UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño
Máster Universitario en Ingeniería Aeronáutica

Trabajo Fin de Máster

CONTROL AUTOMÁTICO CON SIMULINK DE UN MINI DRON. APLICACIÓN AL SEGUIMIENTO DE TRAYECTORIAS MEDIANTE REALIMENTACIÓN VISUAL

Autor:
Martos Bianqui, Juan

Tutor:
Ángel Rodas Jordá

- Valencia, 2020/2021 -

Agradecimientos

Primero quiero agradecer a todos los profesores con los que he trabajado a lo largo de toda mi experiencia universitaria, algunos los recordaré y otros no, pero espero que los conocimientos que intentaron trasmitirme perduren. En especial a mi tutor, Ángel Rodas Jordá por su dedicación y por estar siempre a mi disposición.

Agradezco de manera inconmensurable a mis amigos, en especial a Miguel García Andrés y a Juan Diego Marín Re. Gracias por hacer de estos últimos años una aventura de lo que de otra forma hubiese sido un tormento.

Finalmente, gracias a toda mi familia por su apoyo constante e incondicional.

A mis padres, por ofrecerme todo lo que una persona puede ofrecer, sin ellos no sería nada.

Resumen

El presente trabajo fin de máster de ingeniería aeronáutica estudia las posibilidades de control del mini dron Parrot Mambo mediante el entorno de programación Matlab/Simulink. Dicha plataforma destaca frente a otras alternativas ya que admite la modificación del *firmware* del propio dron y, por lo tanto, el acceso a su controlador de vuelo, aun siendo un dron comercial de bajo coste.

Este hecho permite múltiples posibilidades a través del ajuste de los algoritmos de control, así como el acceso a la lógica interna que gobierna el vuelo del Parrot Mambo, permitiendo ampliarla o mejorarla. Como resultado se desarrolla una aplicación que hace uso de la realimentación visual proporcionada por la cámara cenital del dron y su procesamiento embebido con el objetivo de realizar el seguimiento de trayectorias marcadas en el suelo.

Abstract

The present final thesis of the master's degree in aeronautical engineering studies the control possibilities of a minidrone Parrot Mambo through the programming environment Matlab/Simulink. Said platform outlines its alternatives since allows the modification of the firmware, therefore granting acces to the flight controller, even though it is a low-cost mini drone.

This fact permits multiples possibilities through the adjustment of the control algorithms just as the internal logic that commands the flight of the Parrot Mambo, allowing improvements and extensions. As a result an application for trajectory tracking is developed using visual feedback provided by the drone's camera.

Resum

El present treball fi de màster d'enginyeria aeronàutica estudia les possibilitats de control del mini dron Parrot Mambo per mitjà de l'entorn de programació Matlab/Simulink. La dita plataforma destaca enfront d'altres alternatives ja que admet la modificació del *firmware* del propi dron i, per tant, l'accés al seu controlador de vol, encara sent un dron comercial de baix cost.

Este fet permet múltiples possibilitats a través de l'ajust dels algoritmes de control, així com l'accés a la lògica interna que governa el vol del Parrot Mambo, permetent ampliar-la o millorar-la. Com resultat es desenrotlla una aplicació que fa ús de la realimentació visual proporcionada per la cambra zenital del dron i el seu processament embegut amb l'objectiu de realitzar el seguiment de trajectòries marcades en el sòl.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción y objetivos

1.1 Introducción y motivación

Los avances en electrónica como la miniaturización, el aumento en la capacidad de computación y la mejora de las conexiones inalámbricas entre otros han permitido el desarrollo de un nuevo tipo de aeronaves: los UAS (*Unmanned Aerial System*). Popularmente conocidos como *drones*, su industria ha tenido un crecimiento en los últimos años casi exponencial.

Además, la industria de los drones suele hacer uso de tecnologías muy novedosas. El uso de la realimentación visual para el guiado o el diseño de sistemas autónomos son algunas de las tendencias que producen mayor interés tanto a nivel usuario como empresarial. Por ello, el conocimiento de estas tecnologías junto con sus principios de funcionamiento resulta una evidente motivación de cara al futuro desarrollo de la industria.

1.2 Objetivos

Dada esta motivación, para el presente trabajo de fin de Máster se planteó como objetivo principal estudiar las posibilidades de control haciendo uso de drones comerciales y, más en concreto, se buscaba la capacidad de trabajar con su lógica interna y sus algoritmos. Así, se obtendría una visión global de su funcionamiento, desde sus aspectos más superficiales hasta los más profundos.

Con esto en mente surgieron una serie de objetivos menores que, en su conjunto, sirviesen de conductores para alcanzar el propósito general. De esta forma, el cumplimiento de estos subobjetivos serviría para desarrollar una aplicación concreta que represente el fruto del trabajo y las intenciones expuestas.

- Estudiar el estado del arte. Elegir el dron y la plataforma de trabajo en concreto.

- Conocer los principios físicos detrás del vuelo de un dron a través del problema de control.
- Estudiar el modelo de simulación de un cuadricóptero para comprender todos los elementos necesarios para su funcionamiento.
- Diseñar los parámetros de control que componen el bucle de control a través de simulaciones y ensayos.
- Desarrollar una aplicación al seguimiento de trayectorias mediante realimentación visual.

1.3 Estructura de la memoria

El presente trabajo se estructura entorno a capítulos. A continuación, se ofrece una pequeña descripción de cada uno de ellos.

Primero, en el Capítulo 2 se abordará el estado del arte de los drones donde se discutirá sobre su denominaciones, clasificación y características. También se tratará su desarrollo y potencial futuro junto con una pequeña revisión de sus ventajas. Por último, se expondrán algunas aplicaciones donde el uso de los drones es más prominente.

Se hablará más en concreto de los mini drones a los cuales pertenece el dron elegido y se explicará el porqué de su elección frente a otras opciones. Además, se hará una revisión de trabajos previos que hayan tratado con las herramientas de trabajo propuestas.

En el Capítulo 3 se procederá a explicar en profundidad las herramientas con las que se ha trabajado. Para ello se diferenciará entre *hardware* y *software*, en la sección referente al primero se dará una lista de los materiales necesarios, se ofrecerán especificaciones técnicas y se explicarán los sensores embarcados.

En el apartado de *software* se presentará el flujo de implementación de algoritmos de vuelo diseñados, así como una descripción de los instrumentos utilizados. Luego, se mostrará el procedimiento a seguir para establecer conexión entre el PC y el dron.

Después, en el Capítulo 4 se emprenderá el estudio del modelado de vuelo de los cuadricópteros, primero se planteará el problema de control y cuáles son sus principios, así como de los mecanismos que dispone un dron para navegar el espacio tridimensional. También, se realizará un análisis de su modelado matemático no lineal, donde se expondrán las ecuaciones de movimiento.

Seguidamente, en el Capítulo 5 se profundizará en las herramientas principales de trabajo. Se explorará el modelado del dron elegido, así como las herramientas de simulación que permiten realizar un desarrollo basado en su modelo, este concepto también es explorado.

Previo al desarrollo de la aplicación se ha planteado el ajuste de los controladores PID que controlan el vuelo del dron. Por ello, en el Capítulo 6 primero se profundizará en el problema de control y luego se presentarán los mecanismos disponibles para su ajuste. Se mostrarán los resultados simulados y los obtenidos en pruebas sobre el *hardware* para comprobar si se han alcanzado los objetivos de ajuste.

