopos, los tres acelerómetros que incluye el sensor se configuran conjuntamente. El problema que surge aquí es igual al del giróscopo, se debe seleccionar un rango que garantice la mejor resolución posible dentro del rango de trabajo. En este caso hay que tener en cuenta el factor de la gravedad, lo que supone la existencia siempre de una aceleración vertical en el sistema. Por tanto, se debe elegir un rango superior a 1g (valor de la aceleración de la gravedad terrestre), para no perder información si se produce un descenso vertical. Por otra parte, como se trata de un sistema controlado que impide movimientos bruscos, los valores de las aceleraciones no será excesivos, por lo que rango no debe ser grande La opción escogida que satisface estos criterio es de ±2g.

Una vez realizada la configuración, la lectura de datos resulta muy sencilla dado que la IMU cuenta internamente con un controlador que se encarga de efectuar las medidas periódicamente y procesarlas, y un conversor analógico-digital para cada variable, lo que permite almacenarlas simultáneamente cada una en un registro. Basta, por tanto, en cada ciclo de la tarea leer los 12 bytes de información (6 medidas de 2 bytes de longitud cada una) e interpretarlos correctamente [8].

Los valores que se obtienen de la lectura son un número comprendido entre 0 y 65534, valor máximo decimal que se puede obtener con 2 bytes (16 bits). Conocido el rango, es fácil obtener el valor real con una sencilla ecuación lineal. Aquí es donde entra en juego la selección apropiada del rango de medida: dado que la longitud de palabra es fija, ampliar el rango supone reducir la resolución, que en el caso de la velocidad angular sería de 0,01526º/s (valor mínimo que se puede medir).

Es importante calibrar correctamente la IMU, es decir obtener correctamente su *offset* de medida y su distribución de ruido. El primero permite ajustar la recta para garantizar que pasa por el origen de coordenadas y evitar acumulación. Esto resulta muy importante en el caso del yaw, como se verá más adelante, para reducir al máximo las desviaciones, El segundo, por otro lado, se usará en una de las etapas de filtrado de señal.

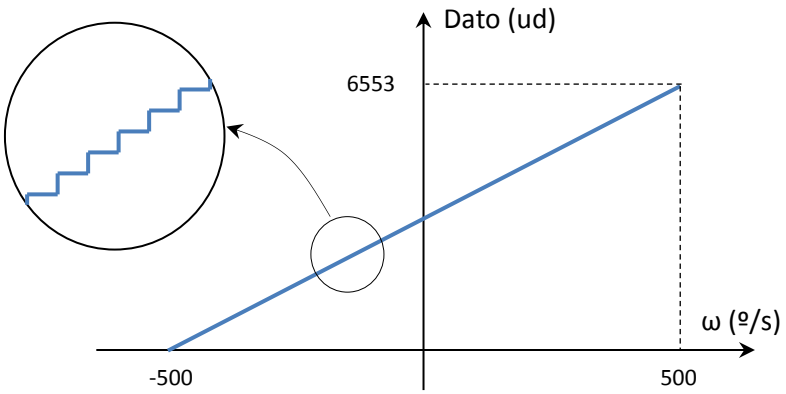

Ilustración 37: Función de transferencia del giróscopo

6.2.2 Lecturas del barómetro

La lectura del barómetro presenta notables desventajas respecto a la manipulación de la IMU pues carece de un procesador interno y de conversores individuales. Esto presenta dos grandes problemas: no se puede configurar inicialmente el sensor con una frecuencia de muestreo ni se pueden medir ambas variables (presión y temperatura) en cada ciclo. Esto obliga a enfocar las y las lecturas de una forma diferente.

Al no poder configurar la frecuencia de muestro, cada vez que se desee una nueva medida se debe "solicitar" al sensor. El tiempo entre la solicitud y la entrega de la misma es configurable, afectando esta selección directamente la longitud (número de bits) de la medida resultante. Cuanto menor sea el tiempo que disponga el sensor para realizar la medida, menor será el rango de la medida obtenida (y con él, también la resolución). Al finalizar la rutina de lectura del barómetro, se ordena la mesura de la siguiente medida. Por tanto, es importante ajustar correctamente la frecuencia de ejecución del hilo de lectura del sensor y el tiempo que se permite para realizar la medida para evitar incoherencia de datos.

Por otro lado, al no existir conversores para cada una de las dos magnitudes impide la medición simultánea de ambas, por lo que en cada ejecución del hilo se leerá una, alternativamente. Al finalizar la lectura, como se indicó más arriba, se solicitará la captura de la siguiente medida, que deberá efectuarse sobre la otra variable. Esto supone dos inconvenientes; por un lado, las medidas de ambas magnitudes nunca se producirán en el mismo instante, lo que el consiguiente cálculo de la altura, dependiente de estas dos mediciones, sufrirá un ligero desajuste; por otro lado, al requerir dos ciclos de rutina para realizar una medida exacta, supone que la frecuencia efectiva se reduce a la mitad, por lo que la velocidad de respuesta a cambios de altura puede reducirse [9].

Dado que el sensor no incluye una unidad de procesado de mediciones, los valores obtenidos sufren muchas desviaciones, por lo que deben ajustarse mediante unos parámetros de calibrado del propio sensor, para corregir dichas mediciones.

Una vez obtenidos unos valores de presión y temperatura más fiables, se puede realizar un cálculo aproximado de la altitud absoluta respecto a nivel del mar (ecuaciones 6.1 y 6.2) [18].

$$p(h_1) = p(h_0) \cdot e^{-\frac{Mg}{RT} \cdot \Delta h} \tag{6.1}$$

$$\Delta h = -\frac{\ln\left(\frac{p(h_1)}{p(h_0)}\right) \cdot R \cdot T}{M \cdot g} \tag{6.2}$$

6.2.3 Gestión del bus

La placa que contiene los únicos dos sensores está conectada al único puerto I^2C que tiene la Raspberry. Ambos sensores harán uso de dicho bus durante el funcionamiento del programa, lo que hace imperativa una correcta gestión del mismo.

La lectura de la IMU y del barómetro se realiza en tareas independientes, por lo que a priori no existe ningún control de cual tiene autorización para usar el bus. Si uno de los hilos intenta acceder a su respectivo sensor mientras el bus está ocupado, se le impedirá el acceso y proseguirá con la ejecución de su rutina sin tomar un nuevo valor de medida, tornando este ciclo de ejecución inútil. Para solucionar este problema, su usa un mutex. Tal y como se explica en la sección correspondiente (ver 5.2.2 Características de un SOTR), los mutex se implementan para controlar el uso de un recurso compartido entre varias tareas. En este caso, el recurso sería el bus I^2C y las tareas en cuestión serían los hilos de la IMU y el barómetro.

Cuando una tarea inicia la comunicación con su respectivo sensor, bloquea el mutex asociado. De este modo, si el planificado del sistema operativo da paso a la tarea del otro sensor, esté encontrará el mutex bloqueado y no podrá acceder al bus, por lo que detendrá su ejecución en ese punto. Cuando la tarea del primer sensor termine la comunicación, desbloqueará el mutex, permitiendo a la segunda tarea retomar la ejecución y conectar con su sensor. De este modo se evitar perder un ciclo del hilo.

Con tan solo dos sensores la posibilidad de fallo es muy baja, pero tratándose de las lecturas de la IMU, que resultan críticas para la estabilidad del quadrotor, se deben evitar al máximo problemas de este tipo. Además, esta gestión permitiría en futuros trabajos la inclusión de sensores adicionales u otros dispositivos que también realicen la comunicación por este bus.

6.3 Procesado de Señal

6.3.1 Necesidad de procesar las mediciones

Los valores medidos, aun habiendo sufrido una etapa de procesado, siguen sin coincidir con los valores deseados. En el caso de la IMU, se obtienen velocidades angulares y aceleraciones lineales. No obstante, lo que se busca son los ángulos de orientación (pitch, roll y yaw), por lo que habrá que obtenerlos a partir de los anteriores. En el caso del altímetro, la medida de altitud presenta excesivo error, por lo que se usarán las aceleraciones para corregirla.

Toda esta etapa la ejecutará la tarea asociada al objeto Filtro, que recibe este nombre por el motivo que se verá a continuación.

6.3.2 Fundamentos del filtro de Kalman

Desarrollado por Rudolf E. Kalman en 1960, el filtro de Kalman es un algoritmo que permite estimar el valor de unas determinadas variables de estado dadas una serie de mediciones y el modelo del sistema. El filtro contempla los errores, tanto del modelo como de las mediciones, y los modela como distribuciones normales [19].

Se trata de un algoritmo recursivo que aprovecha el último valor y la ecuación de comportamiento del sistema (lineal y de tiempo discreto) para realizar una predicción de las variables en el siguiente instante. Después, con la nueva medida, corrige la predicción realizada anteriormente, basándose en la fiabilidad de cada una ellas.

Es un filtro ampliamente extendido y que se utiliza en multitud de campos, desde la navegación espacial, donde tuvo su primera aplicación al usarse para el control de posición del programa Apolo; hasta la fotografía, donde se utiliza para el seguimiento de rasgos en una secuencia de imágenes. En este caso, se usará el cálculo de la orientación y la altura del quadrotor.