Finalmente, en el Capítulo 7 se presentarán los algoritmos desarrollados para llevar a cabo la aplicación. Se comenzará con el procesado y análisis de la imagen procedente de la cámara para luego mostrar los algoritmos que se encargan de utilizar dicha información para el guiado. Se llevarán a cabo pruebas tanto simuladas como en el *hardware* para poder estudiar sus resultados.

De estos resultados se derivarán unas conclusiones, así como comentarios adicionales sobre posibles líneas de trabajo futuras, problemas y distintas consideraciones.

Capítulo 2

Estado del arte

2.1 Introducción a los vehículos aéreos no tripulados

Los *drones* presentan multitud de nomenclaturas según el caso, si se refiere únicamente al vehículo se suele utilizar el término *Vehicle* como en UAV (*Unmanned Aerial Vehicle*) mientras que el uso del término *System* hace referencia además a la estación en tierra y al enlace de comunicaciones. También es común el uso de RPA (*Remotely Piloted Aircraft*) y RPAS (*Remotely Piloted Aircraft System*) cuando el pilotaje se realiza a distancia. Todos estos nombres hacen referencia en última instancia a lo mismo: vehículos aéreos sin piloto a bordo.

Los drones se presentan de muchas formas distintas y por tanto existen varias formas de clasificarlos.

Clasificación según el tipo de ala:

- Ala fija. El ala fija permite al drones disponer de mayor autonomía, desplazarse a mayores distancias y cubrir áreas mucho más extensas, pero no les permite el vuelo en punto fijo. Se podrían comparar con los aviones.
- Alas rotatorias. Estos drones se sustentan mediante el uso de rotores paralelos al suelo por lo que son comparables con los helicópteros. Su principal ventaja reside en la capacidad de volar en punto fijo, lo cual les permite llevar a cabo aplicaciones que son imposibles sin esta característica.
- Híbrido. Se trata de drones que poseen ala fija y a su vez hacen uso de rotores que pueden ser orientables o no. Son menos comunes, pero cubren nichos específicos donde el uso de ambos tipos de alas es valorado.

Clasificación según el control:

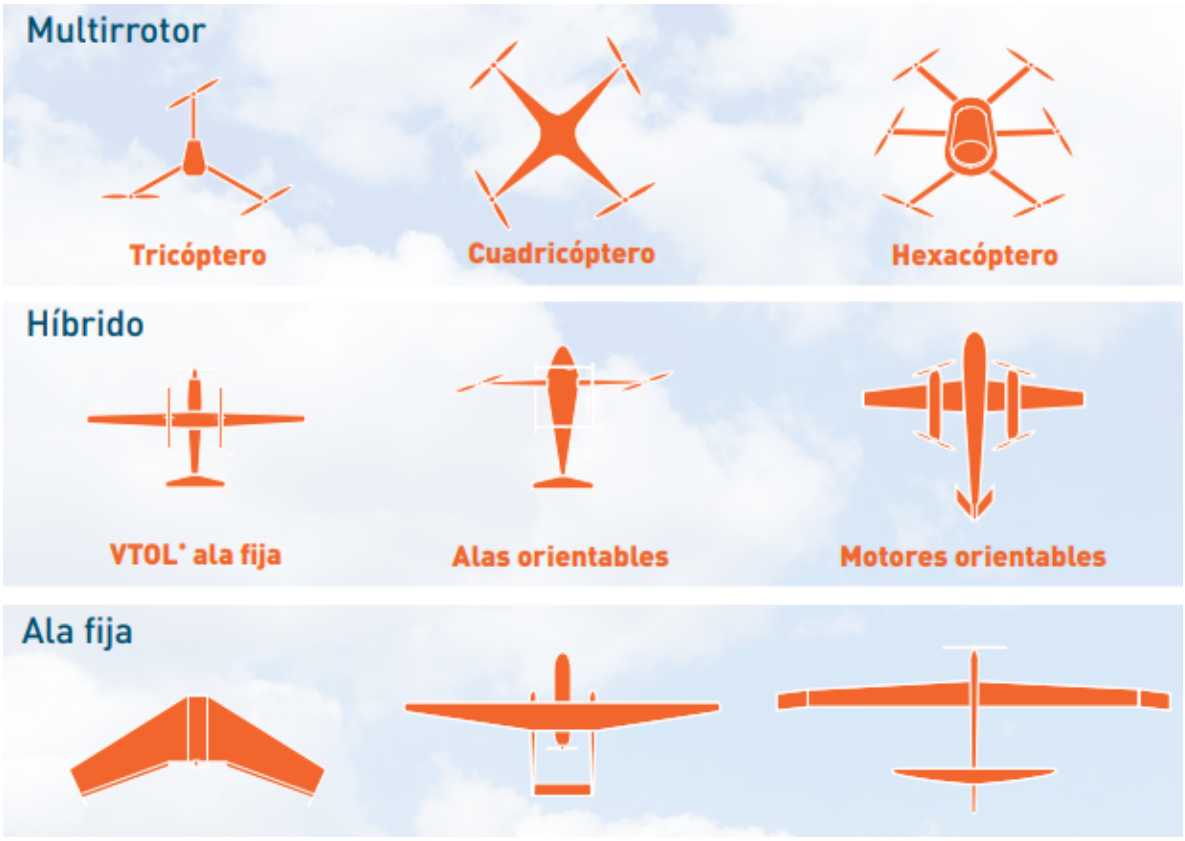


Figura 2.1: Dibujo esquemático donde se muestran distintas tipos de alas (7)

- Autónomo. Se trata de drones que no necesitan piloto ni control directo para poder operar. El control es llevado a cabo por los sistemas internos que, a través de la información recibida mediante los sensores embarcados aplican su lógica interna para llevar cabo la misión.

- Con control remoto. Se trata de los RPAS comentados anteriormente, precisan de un piloto que realiza el control a distancia mediante un dispositivo de control.

- Supervisados. Esta categoría incluye aquellos que no se pueden considerar ni autónomos ni controlados remotamente. Estos drones suelen trabajar de forma autónoma, pero con una monitorización constante de un piloto que puede tomar el control o influir en el comportamiento en tiempo real.

Además, existen otras clasificaciones como según su uso (civil o militar), o según su peso y prestaciones para determinar la licencia pertinente para operar con ellos de forma legal.

2.2 Desarrollo del mercado UAS

El interés en los UAS ha crecido de manera vertiginosa en los últimos años, prueba de ello son los numerosos informes realizados tanto por empresas privadas como entes públicos sobre la evolución y potencial del mercado UAS. *Goldman Sachs* en 2016 publicó el informe *Drones: Reporting for Work* (30) donde los drones se comparaban con internet por su capacidad de convertirse en una poderosa herramienta de negocios, destacando la multitud de aplicaciones donde su adopción iba a producir nuevas oportunidades.

La firma de inversión *Levitate Capital* especializada en movilidad aérea e industria espacial publicó en 2020 el informe *The Future of the Drone Economy* (5) donde se expone el estado del mercado y se realiza un análisis del potencial económico y estratégico de la industria con horizonte 2030. A fecha de 2020 el mercado global de drones está valorado en 14 billones americanos de dólares, divididos en los sector militar (8 billones), consumidor (3 billones) y empresarial (3 billones).