La estructura del filtro es la siguiente. Dadas unas variables de estado a controlar, el vector de estado *x* en el instante *k* puede escribirse en función de los valores del instante previo, *k -1*, conocido el modelo del sistema, se puede expresar lo siguiente (ecuación 6.3).

$$x_k = A \cdot x_{k-1} + B \cdot u_k \tag{6.3}$$

Por otra parte, el vector que contiene las medidas de estas variables de estado, para un cierto instante, puede expresarse de la siguiente manera (ecuación 6.4):

$$z_k = H \cdot x_k \tag{6.4}$$

No obstante, estas expresiones no modelizan correctamente el sistema real, al no incluir los errores de medida y los errores del modelo respecto al comportamiento real. Incluyendo el error en las expresiones anteriores se obtienen las ecuaciones 6.5 y 6.6.

$$x_k = A \cdot x_{k-1} + B \cdot u_k + w_{k-1} \quad (6.5)$$

$$z_k = H \cdot x_k + v_k \quad (6.6)$$

Nótese que el error del modelo tiene subíndice *k -1* mientras que el de la medida tiene *k*. Esto se debe a que el cálculo para el instante *k* se hace con valores del instante previo, por lo que el resultado dependerá del error previo. Por otro lado, la medida *k* se efectúa en ese mismo instante.

El objetivo ahora es realizar una combinación de ambas estimaciones para obtener una mejor aproximación. Y la clave está en la matriz de ganancia *K*, la cual determina que "peso" se le debe dar a cada uno de ellas (medición y predicción). Esta matriz se varía en el tiempo, ajustándose a la dinámica del sistema, recogiendo las desviaciones y ruidos de las aproximaciones.

Se suponen los ruidos del sistema y de las mediciones de distribución gaussiana, con matrices de covarianza conocidas (*Q* y *R* respectivamente). La matriz P contiene la correlación de errores de la predicción respecto al valor corregido. Con esto, se pueden escribir las ecuaciones de predicción y corrección como sigue:

Ecuaciones de predicción:

$$\hat{x}_k = A \cdot x_{k-1} + B \cdot u_k \quad (6.7)$$

$$P_{k-1} = A \cdot P_{k-1} \cdot A^T + Q \quad (6.8)$$

Ecuaciones de corrección:

$$K_k = P_{k-1} \cdot H^T \cdot (H \cdot P_{k-1} \cdot H^T + R)^{-1} \quad (6.9)$$

$$\hat{x}_k = \hat{x}_{k-1} + K_k \cdot (z_k - H \cdot \hat{x}_{k-1}) \quad (6.10)$$

$$P_k = (I - K_k \cdot H) \cdot P_{k-1} \quad (6.11)$$

Donde el acento circunflejo sobre el vector de estado *x* indica que los valores de ese vector son fruto de la predicción realizada. Se puede observar como la matriz *K* responde a la exactitud de ambas aproximaciones. Planteado los casos de error cero tanto para la medida como para predicción, se obtienen los siguientes valores de *K* (ecuaciones 6.12 y 6.13).

Para error de medida nulo:

$$\lim_{R \to 0} K = H^{-1} \tag{6.12}$$

Para desviación de predicción nula:

$$\lim_{P \to 0} K = 0 \tag{6.13}$$

Que una vez aplicados ambas matrices en la ecuación 6.8, permiten obtener sendos del vector de estado:

$$x_k = H^{-1} \cdot z_k \tag{6.14}$$

$$x_k = \hat{x}_k \tag{6.15}$$

Como se puede observar, la ecuación 6.14 resulta de despejar el vector de estado de la ecuación de la medida 6.4. Por tanto, la matriz de ganancia dará prioridad a aquellas aproximaciones que tengan el menor error posible [20].

6.3.3 Corrección de la altura

La altura barométrica ofrecida por el altímetro tiene demasiado ruido (demasiado para el rango de altura en el que se mueve el quadrotor, para otro tipo de aplicación puede ser más aceptable), lo que obliga a corregirlo, en la medida de lo posible. Observado el buen resultado obtenido con el Filtro de Kalman en control de pitch y roll en proyectos anteriores, se opta por esta solución [21].

El sensor nos aporta la medida, queda por determinar el modelo fisco del sistema. La ascensión se produce por un movimiento rectilíneo [uniformemente] acelerado, que se rige por las conocidas ecuaciones 6.16 y 6.17:

$$y = y_0 + v_0 \cdot t + \frac{1}{2} \cdot a \cdot t^2 \tag{6.16}$$

$$v = v_0 + a \cdot t \tag{6.17}$$

Cabe destacar el detalle de uniformemente acelerado; si bien es cierto que la aceleración no tiene por qué ser constante en todo el proceso de ascensión, al trabajar el algoritmo en tiempo discreto, se puede considerar que la aceleración es constante durante un pequeño lapso de tiempo. Conocido esto, basta aplicar el filtro.

Plantéese las variables de estado del sistema como la altura, la velocidad de ascensión (o descenso, según sea el caso) y la aceleración vertical. Por tanto, puede escribirse el vector de estados de la siguiente forma (ecuación 6.18).

$$x = \begin{pmatrix} h \\ \dot{h} \\ \ddot{h} \end{pmatrix} \tag{6.18}$$

Como el modelo del sistema es conocido y siendo Δt el intervalo de tiempo transcurrido (este valor en la práctica se obtiene conociendo la frecuencia de ejecución del hilo), se pueden escribir la matriz *A* (ecuación 6.19) del modelo. La matriz *B y* el valor *u* (6.20) no aparecen en este ejemplo:

$$A = \begin{pmatrix} 1 & \Delta t & (\Delta t)^2/2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{pmatrix} \tag{6.19}$$

$$B, u = 0 \tag{6.20}$$

De las variables de estado se tienen dos medidas, una de la altura y otra de la aceleración (que se comentará más adelante). Por tanto, se puede escribir la matriz de medidas como se observa en la ecuación 6.21.

$$H = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{6.21}$$

Por último, se determinan las matrices de covarianza. Al haber dos sensores, la matriz *R* es 2x2. Se supone que no existe interferencia entre los sensores, por lo que dicha matriz será diagonal, con la varianza de cada sensor en la misma, tal y como se observa en la ecuación 6.22. Por otra parte, se supone el que el modelo es exacto, por lo que se suprime la matriz de covarianza del sistema (ecuación 6.23)

$$R = \begin{pmatrix} \sigma_h & 0 \\ 0 & \sigma_{\ddot{h}} \end{pmatrix} \tag{6.22}$$

$$Q = 0 \tag{6.23}$$

La aceleración vertical no es un valor directo, se compone a través de las aceleraciones de los tres ejes del quadrotor. Esto supone hacer una combinación de las tres, conocidos el pitch y roll, para obtener el valor de la aceleración del eje vertical (absoluto).

El acelerómetro mide en todo momento la aceleración de la gravedad (1g) que, salvo que el quadrotor esté inclinado en alguno de sus ejes, se aplicará al eje z del acelerómetro. Si se inclina y el quadrotor no tiene movimiento vertical, la gravedad se "reparte" entre los tres ejes, pero la composición de los tres valores debe dar la gravedad. No obstante, si se encuentra en movimiento, el valor obtenido será diferente a la unidad, lo que permitirá obtener la aceleración vertical (ecuaciones 6.24 – 6-26).

$$\lambda = \sqrt{\sin(\alpha)^2 + \sin(\beta)^2} \tag{6.24}$$

$$a_{corr} = \sqrt{1 - \lambda^2} \tag{6.25}$$

$$\ddot{h} = 1 - \frac{a_z}{a_{corr}} \tag{6.26}$$

La aceleración vertical que se obtiene de la IMU considera positivas las aceleraciones descendentes, de ahí que deba restarse. El ruido de medida se tomará por simplicidad el de la aceleración en el eje z, dado que su valor es el que más peso tiene en el término de la aceleración vertical (ecuación 6.27).

$$\sigma_{\ddot{h}} = \sigma_{a_z} \tag{6.27}$$

6.3.4 Cálculo del pitch, roll y yaw

El pitch y el roll tienen un método de cálculo similar, basado en el mismo modelo. No obstante, este modelo no es aplicable al yaw, por lo que no se logrará tanta precisión como en los otros dos.

El método más sencillo para obtener el ángulo (cualquiera de los tres) es mediante la integración de su respectiva velocidad angular a lo largo del tiempo. No obstante, las medidas de velocidad no son exactas y, aun cuando no existe movimiento, las medidas son distintas de cero, lo que produce una acumulación de error en la medida del ángulo al transcurrir del tiempo. Estas desviaciones pueden producir fallos que afecten al funcionamiento del sistema, especialmente en lo que refiere al pitch y el roll, que son los ángulos determinantes para la estabilidad. Esto obliga a buscar un método más preciso para el cálculo de estos dos ángulos críticos.

Como las aceleraciones lineales son conocidas, se pueden usar para ajustar las medidas. En el caso de ausencia de inclinación, la aceleración de la gravedad, como se vio en el punto anterior, se concentra únicamente en el eje z del quadrotor. No obstante, cunado la orientación cambia, la gravedad afecta también a los otros ejes. Con unos simples cálculos trigonométricos, se puede calcular el ángulo de cada uno de ellos para que se ajusten a la distribución de aceleración (si bien se ha indicado que la aceleración debe ser la de la gravedad, los cálculos son válidos para una aceleración diferente, siempre que sea únicamente vertical).

$$a_v = \sqrt{a_x^2 + a_y^2 + a_z^2} \tag{6.28}$$

$$sin(\phi) = \frac{a_y}{a_g} \tag{6.29}$$

$$sin(\theta) = \frac{a_x}{a_g} \tag{6.30}$$

De esta manera (ecuaciones 6.28 – 6.30), se puede lograr una aproximación del ángulo conocido a través de las aceleraciones. Por otra parte, se conoce un modelo matemático que relaciona el roll y su velocidad angular, en la ecuación 6.31:

$$\phi = \int_{t_0}^{t} \dot{\phi}\,(t)\,dt \tag{6.31}$$

Suponiendo el tiempo discreto, se puede considerar que la velocidad es constante en un pequeño lapso de tiempo, por tanto la expresión anterior se puede reescribir, para obtener la expresión 6.32:

$$\phi = \phi_0 + \dot{\phi} \cdot (t - t_0) \tag{6.32}$$

Para el pitch, las expresiones serían idénticas a las del roll, cambiando por la correspondiente velocidad angular. Al igual que en el caso de la altura, se tiene una medida y un modelo del sistema. Por tanto, aplicando un desarrollo similar al anterior, se puede obtener el filtro de Kalman para el pitch y el roll.

Por otro lado, el yaw no permite la aplicación de este algoritmo pues no existe la posibilidad de obtener una medida del yaw (directa o indirecta). Por tanto, la única posibilidad de es obtener una estimación a través del modelo matemático que se ve en la ecuación 6.33:

$$\psi = \int_{t_0}^{t} \dot{\psi}\,(t)\,dt \tag{6.33}$$

Y tomando de nuevo la consideración de velocidad constante durante un pequeño lapso de tiempo, se obtiene la expresión 6.34:

$$\psi = \psi_0 + \dot{\psi} \cdot (t - t_0) \tag{6.34}$$

Como ya se ha indicado, esta expresión acumula error a lo largo del tiempo debido al error de la medida de la velocidad. No obstante, este error no es crítico pues no compromete la estabilidad del RPAS.

6.4 Algoritmo de Control

6.4.1 Fundamentos de un PID

Esta es la parte más relevante del programa, donde se realiza el control de estabilidad y orientación del quadrotor. El algoritmo implementado es un controlador PID (*Proportional–Integral–Derivative*) en paralelo, el cual establece un lazo cerrado de control.

El controlador PID se basa en la realimentación del valor de la variable a controlar y la entrada del valor de referencia de la misma. Esto proporciona un error, entre el valor deseado y el real. Con este error se calcula mediante el PID la acción de control. Esta acción de control consta de tres partes:

- Proporcional: acción directamente proporcional al error actual.
- Integral: acción asociada a la acumulación del error a lo largo del proceso.
- Derivativa: proporcional a la derivada (o velocidad) del error

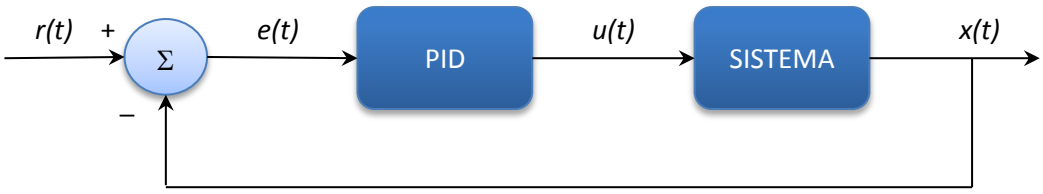

Ilustración 38: Bucle de control PID

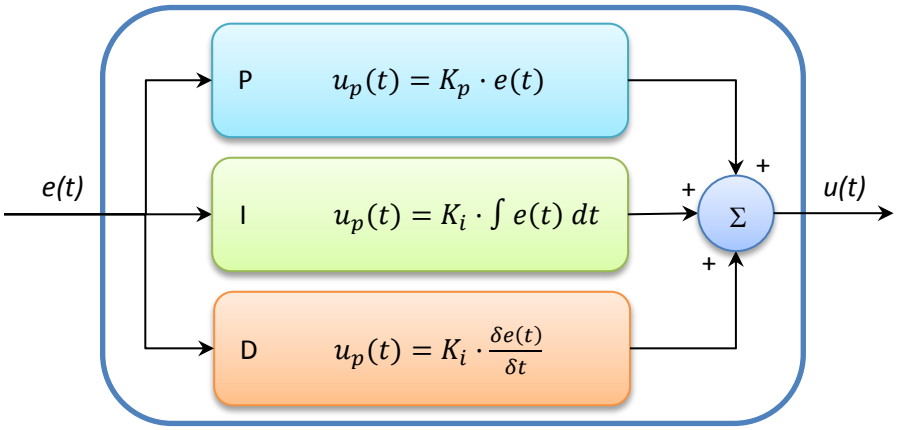

Ilustración 39: Estructura interna de un controlador PID

En la Ilustración 38 puede observarse como la diferencia entre la referencia (r) y el valor de la variable controlada (x) resultan en el error (e). Este error se introduce en el controlador PID, el cual dados los parámetros de control (K_p, K_i y K_d) calcula la pertinente acción de control (u) que se introduce en el sistema para intentar corregir la desviación con la referencia. El ajuste de las ganancias del controlador permite adaptar la respuesta del sistema.

6.4.2 Determinación de los parámetros de control

Este bucle debe aplicarse a todas las variables a controlar, que en este caso son cuatro (pitch, roll, yaw y altura). Cada una de ellas tiene un comportamiento diferente (diferente sistema), por lo que el ajuste del controlador deberá realizarse de forma individual para cada una de ellas. En los cuatro casos la acción de control será la misma: el régimen de giro de los motores.

Existen dos maneras de calcular las tres constantes del PID: conociendo el modelo matemático que rige el comportamiento de la variable en función de la acción o mediante sintonizado experimental.

Si se conoce el modelo matemático (relaciona x con u), introduciendo el PID (relaciona u con e), entonces se puede obtener una relación directa entre el error y la respuesta. Como esta función de transferencia depende de los parámetros del PID, se podrá ajustar para que el sistema tenga un cierto comportamiento (respuesta rápida, baja oscilación, error cero, etc.). No obstante, en muchas ocasiones, el modelo del sistema es desconocido o extremadamente complejo, como ocurre en este caso. Si bien hay modelos matemáticos simplificados que permiten entender el comportamiento del quadrotor, resultan incompletas en la práctica porque no tienen en cuenta factores importantes que afectan al sistema. Esto condiciona que, aun obteniendo unos valores para los parámetros, haga falta un reajuste experimental posterior.

En este caso se ha optado por un sintonizado experimental de los parámetros de control, dado que se pueden obtener muy buenos resultados sin necesidad de conocer el modelo matemático. A la hora de realizar el ajuste experimental, hay una serie de consideraciones a tener en cuenta para obtener una buena respuesta [22]:

- Proporcional: Tiene una respuesta rápida, pero puede desestabilizar el sistema si el valor es excesivo.

- Integral: Tiene respuesta lenta, pero garantiza error nulo

- Derivativa: Permite responder fuertemente a cambios en el error, pero resulta muy sensible al error de medida.

6.4.3 Control de orientación y altura

Las ecuaciones del PID se plantean para el caso de tiempo continuo. Sin embargo, internamente el control del quadrotor se realiza de forma discreta, en cada ciclo del hilo, por lo que dichas ecuaciones deben adaptarse a estas condiciones. Las ecuaciones para cierto instante k, conocido el espacio de tiempo (*Δt*), serán 6.35 – 6.38.

$$u_p^k = K_p \cdot e^k \tag{6.35}$$

$$u_i^k = u_i^{k-1} + \frac{\left(e^k - e^{k-1}\right)}{2} \cdot \Delta t \tag{6.36}$$

$$u_d^k = \frac{\left(e^k - e^{k-1}\right)}{\Delta t} \tag{6.37}$$

$$u^k = u_p^k + u_i^k + u_d^k \tag{6.38}$$

La acción de control u representa el valor porcentual del régimen de los motores. Para evitar que una cierta acción (proporcional, integral o derivativa) tome un valor excesivo, se saturan los valores máximos y mínimos que pueden tomar cada una. Esta acción de control deberá calcularse para las cuatro variables (pitch, roll, yaw y altura), y después asignarlas a cada motor de forma que se estabilice el sistema, como se puede ver en las ecuaciones 6.39 – 6.42.

$$u_{motor_1} = +u_{roll} - u_{pitch} + u_{yaw} + u_{altura} \tag{6.39}$$

$$u_{motor_2} = +u_{roll} + u_{pitch} - u_{yaw} + u_{altura} \tag{6.40}$$

$$u_{motor_3} = -u_{roll} + u_{pitch} - u_{yaw} + u_{altura} \tag{6.41}$$

$$u_{motor_4} = -u_{roll} - u_{pitch} + u_{yaw} + u_{altura} \tag{6.42}$$

6.5 Actuación sobre los Motores

6.5.1 Saturación de la acción de control

La acción sobre el régimen de giro debe ser porcentual, asignando a cada motor un régimen entre un valor mínimo y un máximo. No obstante, la acción de control enviada puede tomar valores negativos, lo que se traduce en inversión de giro de los motores, lo cual no es posible. Por tanto, se debe aplicar saturación a las acciones de control de cada motor para que no tengan valores inferiores a 0 ni superiores a 1.