Levitate Capital estima que el tamaño de mercado crecerá hasta los 90 billones en 2030 y el segmento mayoritario será el empresarial a partir de 2024. El segmento empresarial es el menos desarrollado, pero también es el que está teniendo un crecimiento mayor y se espera que siga así hasta 2030. Además, expone que una de las mayores de barreras que ha retrasado el crecimiento a nivel empresarial era la falta de regulación que permitiese el uso de UAS en distintos entornos.

Por otro lado, el Ministerio de Fomento publicó el *Plan estratégico para el desarrollo del sector civil de los drones en España 2018-2021* (7) donde destacan el gran crecimiento que ha tenido este sector en los últimos años, no solo en el mercado global, sino también en el nacional. Destacan la posibilidad de mejorar la competitividad de la industria española mediante la adopción de UAS por multitudes de razones:

- **Disminución de costes**. Una de la mayores ventajas que puede ofrecer el uso de UAS, tanto en el coste de las aeronaves como en los costes operativos si se comparan con la navegación tripulada. Además existen estudios que han intentado cuantificar el ahorro del uso de UAS en distintas aplicaciones, por ejemplo en (16) se concluye que se puede llegar a reducir el coste de las inspecciones en centrales eólicas marinas en un 70 % y reducir la perdida de ingresos en un 90 % por tiempo de inactividad.
- **Mejora de la seguridad laboral**. El uso de UAS evita que el operario tenga que exponerse a entornos peligrosos como zonas contaminadas, incendios o edificios con baja seguridad estructural. Existen distintos estudios que han tratado la mejora de seguridad que se obtiene con el uso de UAS en zonas de construcción como (13) o (15).
- **Reducción del impacto ambiental**. Si se compara con las operaciones de las aeronaves tripuladas, la emisión de contaminantes y la contaminación acústica se

reduce mediante el uso de drones.

- **Mayor flexibilidad de despegue**. El no necesitar infraestructura especifica para aterrizaje y despegue permite al UAS trabajar prácticamente en cualquier lugar.
- **Ampliación de los campos de aplicación**. La mayor ventaja del uso de drones radica en su versatilidad que permite el uso de nuevas aplicaciones donde las plataformas aéreas no se habían contemplado.

No existe mayor prueba del interés que existe por el mercado UAS que el desarrollo de normativa especifica por parte de entes estatales. En Europa se publicó en 2019 los Reglamentos EU 2019/945 y EU 2019/947, conocidos como Reglamento de Ejecución y Delegado respectivamente. En ellos se exponen las nuevas regulaciones aplicables en la Unión Europea con comienzo en 2021, se realizan descripciones de las categorías de operación y requisitos para los operadores, escenarios operacionales, certificados de pilotaje... En resumen, la nueva normativa amplia la base legal en gran medida para operaciones comerciales con UAS que hasta la fecha estaba muy limitadas.

Además la UE también ha aprobado el reglamento U-Space con el cual planean proveer una serie de servicios que crearan una infraestructura diseñada para facilitar el uso de UAS en cualquier tipo de operación comercial y civil. Entre los servicios propuestos destaca el *geofencing* el cual es un perímetro espacial al rededor de un área que delimita el acceso no autorizado a la misma, la gestión de conflictos y la gestión dinámica de la capacidad aérea. La implantación de los servicios esta dividida en 4 fases cada una con su horizonte temporal siendo el último en 2030 con el objetivo final de integrar la aviación tradicional y UAS en el mismo espacio aéreo.

2.3 Aplicaciones

Una de las características que dotan a los drones de gran versatilidad es la variedad de sensores embarcados que pueden portar, esto les confiere un gran rango de prestaciones. Algunas de las aplicaciones donde los UAS son más prominentes son las siguientes:

- **Inspección de infraestructuras y evaluación de daños**. Los UAS permiten llegar zonas de difícil acceso con mayor seguridad y facilidad. Mediante cámaras digitales, térmicas y multiespectrales se pueden detectar grietas, zonas corroídas, desgaste etc. Los lugares donde este servicio es valorado son muchos: estaciones petroleras, tendidos eléctricos, edificios históricos, puentes...
- **Cartografía**. La posibilidad de embarcar láser escáner y sistemas Lidar (*Laser Imaging Detection and Ranging*) los hace perfectos para labores de cartografía y topografía. La obtención de mapas de puntos, modelos digitales de elevación, seguimiento y control territorial son algunos ejemplos.

Figura 2.2: Dibujo esquemático donde se muestran distintas aplicaciones (7)

- **Exploración y explotación de recursos**. La petrolera BP lleva desde 2006 trabajando con UAS (3) para multitud de aplicaciones como la inspección de torres de refrigeración en refinerías o monitorizar equipos de perforación en Alaska.

- **Agricultura de precisión**. Mediante la adquisición de datos con sensores embarcados se puede determinar las condiciones del terreno como el estrés hídrico, el estado de los cultivos o la identificación de plagas. Esto permite una mejora de la productividad.

- **Conservación de especies**. Mediante el uso de cámara térmicas se puede realizar el seguimiento y supervisión de fauna de forma de remota. Por ejemplo, la fundación *Lindbergh* lleva a cabo la iniciativa *Air Shepherd* con la que protegen elefantes y rinocerontes de la caza furtiva en reservas naturales de África mediante el uso de UAS (35).

- **Espectáculos**. Las mejoras en los algoritmos de control permiten hacer uso de los UAS para llevar a cabo espectáculos aéreos. Durante la ceremonia de apertura de los Juegos Olímpicos de Tokyo 2020 se usaron más de 1800 drones coordinados como exhibición.

- **Películas y fotografía**. Los UAS reemplazan el uso de helicópteros y grúas para la toma de imágenes desde cualquier ángulo por la gran reducción de costes que supone.

- **Seguridad y emergencias**. La capacidad de los UAS de alcanzar zonas de difícil acceso los hace perfectos para realizar tareas de búsqueda y rescate, combate

contra incendios o investigación de accidentes entre otros.

2.4 Mini drones

Dado que el objetivo principal del trabajo es estudiar el control de los drones, desde un primer momento quedó claro que el trabajo en ambientes indoor era la mejor opción. Un ambiente interior permite eliminar elementos como el viento que pueden influir en el desempeño del dron y al mismo tiempo constituye un entorno de experimentación más controlado.

Los mini drones son la mejor opción a la hora de trabajar en indoor. Drones de mayor tamaño y potencia sufren de mayores complicaciones en estos ambientes, ya que la interacción de la planta motora con los límites de la habitación puede influir de manera negativa en sus funciones.

Los mini drones son aquellos de menor tamaño disponibles en el mercado, no existe un baremo exacto para determinar esta clasificación, pero se suele usar la posibilidad de colocarse en la palma de la mano. En este caso se consideran mini drones aquellos que se encuentran por debajo de los 600g de peso base.

Destacan por su simplicidad que les permiten tener un coste muy ajustado. Existen gran variedad de ellos, así como de fabricantes, en la Figura 2.1 se muestra una selección de mini drones junto a sus características técnicas.

Como se puede ver se caracterizan por su bajo peso y su coste reducido, fruto de ello disponen de poca autonomía de vuelo. También se observa que casi todos disponen de cámaras que suele ser tipo frontal, aunque existan alternativas. Disponen de conexiones estándar para su control ya sea a través de Wi-Fi o red 2.4GHz.

Debido a los objetivos del trabajo la característica más importante es la posibilidad de programación de su control. Por ello se ha incluido en la lista la categoría de *Programación*, como resultado se observa que solo el Parrot Mambo y el DJI Tello EDU son posibles candidatos.

Que estos drones incluyan la posibilidad de programación en ellos se debe a que algunos fabricantes vieron el uso de los drones como una posibilidad a nivel educativo, elevándolos así por encima de sus competidores. Resulta que el trabajo con drones puede resultar muy didáctico, sirven como punto de iniciación en muchas disciplinas como pueden ser: programación y lógica, procesamiento de imagen, diseño basado en modelos, control automático, aerodinámica y mecánica, electrónica etc.

Debido a esto fabricantes y particulares han desarrollado una variedad de plataformas y aplicaciones que facilitan el trabajo con distintos mini drones. Se pretende hacer una pequeña recopilación de las posibilidades que ofrecen los mini drones DJI Tello EDU y Parrot Mambo al ser estos los más prominentes en este campo.

Nombre	Peso	Tiempo de vuelo	Precio	Cámara	Conexión	Programación
Tech rc Drone	480g	12 min	80€	Frontal	Wi-Fi	No
Potensic Mini Drone	400g	20 min	60€	Frontal	2.4GHz/Wi-Fi	No
Atoyx Mini Drone	180g	7 min	32€	No	2.4GHz	No
RC Drone	540g	15 min	76€	Angular 120º	Wi-Fi	No
Obest Mini Drone	230g	9 min	50€	Frontal	2.4GHz	No
Xiaomi MITU Mini Drone	220g	10 min	70€	Frontal	Wi-Fi	No
Parrot Mambo	63g	8 min	60€	Vertical	Bluetooth	Si
DJI Tello EDU	87g	13 min	110€	Frontal	Wi-Fi	Si

Tabla 2.1: Lista comparativa de mini drones junto a sus características técnicas

2.4.1 DJI Tello EDU

El DJI Tello EDU es una revisión del DJI Tello con la intención de mejorar sus posibilidades a nivel educacional. Tiene un peso de 87 gramos incluyendo batería y cuenta con una cámara frontal HD. Este mini dron se puede programar mediante Scratch, un lenguaje de programación desarrollado por el MIT sencillo y visual basado en bloques enfocado en la educación de niños y adolescentes (32).

Para un control más avanzado dispone de un SDK (*Software Development Kit*) que se conecta mediante Wi-Fi al mini dron permitiendo su control directo mediante co-

Figura 2.3: DJI Tello EDU (34)

Figura 2.4: Parrot Mambo (9)

mandos de texto con el lenguaje de programación Python3. Para comprobar todas las posibilidades que ofrece esta plataforma se puede ver en el manual de usuario (31), el SDK provee una serie de comandos que ordenan ciertas acciones al mini dron así como comandos que solicitan información del mismo. Por ejemplo, el comando *go x y z speed* sirve para indicarle una posición a la que ir y la velocidad deseada.

Lo interesante de este SDK es la posibilidad de realizar *scripts* que son series de comandos con una misión concreta, si además hacemos uso de la cámara frontal se pueden llevar a cabo aplicaciones con realimentación visual.

2.4.2 Parrot Mambo

El Mambo del fabricante francés Parrot forma parte de una línea de mini drones que ofrecen una gran cantidad de opciones para su control. Al igual que con el DJI Tello EDU se puede trabajar mediante Scratch, también existen librerías para la programación mediante Python3 (28) y JavaScript (1) con las que aplicaciones.

Además existe una Toolbox de Matlab llamada *Parrot Minidrones Support from Simulink* (23) que permite diseñar y desarrollar algoritmos de control para minidrones Parrot. Ofrece un control a más bajo nivel ya que permite modificar el firmware del dron, es decir, el ajuste de los algoritmos y de la lógica interna, yendo más allá de las posibilidades que ofrece el DJI Tello.

Además, el equipo de Matlab ha creado una serie de modelos de simulación específicos del Parrot Mambo. Estos que permite realizar el desarrollo de aplicaciones y controles en un ambiente simulado.

Tras conocer las capacidades de ambos drones se decidió elegir el Parrot Mambo para este trabajo. Lo que decantó esta decisión fue las posibilidades de control a bajo nivel mediante la ToolBox de Matlab. Esto permite indagar más en la mecánica de vuelo de los cuadricópteros y no quedarse solo en el desarrollo de aplicaciones.

Sin embargo, la elección del Parrot Mambo como plataforma de *hardware* tiene sus consideraciones. Se ha de tener en cuenta que se está trabajando con un mini dron de

bajo coste. Esto repercute en gran medida en sus capacidades, su conexión es *bluetooth* la cual no permite la emisión de datos entre el PC y el dron en tiempo real debido al limitado ancho de banda. Por ello, todo el procesamiento recae en la propia computación del Parrot Mambo, la cual es limitada aumentando el tiempo de respuesta.

También sus materiales de fabricación y planta motora no es equiparable con drones de precios superiores, por lo que su capacidad de estabilización y la fiabilidad de sus instrumentos es limitada. Estas consideraciones sirven para poner en relieve las capacidades del Parrot Mambo.

2.5 Trabajos previos

El uso del Simulink junto al Parrot Mambo ya ha sido estudiado por otros autores. Tanto a nivel universitario como de investigación se han realizado publicaciones, en esta sección se pretende mostrar algunas de ellas para conocer el trabajo previo. Se comentará de cada una aquellas cosas que han sido de utilidad para el planteamiento de este trabajo.

Diversos TFG/TFM han utilizado el Parrot Mambo junto con la *Toolbox Parrot Minidrones Support from Simulink* como objeto de estudio.

Ángel Bello en (36) exploró el ajuste de controladores mediante ensayos en busca de una mejora en la estabilidad de vuelo, solamente utilizó el método de *prueba y error* y no trabajó en la guiñada. Sin embargo, mostró que mediante diseño de controladores se podía mejorar el desempeño del dron.

Marcos Roa en (11) investigó las posibilidades de usar el Parrot Mambo en un ambientes formativos, como podrían ser asignaturas de universidad. También elaboró sobre los problemas de trabajar con estas herramientas y dio indicaciones a futuros trabajos como este.

Daniel García en (14) estudió el posicionamiento y seguimiento de trayectorias, elaboró en las limitaciones de la sensorización embarcada del Parrot Mambo. Asimismo, estudió la posibilidad de realizar el procesamiento de imagen mediante el ordenador y enviar la información en tiempo real al dron para el guiado. Sin embargo, concluyó que esta no era una metodología válida debido al retraso en el envío de información mediante *bluetooth*.

Finalmente, Gallo-Sanabria *et al.* exploraron seguimiento autónomo de trayectorias mediante visión artificial en (12). Concluyeron que era posible alcanzar el vuelo autónomo incluyendo el aterrizaje. Su trabajo ha servido de apoyo a la hora de desarrollar algoritmos de control y para el procesamiento de imagen.

Estos trabajos previos permitieron conocer las posibilidades y los límites de la plataforma de trabajo propuesta, facilitando así el desarrollo de este trabajo.