No obstante, convertir todas las acciones de control negativas en 0 produce un sesgo que puede producir errores. Para poder tener en cuenta estas acciones, se le asigna un patrón de referencia a todos los motores (ecuación 6.43), de tal manera que las acciones negativas tengan un margen para ser interpretadas.

$$u_{motor} = u_{referencia} + u_{control} \tag{6.43}$$

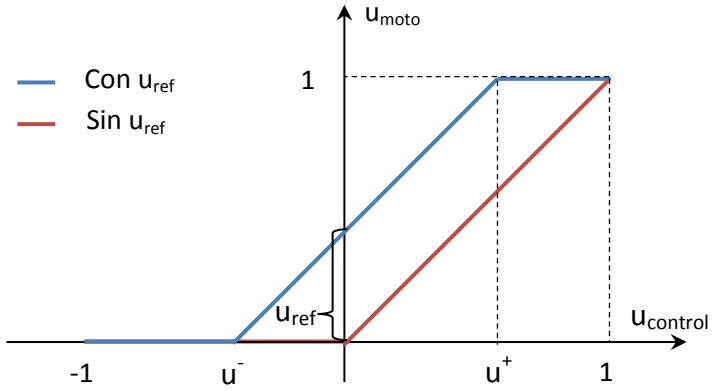

Ilustración 40: Acción de los motores en función de la acción de control

Como puede observarse en la Ilustración 18, añadir el valor de referencia tan solo desplaza la banda de valores de la acción de control. Sin embargo, esto resulta beneficioso, pues serán más comunes acciones negativas que valores positivos cercanos a la unidad.

6.5.2 Generación de la señal PWM

La señal PWM tiene dos parámetros para ajustar: el periodo (o frecuencia) y el ancho de pulso. La frecuencia de la señal se asignará en función de la frecuencia que sean capaces de interpretar los drivers, que en este caso es de 100Hz. Esta frecuencia coincide con la de ejecución del hilo de los motores, por lo que en cada ciclo se calcula el ancho de pulso del siguiente periodo.

Por otra parte, el ancho de pulso se determina en función de la acción, escalando entre el valor mínimo y máximo que son capaces de interpretar los drivers, que en este caso es entre 1145 y 1800 µs.

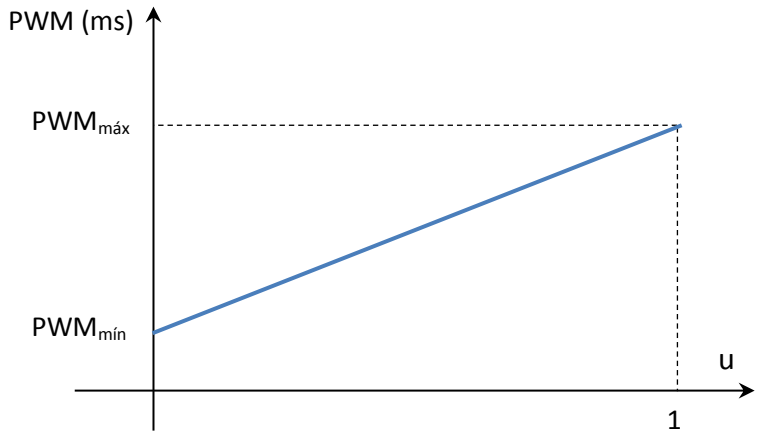

Ilustración 41: Función de trasferencia del pulso PWM

6.6 Control desde Tierra

Desde el ordenador de tierra se pueden controlar distintos aspectos que afectan al vuelo del quadrotor durante el mismo:

- Selección de modo: permite seleccionar entre modo automático y modo manual
    - Modo manual: se selecciona el régimen deseado de cada motor
    - Modo automático: se ajustan los parámetros de control y se deja que el algoritmo de control actúe sobre los motores
        - Referencias de la orientación y altura
        - Ganancias de los respectivos PID
        - Régimen de referencia de los motores

- Start/Stop control: inicia o detiene el control seleccionado. Cuando no hay control (ya sea manual o automático), los motores estarán detenidos)

- Resteo IMU: Toma la orientación y altura en el momento dado y los establece como referencia

Toda esta información se envía desde el Interfaz Hombre-Máquina y la Raspberry la recibe, para posteriormente incluirla en el algoritmo de control.

7 RESULTADOS EXPERIMENTALES

7.1 Introducción

Una vez construidos el quadrotor y la plataforma de ensayo y desarrollado el software de control, se puede iniciar el proceso de estudio de la capacidad de la Raspberry para realizar el control de un quadrotor y la competencia de la plataforma de ensayo.

El análisis del desempeño se hará mediante el estudio de las gráficas que se muestran a continuación, donde se reflejan las variables controladas. Es importante tener en cuenta que, aunque en las gráficas solo aparece la respuesta de una única variable, durante el proceso se están controlando todas a la vez.

7.2 Medida de Altura

Uno de los objetivos del trabajo era lograr realizar el control de altura del quadrotor. Para ellos se hizo uso del sensor barométrico que incluía la IMU (ver 4.3.2 Barómetro MS5611) y un ajuste posterior de la medida mediante un filtro de Kalman (ver 6.3.3 Corrección de la altura).

De partida se sabía que el error en el barómetro era alto para el rango de trabajo. Según las especificaciones técnicas, debería tener un error de ±10 cm en el caso óptimo. No obstante, debido a las condiciones de laboratorio y el encapsulamiento del sensor, el ruido se situaba en torno a los 20 cm, tal y como se puede observar en la gráfica de la Ilustración 42.

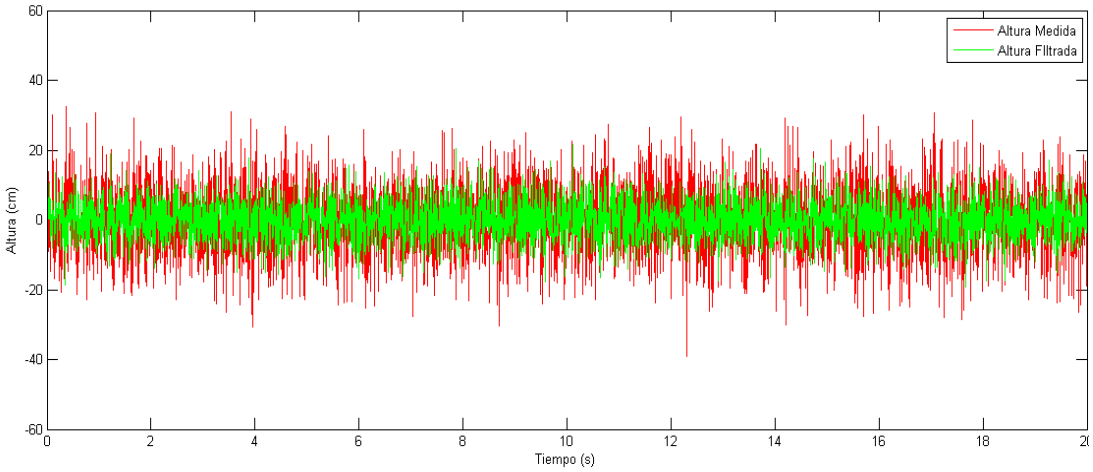

Ilustración 42: Medida de la altura

Tras la aplicación de la etapa de filtrado, se logró reducir el error de la medida. No obstante, los valores resultan excesivos para una medida estática, al situarse el ruido en unos ±12 cm, para un banco de pruebas que permite una libertad de movimiento de 40 cm

7.3 Control del Pitch

El ajuste del PID que controla el pitch, al igual que el de los restantes ángulos, se ha realizado, tal y como se indicó previamente (ver 6.4.2 Determinación de los parámetros de control), de forma experimental.

Un controlador PD, es decir, sin acción integral, es suficiente para garantizar la estabilidad del quadrotor. No obstante, si se pretende eliminar el error de referencia, como en este caso, se debe recurrir a un controlador PID.

Tras el proceso de ajuste de los parámetros del controlador y los posteriores ensayos, se obtiene su comportamiento ante cambio de referencia y perturbación.