Capítulo 3

Materiales

Como ya se ha explicado previamente, las principales plataformas de trabajo que se van a utilizar son el Parrot Mambo en cuanto a *hardware* y Matlab/Simulink en *software*. En este capítulo se pretende profundizar en las herramientas utilizadas.

3.1 Hardware

El Parrot Mambo ya ha sido presentado, en este apartado se mostrarán sus características técnicas en profundidad. También, se mostrarán los sensores embarcados junto a sus funcionalidades, así como de su sistema de coordenadas y la disposición de los rotores.

Sus características técnicas son las siguientes

- Peso: 63.5 gramos
- Dimensiones: 13.21 x 13.21 x 4.06 centímetros
- Batería de polímero de litio
- Autonomía de 8 minutos de vuelo

La placa base del dron es una SIP6 con 800MHz ARM A9 que usa como sistema operativo Linux. Esta placa base hace de unidad de procesamiento y también sirve como base para la sensorización embarcada. Dicha sensorización es la siguiente:

- **IMU**. Unidad de medición inercial (*Inertial Measurement Unit*). Mide las velocidades angulares y aceleraciones lineales mediante giroscopios y acelerómetros. Se utiliza para estimar la posición y orientación del dron de forma precisa (10).

Figura 3.1: Parrot Mambo (2)

- **Sonar(Sound Navigation and Ranging)**. Se utiliza para estimar la altura a la que se encuentra el dron. El sonar es capaz calcular la distancia a un obstáculo mediante el cómputo del tiempo de viaje de ida y vuelta de las ondas emitidas por el propio sonar.
- **Cámara vertical 60 fps**. Hace uso de una metodología llamada flujo óptico que compara los fotogramas sucesivos para determinar cambios de formas y el desplazamiento de objetos. Esto se usa para determinar la velocidad horizontal y su estabilización.
- **Sensor de presión**. Ayuda al cálculo de la altura al medir la variación de presión durante los movimientos verticales.

También es necesario especificar como están orientados los ejes del sistema de referencia centrado en el dron. Son ortogonales y nacen del centro de gravedad.

- Eje X es positivo en la dirección hacia delante y negativo hacia atrás.
- Eje Y es positivo en la dirección derecha y negativo hacia la izquierda.
- Eje Z es positivo en la dirección hacia al suelo y positivo hacia arriba.

Figura 3.2: Ejes del sistema de referencia centrado en el dron

La distribución de rotores es la clásica en un cuadricóptero con forma de X, es decir, los rotores se encuentran a $\{-45°, 45°, 135°, -135°\}$ del eje X en el plano XY.

Para poder hacer uso del mini dron son necesarias otras herramientas complementarias como son:

- Cable USB-microUSB. Necesario para cargar las baterías y conectar el dron al PC para modificar su *firmware*.
- Conexión *Bluetooth*. La conexión entre PC y dron se realiza mediante un adaptador USB *Bluetooth*, es necesario que sea de 4.0 *Low Energy*. Además, es necesario la correcta configuración de los *drivers*.
- Gafas de seguridad.
- Cartulinas y cinta azul. Utilizada para hacer la trayectoria de la aplicación propuesta que el dron deberá seguir. Se ha elegido el azul ya que es un color que contraste bien con el suelo y además facilita el tratamiento de la imagen al ser uno de los colores básicos.

3.2 Software

Como ya se vio en el Capítulo 2 existen varios entornos y aplicaciones para poder trabajar con el Parrot Mambo. En este proyecto se ha elegido Simulink junto a la *Toolbox Parrot Minidrones Support from Simulink* (23) del entorno de Matlab. Esta opción permite un control a más bajo nivel del Parrot Mambo al poder modificar su *firmware* y por ello se alinea mejor con los objetivos del trabajo.

Además, al trabajar en este entorno se tiene a disposición una serie de *Toolboxes* que ofrecen gran cantidad de opciones a la hora del desarrollo. Se podría decir que al trabajar con ellas se obtiene un efecto sinérgico ya que unas se complementan con otras generando una situación que ofrece muchas facilidades y opciones durante el trabajo. Una lista de las principales *Toolboxes* utilizadas se encuentra a continuación.

- **Aerospace Blockset y Aerospace Toolbox**. Estas *Toolboxes* ofrecen una variedad de bloques para el modelado, simulación y análisis del ámbito aeronáutico y aeroespacial. Por ejemplo, en este caso se ha hecho uso de sus modelados para la simulación de la planta de un cuadricóptero y su entorno físico. También incluye un modelado CAD del Parrot Mambo. (18), (19)
- **Simulink 3D animation**. Vincula los algoritmos de Matlab y Simulink con una visualización 3D de los objetos modelados, esto permite observar en tiempo real las simulaciones. Si además se hace uso del modelo CAD del Parrot Mambo ofrecido por las anteriores *Tolboxes* se tiene un entorno de simulación perfecto para el propósito del trabajo. (25)

- **Signal Processing Toolbox**. Provee una serie de funciones y aplicaciones para el preprocesado, análisis y filtrado de señales, esta *Toolbox* es necesaria para la gestión de los canales de información que conforman el entorno de trabajo. (24)
- **Control System Toolbox**. Ofrece algoritmos y aplicaciones para el análisis, diseño y puesta punto de sistemas de control lineales. Incluye la aplicación PID Tuner que nos permite ajustar los controladores PID en función de las necesidades particulares de cada caso. (21)
- **Image Processing Toolbox**. Este conjunto de aplicaciones y funciones permiten el procesado y análisis de imágenes en tiempo real. Sus herramientas junto con la cámara embarcada permiten llevar a cabo la aplicación propuesta, ya que facultan al Parrot Mambo a detectar la trayectoria y guiarse con ella. (22)
- **Simulink Control Design**. Esencial en el ajuste de controladores PID ya que permite la linealización del modelado del dron. (26)

Asimismo, la *Toolbox Embedded Coder* es necesaria para poder cargar los algoritmos desarrollados en el Parrot Mambo. La explicación reside en que la programación del dron está en el lenguaje C. *Embedded Coder* hace de puente entre el lenguaje de trabajo en Simulink (lenguaje M) al transcribirlo a C. De esta forma se tiene el código transcrito compatible con el Parrot Mambo de forma directa.

Figura 3.3: Esquema del flujo implementación de algoritmos en el Parrot Mambo

3.3 Conexión Parrot Mambo y Simulink

En esta sección se pretende ilustrar el proceso de conexión entre el *software* y el *hardware* como muestra del flujo de trabajo seguido.

Como ya se ha comentado es necesario tener instalada la *Toolbox Parrot Minidrones Support from Simulink* para ello será necesario navegar al menú Add-Ons en la pestaña Home y seleccionar *Get Addons*. Se abrirá una ventana donde podremos buscar la herramienta e instalarla.

Una vez instalada la *Toolbox* y con el Parrot Mambo conectado al PC mediante el cable USB-microUSb se navega una vez más al menú Add-Ons y se selecciona *Manage Add-Ons*. Aquí aparecen todas las *Toolboxes* que estén instaladas, se busca *Parrot Minidrones Support from Simulink* y se pulsa el símbolo de tuerca.