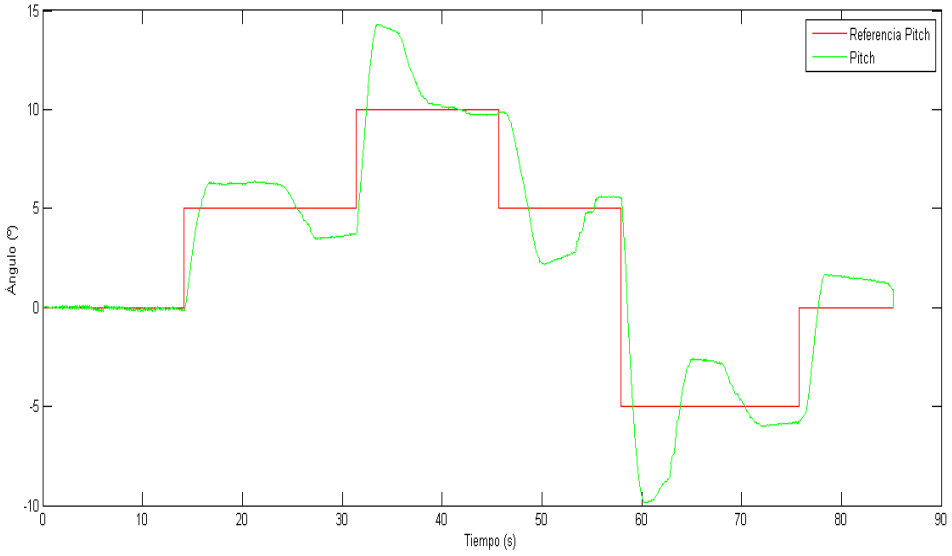

Ilustración 43: Respuesta del pitch ante cambio de referencia

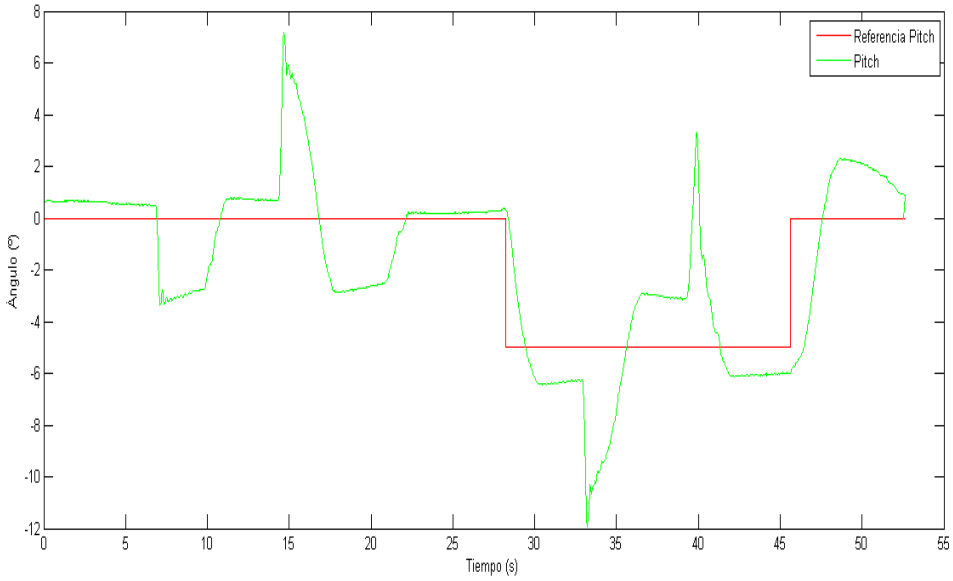

Ilustración 44: Respuesta del pitch ante perturbación

Como puede observarse en la primera gráfica, presenta un buen seguimiento de la referencia y una respuesta rápida, con ligera sobresocilación para referencias grandes. No obstante, presenta una pequeña desviación, como ocurre entre el segundo 40 y el 45, donde se aleja un grado. Esta discrepancia, pese a tener integrador, se debe a la rigidez que aporta la rótula de la plataforma: cuando el ángulo desviado es pequeño, las acciones correctoras también lo son, por lo que el par generado lo es lo suficientemente grande para contrarrestar la diferencia. Si se aumentara la constante de la integral, se lograría compensar esta divergencia, pero podría desestabilizar el sistema, lo cual sería peor que la pequeña desviación.

Por otra parte, cuando se produce una perturbación (en lo segundos 6, 15, 17, 34 y 40), el sistema corrige favorablemente, sin producir sobresocilaciones, presentando el error comentado debido a la rótula. Cuando la referencia cambia y se abandona el punto de equilibrio (segundo 28), la respuesta empeora, pero se mantiene siempre dentro de unos límites aceptables, sin superar el error de un grado.

7.4 Control del Roll

El roll, al igual que el pitch, se controla también con un PID. Al tener, en teoría (ver 3.2 Modelo Teórico de un Quadrotor) la misma dinámica que el pitch, los parámetros deberían ser iguales. No obstante, debido a fenómenos no modelados, existen varias discrepancias entre los valores para cada ángulo.

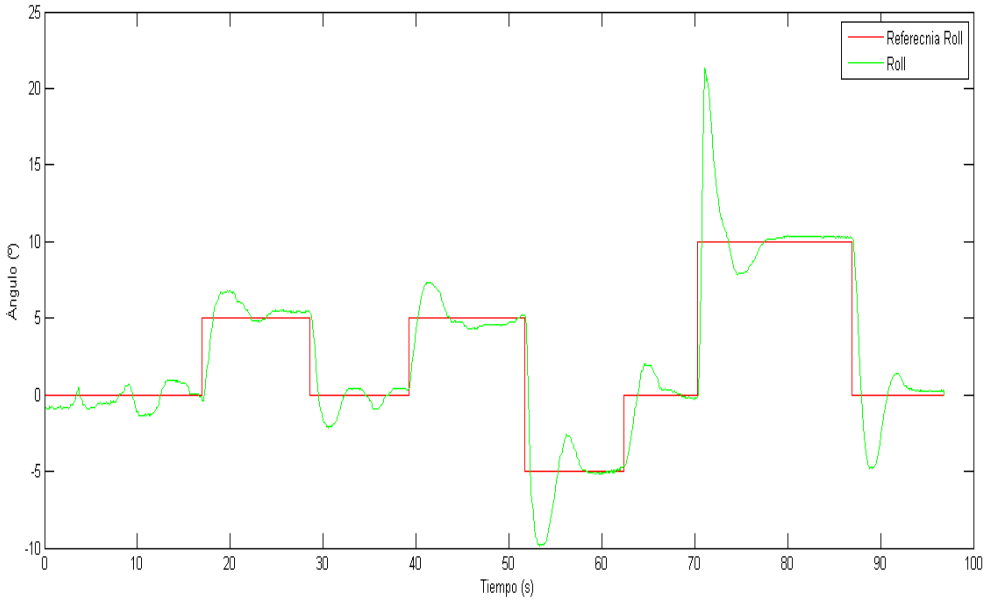

Ilustración 45: Respuesta del roll ante cambio de referencia

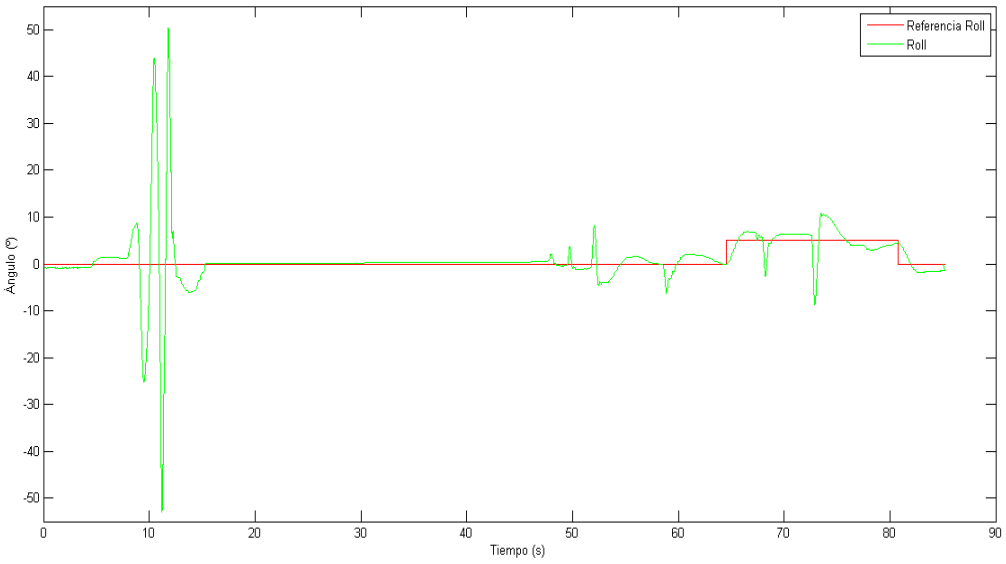

Ilustración 46: Respuesta del roll ante perturbación

El roll tiene una respuesta excelente ante cambios de referencia, logrando error cero (Ilustración 45, segundo 60), o en el peor de los casos, una desviación menor de medio grado. La dinámica de respuesta también es más favorable que en el caso del pitch, corrigiendo más rápido y con una oscilación más amortiguada.

El sistema tiene capacidad de corregir grandes perturbaciones sin problema. Como se observa en la Ilustración 46, en el segundo 10 se producen una secuencia de grandes perturbaciones (los cuatro picos, de hasta 50º), de la cual se recupera rápidamente sin apenas oscilar. Fuera del punto de referencia, la respuesta es también inmediata y se mantiene dentro del medio grado marcado en el ensayo de referencia.

En este caso también se sufre, aunque en menor medida, de la restricción de la rótula, que impide alcanzar el error cero. Como se puede observar en la gráfica de ensayo de referencia, entre los segundos 75 y 85, la respuesta es mantenida, el roll no varía aun habiendo error. Esto es indicio de una acción externa que impide la corrección. De esta manera se determina que es la rótula la causante de esta desviación.