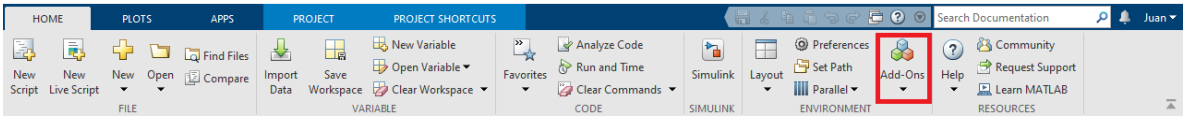


Figura 3.4: Add-Ons

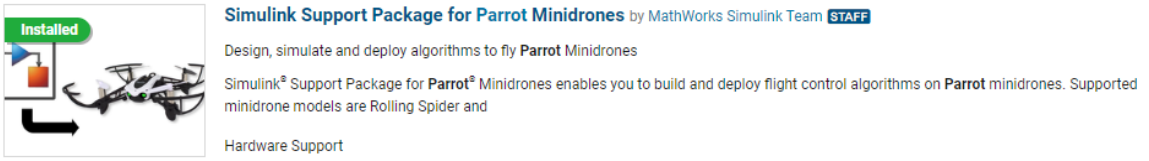


Figura 3.5: *Toolbox Parrot Minidrones Support from Simulink*. En este caso muestra como ya está instalada.

Figura 3.6: *Toolbox Parrot Minidrones Support from Simulink* instalada.

Al pulsar el botón comenzará el proceso de configuración. Primero se pedirá conectar el dron al PC y nos informará de los requisitos para el funcionamiento.

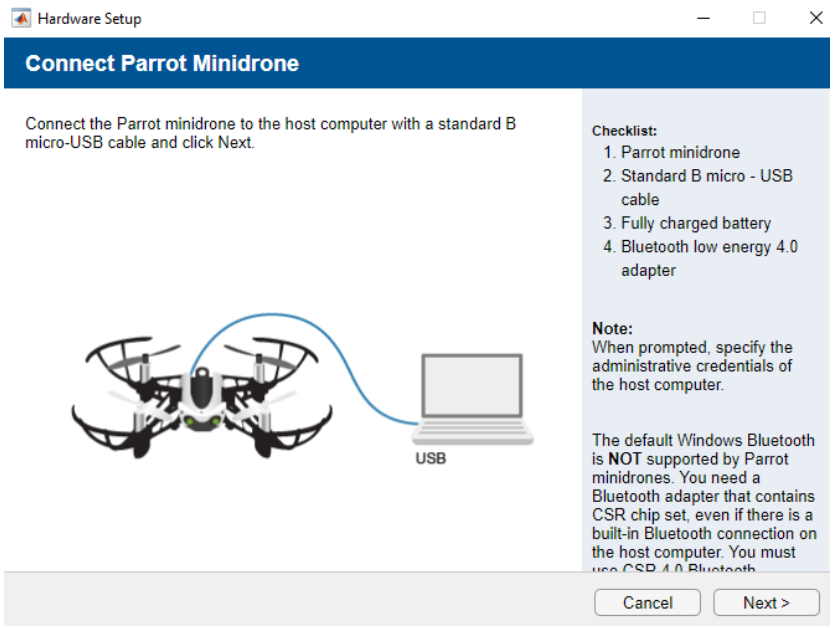


Figura 3.7: Primer menú en el proceso de configuración. Vemos como nos informa de que se necesita tecnología *Bluetooth* CSR de un adaptador especial

El segundo menú sirve para la realizar la detección del dron. Para ello está el botón Detect, al pulsarlo nos indicará qué dron está conectado y en caso de que no detecte

ninguno lo indicará.

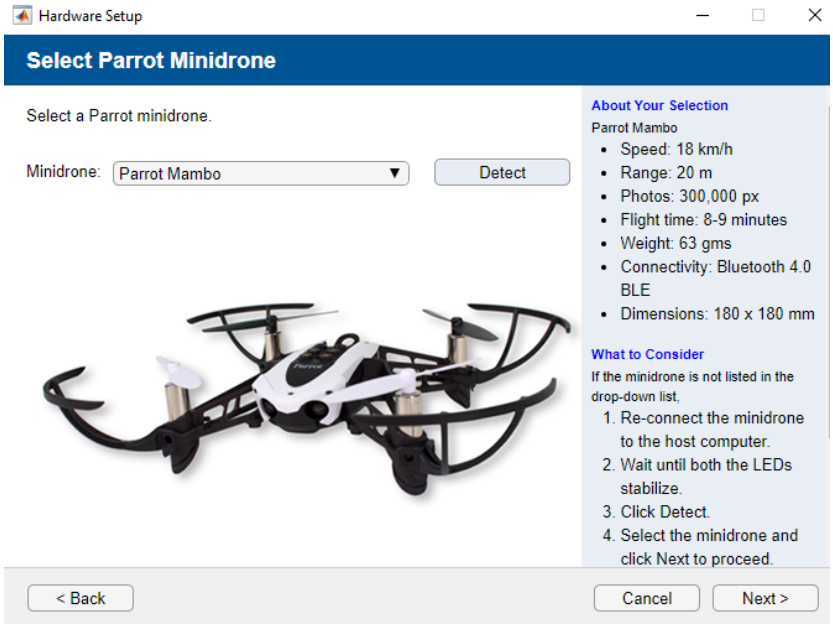


Figura 3.8: Segundo menú. A la derecha nos indica los pasos a seguir en caso de que no se reconozca el Parrot Mambo

Cuando se pulse *Next* comenzará el proceso de modificar el *firmware* del dron. Esta modificación sirve habituar al Parrot Mambo al entorno de Simulink, de forma que se permita la carga de código procedente de este.

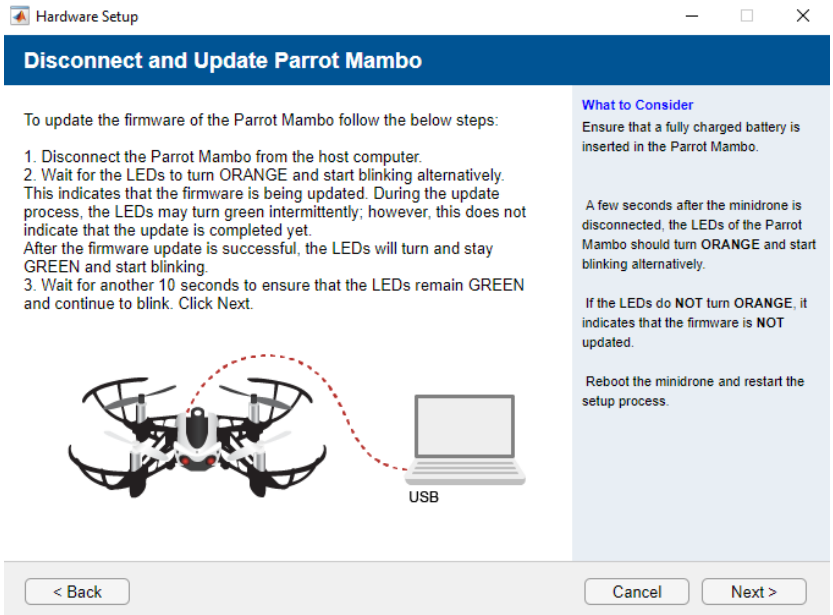


Figura 3.9: Tercer menú. Nos indica que hay que desconectar el dron y a través de los LEDs se sabrá si la actualización *firmware* ha sido satisfactoria.