7.5 Control del Yaw

En el control del yaw, que no resulta imprescindible para la estabilidad del quadrotor, se esperaba un desempeño menos satisfactorio, dado el carácter de la medida del mismo (ver 6.3.4 Cálculo del pitch, roll y yaw). No obstante, los resultados obtenidos superan las expectativas y demuestran un control más efectivo del esperado. Al no ser evitar inestabilidades, se prescinde del integrador y se implementa un controlador PD.

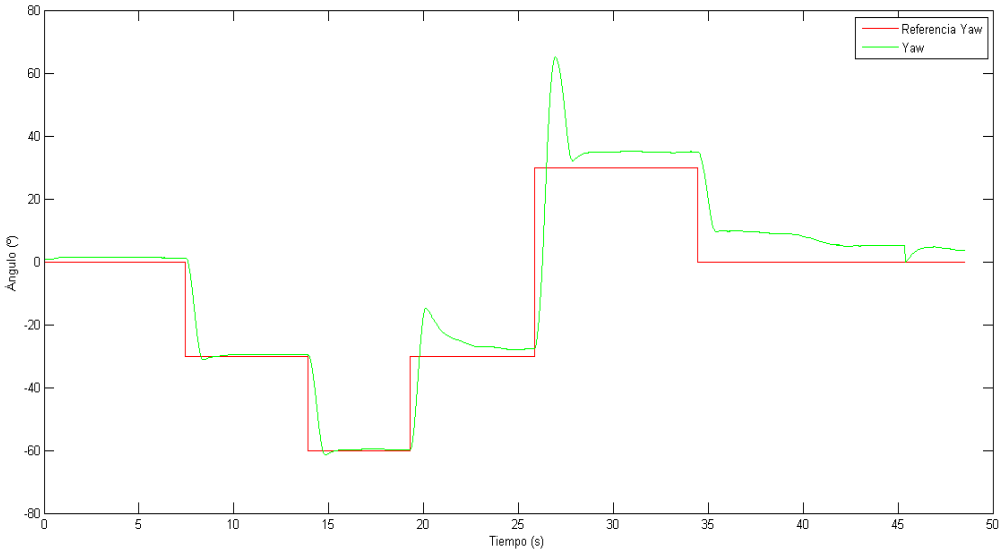

Ilustración 47: Respuesta del yaw ante cambios de referencia

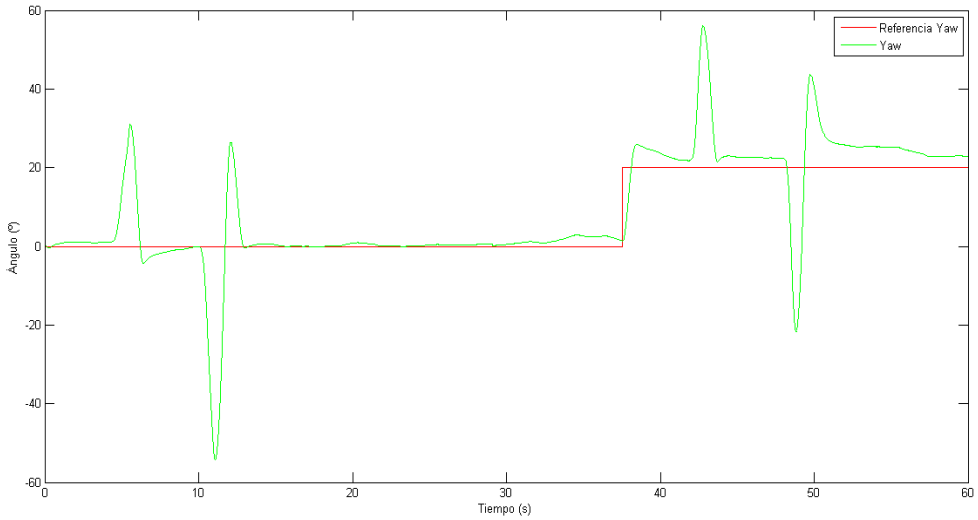

Ilustración 48: Respuesta del yaw ante perturbación

Como se puede observar en la Ilustración 47, en los primero cambios de referencia, la respuesta es rápida y sin sobresocilación (gracias a la ausencia de integrado), y además con un error muy bajo para el ángulo desplazado. No obstante, a medida que se suceden los giros, la precisión decae y muestra algo más de oscilación, pero realiza un seguimiento aceptable de la referencia.

Por otro lado, la respuesta ante perturbaciones en el punto de equilibrio (Ilustración 48, segundos 8 y 11) es rápida y con buena precisión. Al variar la referencia, la respuesta dinámica se mantiene pero presentar mayor desviación. Esto se debe, de nuevo, a la baja tolerancia de la rótula en pequeños movimientos.

8 CONCLUSIONES Y TRABAJOS FUTUROS

8.1 Conclusiones

A lo largo de este documento, se han detallado los objetivos y los problemas que derivan del alcance de estos, al igual que las soluciones adoptadas y las impresiones obtenidas. En este punto se pretende recoger todos estos resultados y analizar el cumplimiento de los objetivos marcados.

Se he logrado dotar al mini ordenador Raspberry Pi de un sistema operativo de tiempo real que cumple los requerimientos de respuesta y tiempo de ejecución que son necesarios para el control efectivo del quadrotor.

El sistema tiene capacidad de determinar su orientación en los tres ángulos (pitch, roll y yaw) gracias a la unidad de medida inercial que incluye el quadrotor. Además, también se ha logrado realizar el cálculo de la altura relativa y, aunque no ha logrado cumplir los requisitos del rango de trabajo, resulta lo suficientemente buena para vuelos en exterior donde el rango de altura es más permisivo.

Todo el sistema de control se integra en única placa, con el fin de reducir al máximo las fuentes de error y los tiempos de respuesta. La posibilidad de generar señales PWM desde la propia Raspberry ha permitido eliminar un nivel que hasta el momento siempre había estado presente.

El programa se comunica en este caso con dos sensores, ambos a través del mismo bus. No obstante, se ha desarrollado para que sea lo suficientemente versátil para habilitar nuevos canales de comunicación o integrar nuevos sensores que permitan ampliar las funcionalidades del programa. Además, se ha conseguido controlar de forma remota a través del ordenador en tierra, permitiendo ajustar los parámetros de control y cambiar el modo de funcionamiento.

Aunque el logro más importante ha sido el desarrollo efectivo de un software capaz de controlar en todo momento la orientación del quadrotor, garantizando la estabilidad en todo momento y permitiendo un vuelo constante. La puesta a prueba ha permitido observar la respuesta del sistema en vuelo, con unos resultados satisfactorios. El programa es capaz de controlar los tres ángulos simultáneamente, responder a los cambios de referencia y corregir correctamente las perturbaciones externas. La posibilidad de estudiar el vuelo de forma segura gracias a la plataforma de ensayo en los tres grados de libertad al mismo tiempo ver el comportamiento del quadrotor en una situación más real. Por tanto, ha quedado probada la capacidad del mini-PC Raspberry Pi 2 de realizar el control efectivo de orientación de un quadrotor.

Este proyecto se ha presentado como un reto multidisciplinar, que me ha permitido poner a prueba las capacidades y conocimientos que he obtenido a lo largo de estos años. Gracias al trabajo desarrollado, he podido entender y ver de primera mano cómo funciona un quadrotor, además de reforzar mis conocimientos en área como la informática, la electrónica y el control, o incluso en área menos relacionadas como las matemáticas y la probabilidad.

8.2 Trabajos Futuros

En este tipo de proyectos, el margen de mejora es inmenso, siempre habiendo la posibilidad de mejorar el rendimiento o incorporando nuevas funcionalidades. No obstante, al partir de cero en el desarrollo del trabajo, se deben fijar unos objetivos realistas. Tras alcanzar los objetivos revistos, se pueden plantear unos nuevos propósitos, que podrían llevarse a cabo en futuros trabajos:

- Obtener una medida más exacta del yaw. Puede aprovecharse el magnetómetro que incluye la IMU para corregir el ángulo en vuelos exteriores.

- Probar nuevos algoritmos de control. El programa usa un PID en paralelo, no obstante, pueden implementarse técnicas más avanzadas de control que podrían mejorar el comportamiento.

- Mejorar la precisión de la medida de altura. Aunque la medida es buena para vuelos de cierta altura, se requiere mejor precisión para momentos críticos como aterrizaje y despegue. Se podría estudiar la inclusión de otro sensor, como un sensor de ultrasonidos, para que trabaje en ese rango.

- Sustitución de la Raspberry Pi 2 por el nuevo modelo Raspberry Pi 3, el cual ya incluye la conexión Wifi, lo que reduciría las fuentes de error, además de contar con más potencia de procesamiento.

- Incluir el control de posición x-y. Para esto podría usarse señal GPS o un sensor óptico que permitiese el posicionamiento tridimensional del quadrotor

- Mejora de la plataforma de ensayo. Reducir la rigidez que presenta la rótula y que adultera las medidas o la sustitución de la misma por otro sistema de rotación. Adicionalmente, podría estudiarse la posibilidad de medir los ángulos de orientación desde la plataforma, con el fin de calibrar la IMU del quadrotor.

BIBLIOGRAFÍA Y REFERENCIAS

[1] RT, «RT - Sepa más,» 7 Diciembre 2012. [En línea]. Available: https://actualidad.rt.com/actualidad/view/80396-vehiculos-aereos-tripulados-hitos-historicos.

[2] U. d. V. -. A. y. O. c. D. -. D. Comunicación, «UV Drones,» 9 Julio 2015. [En línea]. Available: http://drones.uv.es/origen-y-desarrollo-de-los-drones/.

[3] A. Bernardo, «BlogThinkBig,» 15 Enero 2015. [En línea]. Available: http://blogthinkbig.com/la-ciencia-en-guerra/.

[4] N. G. Pandavenes, «infoDron,» 24 Mayo 2016. [En línea]. Available: http://www.infodron.es/id/2016/05/24/noticia-coparan-mercado-aeronautico.html?utm_source=twitterfeed&utm_medium=twitter.

[5] G. Vianna Raffo, Modelado y Control de un Helicóptero Quadrotor, Sevilla, 2007.

[6] I. PE, «Comohacer.eu ¿inventamos juntos?,» 12 Agosto 2014. [En línea]. Available: http://comohacer.eu/comparativa-y-analisis-raspberry-pi-vs-competencia/.

[7] DroTek, «Drotek Electronics,» [En línea]. Available: http://www.drotek.com/shop/en/home/62-imu-10dof-mpu6050-hmc5883-ms5611.html.

[8] IvenSense Inc., «Olimex - MPU60xx Register Map,» 3 Octubre 2012. [En línea]. Available: https://www.olimex.com/Products/Modules/Sensors/MOD-MPU6050/resources/RM-MPU-60xxA_rev_4.pdf.

[9] MEAS Switzerland, «Measurements Specialities - MS5611 Datasheet,» 26 Octubre 2012. [En línea]. Available: http://www.meas-spec.com/downloads/MS5611-01BA03.pdf.

[10] Siagel, «Comerciosendolores,» 2010. [En línea]. Available: http://www.comerciosendolores.com/catalogo%20siagel%20med.pdf.

[11] HobbyKing, «HobbyKing - Turnigy Basic 25A,» [En línea]. Available: http://www.hobbyking.com/hobbyking/store/__3731__TURNIGY_Basic_25amp_Speed_Controller_w_BEC.html.

[12] Comunidad Wikibooks, «Wikilibros,» 27 Noviembre 2015. [En línea]. Available: https://es.wikibooks.org/wiki/Redes_inform%C3%A1ticas/Protocolos_TCP_y_UDP_en_el_nivel_de_transporte.

[13] CCM, «CCM Comunidad Informática,» Abril 2016. [En línea]. Available: http://es.ccm.net/contents/651-sistema-operativo.

[14] freeRTOS, «freeRTOS - What is an RTOS?,» [En línea]. Available: http://www.freertos.org/about-RTOS.html.

[15] freeRTOS, «freeRTOS - Tasks & Co-routines,» [En línea]. Available: http://www.freertos.org/RTOS-task-states.html.

[16] freeRTOS, «freeRTOS - Binary Semaphores,» [En línea]. Available: http://www.freertos.org/Embedded-RTOS-Binary-Semaphores.html.

[17] freeRTOS, «freeRTOS - Mutexes,» [En línea]. Available: http://www.freertos.org/Real-time-embedded-RTOS-mutexes.html.

[18] AMSYS, «AMSYS - Precise height measurement (24 bit) with pressure sensor MS5607,» Abril 2010. [En línea]. Available: http://www.amsys.info/sheets/amsys.en.aan509_e.pdf.

[19] Comunidad Wikipedia, «Wikipedia, Enciclopedia libre - Filtro de Kalman,» 24 Mayo 2016. [En línea]. Available: https://es.wikipedia.org/wiki/Filtro_de_Kalman.

[20] M. Lázaro, «Universidad Carlos II de MAdrid - Dpto. de Teoría de la Señal y Comunicaciones,» 2003. [En línea]. Available: http://www.tsc.uc3m.es/~mlazaro/Docencia/Doctorado/FiltAdapt/Kalman.pdf.

[21] G. Przemyslaw, S. Gardecki, J. Goslinski y W. Giernacki, «Estimation of Altitude and Vertical Velocity forMultirotor Aerial Vehicle using Kalman Filter,» de *Recent Advances in Automation, Robotics and Measuring Techniques*, Poznan, Polonia, Springer International Publishing, 2014, pp. 377-385.

[22] National Instruments Corporation, «National Instruments - PID Theory Explained,» 29 Marzo 2011. [En línea]. Available: http://www.ni.com/white-paper/3782/en/.

DISEÑO DE UNA PLATAFORMA SOFTWARE PARA EL CONTROL DE ORIENTACIÓN 3D DE UN SISTEMAQUADROTOR BASADO EN RASPBERRY PI 2

PRESUPUESTO

1 Introducción

Con el objetivo de realizar un presupuesto más realista, se ha contemplado tanto el desarrollo del software llevado a cabo en el presente trabajo, como el diseño y construcción del quadrotor y la plataforma de ensayo. Por tanto, se ha trabajado con el compañero encargado de dicha parte en la composición de este presupuesto.

Los precios que se muestran a continuación se han obtenido de las facturas de compra o, en caso de no disponer de estas, cotejando con un artículo lo más parecido al original. Los materiales disponibles en el laboratorio, tales como cables y tornillería, y que se han usado en el desarrollo del trabajo, se han ajustado en base a lo artículos disponibles en el mercado. Por último, el precio por hora de las herramientas y aparatos usados se ha calculado en base al precio total y el tiempo de vida útil esperado.

Las horas de mano de obra, dos graduados en tecnologías industriales, un técnico de laboratorio y un carpintero se han realizado de forma aproximada, teniendo en cuenta los días trabajados en el laboratorio y las horas dedicadas al día. También se ha contemplado el desarrollo de software llevado a cabo fuera del laboratorio y el tiempo dedicado a adquirir los conocimientos necesarios para la consecución del trabajo

El presupuesto se divide en cuatro capítulos, que son los que se enuncian a continuación:

- Diseño: cuenta con dos unicidades de obra; una dedicada al diseño y planteamiento del quadrotor; y otra para el diseño y desarrollo de la plataforma de ensayo. En las horas dedicadas también se incluye el tiempo empleado en el estudio de alternativas para la plataforma

- Construcción: dividido en cuatro unidades de obra; una referente a la realización de las pertinentes conexiones eléctricas del quadrotor y acoplamiento de los componentes electrónicos; otra para el montaje la estructura del quadrotor y ensamblaje de los elementos electrónicos; una tercera dedicada al conformado de las piezas que constituyen la plataforma y una última para la construcción de esta.

- Software: constituido por una única unidad de obra, se contempla el desarrollo de todo el software encargado de realizar el control del quadrotor. En las horas se incluye el tiempo dedicado a adquirir los conocimientos sobre el funcionamiento del quadrotor y del funcionamiento de los componentes con los que se trabaja.

- Ensayo: el la unidad de obra de este capítulo se contempla el ensayo y estudio del desempeño del software de control, de la habilidad del quadrotor de realizar un vuelo estable y la capacidad de la plataforma de permitir las pruebas de vuelo.

2 Cuadro de Precios Descompuestos

2.1 Capítulo 1: Diseño

UO.1.01 Diseño del quadrotor

Código	Descripción	Ud.	Rdto.	Precio	Importe
MO GITI	Graduado en Tecnologías Industriales	h	90	25,00	2250,00
MAQ OP	Ordenador portátil	h	75	3,00	225,00
	Costes directos complementarios	%	0,01	2.475,00	24,75
	Costes indirectos	%	0,02	2.499,75	50,00
	Precio total UO				2.549,75 €

UO.1.02 Diseño de la base de pruebas

Código	Descripción	Ud.	Rdto.	Precio	Importe
MO GITI	Graduado en Tecnologías Industriales	h	50	25,00	1250,00
MAQ OP	Ordenador portátil	h	30	3,00	90,00
	Costes directos complementarios	%	0,01	1.340,00	13,40
	Costes indirectos	%	0,02	1.353,40	27,07
	Precio total UO				1.380,47 €

2.2 Capítulo 2: Construcción

UO.2.01 Adecuación de conexiones electrónicas y eléctricas

Código	Descripción	Ud.	Rdto.	Precio	Importe
MAT CAC	Conector alta capacidad T macho	Ud	1	0,90	0,90
MAT CBO	Conector batería oro M+H 3,5mm	Ud	24	0,96	23,04
MAT CCP	Conector para cable de potencia	Ud	2	0,30	0,61
MAT USB	Cable micro USB	Ud	1	3,75	3,75
MAT RET	Tubo retráctil P/Baterías 45mm	Ud	1	2,08	2,08
MAT REG	Regleta 6mm transparente	Ud	2	0,12	0,23
MAT CPT	Cable de potencia	m	1	2,32	2,32
MAT CCM	Cable de comunicación	Ud	4	0,09	0,38
MAT EST	Bobina de estaño	Ud	1	4,54	4,54
MAQ SSN	Soldador de estaño	h	4,5	11,90	53,55
MAQ FA	Fuente de alimentación de 40A	h	10	1,47	14,70
MAQ OSC	Osciloscopio	h	106,5	6,00	639,00
MO GITI	Graduado en Tecnologías Industriales	h	40	25,00	1.000,00
MO TEC	Técnico de laboratorio	h	4	15,00	60,00
	Costes directos complementarios	%	0,01	1.805,10	18,05
	Costes indirectos	%	0,02	1.823,15	36,46
	Precio total UO				1.859,61 €