En los dos siguientes menús se explica como establecer la conexión *bluetooth* con el mini dron. El proceso consiste en, primero añadirlo como dispositivo conocido al PC y después establecer una conexión correcta entre los dos.

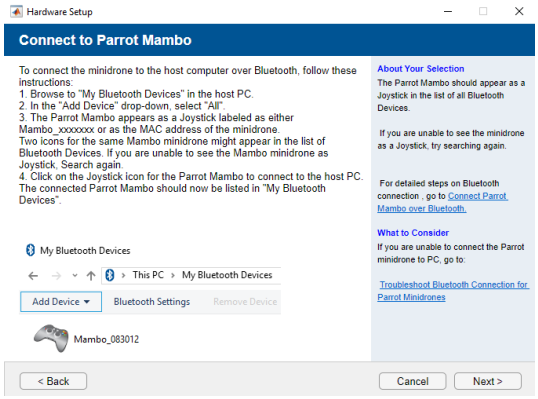


Figura 3.10: Cuarto menú

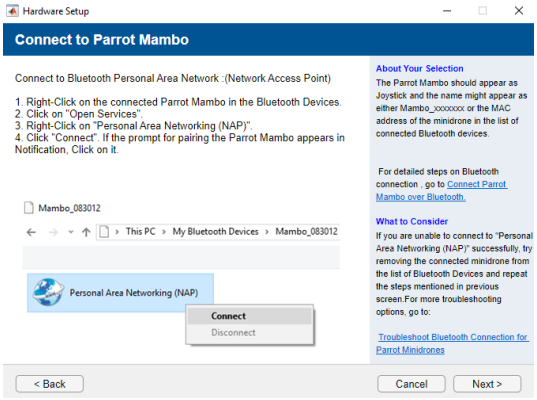


Figura 3.11: Quinto menú

La configuración depende de la distribución de Windows que se esté utilizando, en este caso se muestra el procedimiento para Windows 10.

Primero se navega a la ventana de configuración de *bluetooth* y se hace uso de la funcionalidad de *Agregar Bluetooth u otro dispositivo*. Esto abre una ventana donde se muestran todos los dispositivos detectados mediante *bluetooth*.

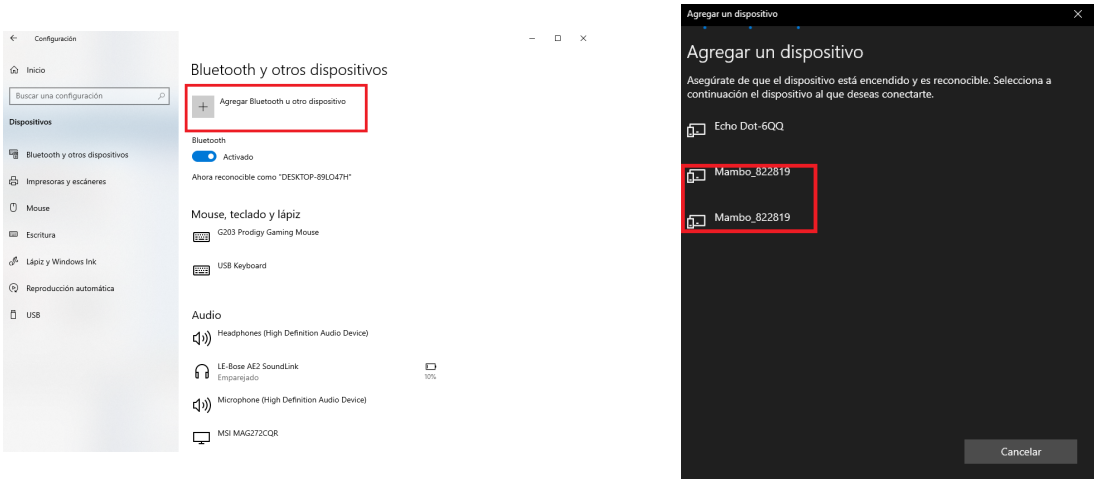
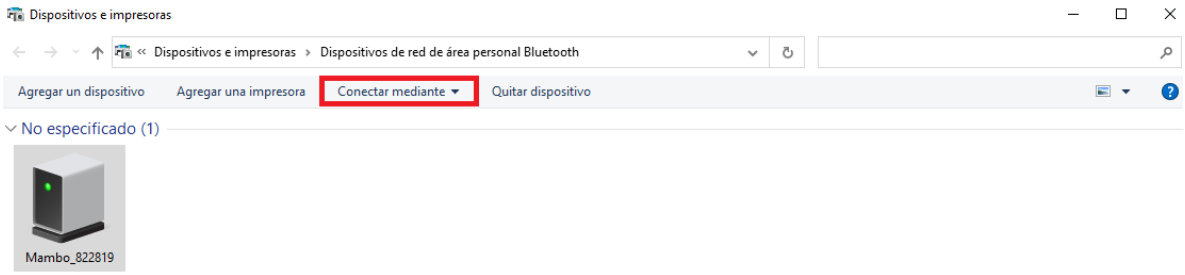


Figura 3.12: A la izquierda menú de configuración *bluetooth* de Windows 10, a la derecha se muestra la lista de dispositivos detectados, entre ellos se encuentra el Parrot Mambo

Este procedimiento ha permitido realizar la conexión preliminar, ahora se muestra cómo establecer el enlace *bluetooth* directo. Para ello se realiza clic derecho en el símbolo de *bluetooth* en la barra de herramientas y se elige *Unirse a una red de área personal*. En

la ventana que se despliega se selecciona el Parrot Mambo ya detectado y mediante la opción *Conectar mediante* se selecciona *Punto de acceso*.



La conexión directa ya está establecida, esto se puede comprobar mediante el último menú de la figura 3.13. El botón *Test Conection* permite verificar que el enlace es correcto.

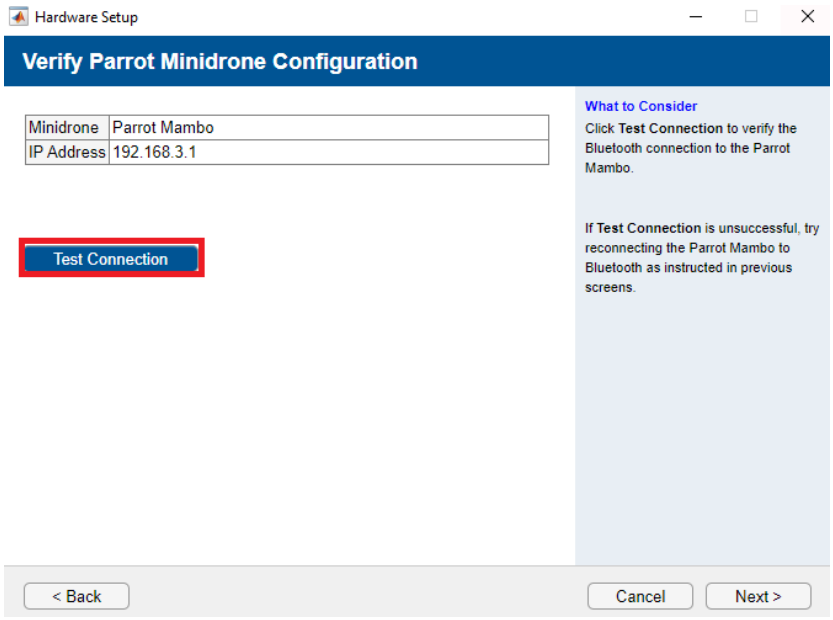


Figura 3.13: Sexto menú

Una vez terminado el proceso no será necesario repetirlo cada vez que se vaya a trabajar con el dron. Solamente será necesario conectarse al dispositivo dron mediante el menú *bluetooth*.