UO.2.02 Montaje del quadrotor

Código	Descripción	Ud.	Rdto.	Precio	Importe
MAT RP2B	Raspberry Pi 2 Modelo B	Ud	1	40,50	40,50
MAT RPA	Raspberry Pi to Arduino Shields Connection Bridge	Ud	1	40,00	40,00
MAT IMU	IMU 10DOF (MPU6050 + HMC5883 + MS5611)	Ud	1	18,90	18,90
MAT CT	Conversor de tensión	Ud	1	4,21	4,21
MAT SD	Tarjeta Micro-SD 16GB	Ud	1	5,55	5,55
MAT WF	Wifi USB	Ud	1	16,36	16,36
MAT MOT	Motor ele. Brushless Trainer 1500	Ud	4	26,60	106,40
MAT ESC	TURNIGY Plush 25amp Speed Controller w/BEC	Ud	4	11,97	47,88
MAT CW	Turnigy Slowfly Propeller 8x4.5 Black (CW) (2pcs)	Ud	2	1,09	2,18
MAT CCW	Turnigy Slowfly Propeller 8x4.5 Black (CCW) (2pcs)	Ud	2	1,09	2,18
MAT CHA	Chasis de mikrokopter mk40	Ud	1	62,05	62,05
MAT PCC	Placa Central para Chasis MK-30, 40 o 50	Ud	2	6,70	13,40
MAT POL	Poliestireno expandido 1m2	Ud	1	2,00	2,00
MAT BPN	Bote de pintura negra	Ud	1	1,50	1,50
MAT BPV	Bote de pintura verde	Ud	1	1,50	1,50
MAT IN	Imán de neodimio N52 8 x 1mm	Ud	8	0,12	0,96
MAT TUE	Tuerca DIN EN 24032 M2,5	Ud	24	0,03	0,72
MAT TOR20	ISO 7045 H M2,5 x 20 - 4.8 - H	Ud	9	0,04	0,36
MAT TOR8	ISO 7045 H M2,5 x 8 - 4.8 - H	Ud	2	0,03	0,06
MAT E15	Espaciadores 5.6mm x 15mm M2,5 Nylon	Ud	10	0,08	0,84
MAT E10	Espaciadores 5.6mm x 10mm M2,5 Nylon	Ud	12	0,08	0,94
MAT PBQ	Placa de baquelita 10 x 10cm	Ud	1	13,30	13,30
MAT PEG	Pegamento instantáneo (50ml)	Ud	1	2,00	2,00
MAT CAD	Cinta aislante	Ud	1	2,30	2,30
MAQ FA	Fuente de alimentación 40A	h	3	1,47	4,41
MAQ KIT	Kit de herramientas	h	15	0,53	7,95
MAQ TAL	Taladradora	h	1	2,10	2,10
MO GITI	Graduado en Tecnologías Industriales	h	30	25,00	750,00
MO TEC	Técnico de laboratorio	h	1	15,00	15,00
	Costes directos complementarios	%	0,01	1.165,55	11,66
	Costes indirectos	%	0,02	1.177,20	23,54
	Precio total UO				1.200,75 €

UO.2.03 Adecuación de las piezas de la base de pruebas

Código	Descripción	Ud.	Rdto.	Precio	Importe
MAT MDF	Tablero MDF 560x400x10mm	Ud	1	4,60	4,60
MAT LCA	Listón cepillado abeto 56x520mm	Ud	1	6,25	6,25
MAT TA45	Tornillo avell poz a.cromo cr3 4x45	Ud	4	0,07	0,29
MAT PTN	Bote de pintura negra	Ud	1	1,50	1,50
MAQ SRB	Sierra radial de banco	h	0,15	1,50	0,23
MAQ FM	Fresadora para madera	h	0,5	2,00	1,00
MO CAR	Carpintero	h	1	10,00	10,00
MO GITI	Graduado en Tecnologías Industriales	h	4	25,00	100,00
	Costes directos complementarios	%	0,01	123,86	1,24
	Costes indirectos	%	0,02	125,10	2,50
	Precio total UO				127,60 €

UO.2.04 Construcción de la base de pruebas

Código	Descripción	Ud.	Rdto.	Precio	Importe
MAT TA20	Tornillo avell poz a.cromo cr3 4,5x20	Ud	8	0,05	0,38
MAT TA30	Tornillo avell poz a.cromo cr3 4x30	Ud	12	0,06	0,69
MAT APA	Arandela plana ancha a.zinc D.5	Ud	20	0,05	0,92
MAT EAI	Escuadra ángulo inox 60x60mm	Ud	4	0,75	3,00
MAT TC	Tubo de carbono 8x6x1000mm	Ud	1	12,40	12,40
MAT VC	Varilla de carbono 2,5x1000	Ud	1	2,70	2,70
MAT ROT	Rótula Igubal GFSM-08-IG	Ud	1	6,84	6,84
MAT PFV	Placa de fibra de vidrio 1,5x100x100mm	Ud	1	2,63	2,63
MAT TOR12	ISO 7045 H M2,5 x 12 - 4.8 - H	Ud	4	0,04	0,14
MAT TUE	Tuerca DIN EN 24032 M2,5	Ud	4	0,03	0,12
MAT PE	Pegamento epoxi 5 min 128g	Ud	1	11,49	11,49
MAQ SIE	Sierra	h	0,25	0,80	0,20
MAQ KIT	Kit de herramientas	h	1,5	0,53	0,80
MAQ TAL	Taladradora	h	0,5	2,10	1,05
MO GITI	Graduado en Tecnologías Industriales	h	5	25,00	125,00
	Costes directos complementarios	%	0,01	168,35	1,68
	Costes indirectos	%	0,02	170,04	3,40
	Precio total UO				173,44 €

2.3 Capítulo 3: Software

UO.3.01 Desarrollo del software

Código	Descripción	Ud.	Rdto.	Precio	Importe
MAQ FA	Fuente de alimentación de 40A	h	40	1,47	58,80
MAQ OP	Ordenador portátil	h	80	3,00	240,00
MAQ OS	Ordenador de sobremesa	h	100	3,45	345,00
MO GITI	Graduado en Tecnologías Industriales	h	200	25,00	5000,00
					0,00
					0,00
					0,00
	Costes directos complementarios	%	0,01	5643,80	56,44
	Costes indirectos	%	0,02	5700,238	114,00
	Precio total UO				5.814,24 €

2.4 Capítulo 4: Ensayos

UO.4.01 Ensayos del quadrotor

Código	Descripción	Ud.	Rdto.	Precio	Importe
MAQ FA	Fuente de alimentación de 40A	h	45	1,47	66,15
MAQ OP	Ordenador portátil	h	60	3,00	180,00
MAQ OS	Ordenador de sobremesa	h	60	3,45	207,00
MO GITI	Graduado en Tecnologías Industriales	h	60	25,00	1500,00
	Costes directos complementarios	%	0,01	1953,15	19,53
	Costes indirectos	%	0,02	1972,68	39,45
	Precio total UO				2.012,14 €

3 Presupuesto Parcial

Capítulo 1. Diseño	
UO.1.01 Diseño del quadrotor	2.549,75 €
UO.1.02 Diseño de la base de pruebas	1.380,47 €
Total Capítulo	3.930,21 €
Capítulo 2. Construcción	
UO.2.01 Adecuación de conexiones electrónicas y eléctricas	1.859,61 €
UO.2.02 Montaje del quadrotor	1.200,75 €
UO.2.03 Adecuación de las piezas de la base de pruebas	127,60 €
UO.2.04 Construcción de la base de pruebas	173,44 €
Total Capítulo	3,.61,40 €
Capítulo 3. Software	
UO.3.01 Desarrollo del software	5.814,24 €
Total Capítulo	5.814,24 €

Capítulo 4. Ensayos		
UO.4.01 Ensayos del quadrotor		2.012,14 €
	Total Capítulo	2,012,14 €
	Total	**15,117,99 €**

4 Presupuesto de Ejecución Material, Presupuesto por Contrata y Presupuesto Base de Licitación

PRESUPUESTO DE EJECUCIÓN MATERIAL	15.117,99 €
15% Gastos generales	2.267,70 €
6% Beneficio industrial	907,08 €
PRESUPUESTO DE EJECUCIÓN POR CONTRATA	18.292,76 €
21% IVA	3.841,48 €
PRESUPUESTO BASE DE LICITACIÓN	**22.134,24 €**

Asciende el presente presupuesto base de licitación a la cantidad de:

VEINTIDÓS MIL CIENTO TREINTA Y CUATRO EUROS CON VENTICUATRO CÉNTIMOS.