Capítulo 4

Modelado Matemático de un Cuadricóptero

Para poder trabajar el control de un dron primero se debe conocer cuáles son las características y principios que gobiernan su vuelo, por ello en este apartado se revisarán los principios físicos y matemáticos que modelan un cuadricóptero.

Primero se planteará cuales son los parámetros de control de vuelo y más tarde se mostrará como estos modelan las ecuaciones de su movimientos

4.1 Problema de Control y Consideraciones previas

Lo expuesto en esta sección se basa en el trabajo de Brian Douglas en (8).

Primero vamos a considerar cual es la composición de un cuadricóptero:

- Estructura central donde encontramos la batería, el procesador, los sensores etc.
- Los actuadores que son los cuatro rotores que junto a sus motores permiten el movimiento del mini dron.

Ahora se va a proponer cual es problema de control de un cuadricóptero a través del esquema de la Figura 4.1

La planta de control representada por el bloque *Drone* está gobernada por los 4 actuadores, es decir, los 4 rotores. Estos rotores producen una serie de fuerzas y momentos que controlan el movimiento del dron. El problema de control reside en cómo se pueden manipular de forma precisa los rotores para que se maniobre en el espacio tridimensional de forma satisfactoria.

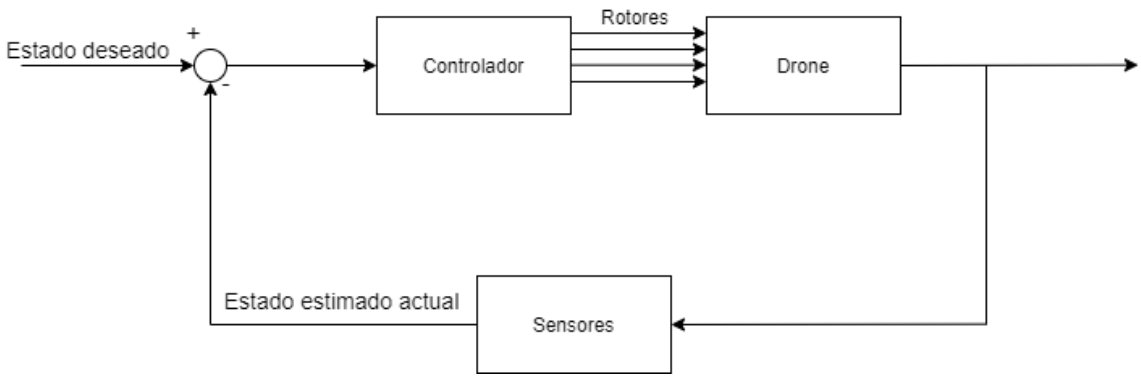


Figura 4.1: Esquema del control del mini dron

Para ello se hace uso de los sensores embarcados representados por el bloque *Sensores*, proporcionan una estimación del estado actual en el que se encuentra el sistema como puede ser la altura, velocidad horizontal, posición etc. Si se compara el estado estimado con el deseado, es decir, con la aplicación o maniobra deseada, se puede desarrollar un controlador (bloque *Controlador*) que en base a esta diferencia determine cual debe ser el comportamiento de los rotores.

De forma preliminar es necesario explicar que un cuadricóptero es un sistema infractuado, esto significa que hay más grados de libertad (GdL) que actuadores. Existen 6 GdL en total:

- Movimiento traslacional. Arriba/abajo, delante/atrás y derecha/izquierda.
- Movimiento rotacional. Alabeo (*roll*), cabeceo (*pitch*) y guiñada (*yaw*).

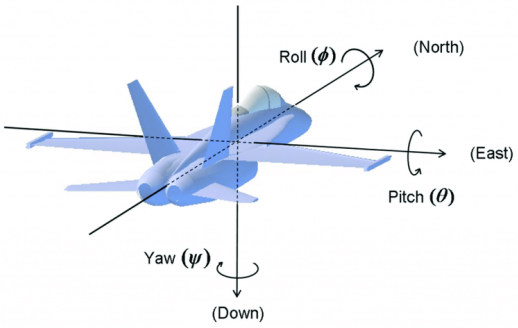


Figura 4.2: Esquema de los ángulos de rotación (6)

Ya que solo se dispone de 4 actuadores (los rotores) frente a los 6 GdL algunos movimientos no se pueden ejecutar de forma aislada. Por ejemplo, si se desea realizar un desplazamiento hacia la derecha se tendrá que realizar un alabeo primero. Esto significa

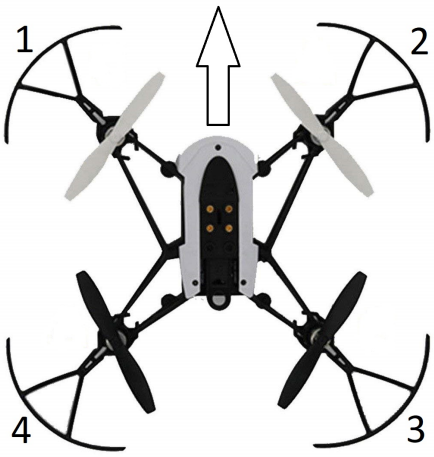


Figura 4.3: Planta del Parrot Mambo donde la flecha indica hacia delante y cada rotor está numerado

que los movimientos en el plano horizontal están acoplados a los ángulos de alabeo y cabeceo.

De esta forma, el sistema dispone como controles básicos el empuje, el alabeo, el cabeceo y la guiñada que se denominarán parámetros de control.

El principio de vuelo de un cuadricóptero se basa en que los rotores producen una fuerza hacia arriba llamada empuje o sustentación. Si esta fuerza es perpendicular al suelo y actúa sobre el centro de gravedad del sistema (CdG), este subirá sin rotación. Si, además, esta fuerza es igual en valor y opuesta en dirección a la gravedad, el sistema se mantendrá a una altura constante. Sin embargo, si esta fuerza no se coloca en el CdG se generará un momento inestabilizando el vuelo.

La solución que presenta un dron a ello es la presencia de los 4 rotores colocados opuestos unos a otros y todos a la misma distancia del CdG. De esta forma, cada rotor genera un vector de fuerza y la resultante total se coloca en el CdG asegurando la estabilidad.

Los cuatro rotores no giran en la misma dirección, sino que dos giran en sentido horario y los otros dos en antihorario, además los rotores que giran en la misma dirección están colocados en lados opuestos. Esta configuración es constante en todos los cuadricópteros y se puede observar en la Figura 4.4, en este caso los rotores 1 y 3 giran en sentido horario y los rotores 2 y 4 en sentido antihorario.

La explicación a este esquema de giros está en que cada rotor genera un momento de reacción contrario al sentido de giro de su hélice. Por lo que, si todos los rotores giran a la misma velocidad, se tendrán dos momentos de igual magnitud y sentido contrario. Estos se anularán haciendo que el cuadricóptero no rote sobre sí mismo durante el vuelo.

Sin embargo, la rotación sobre sí mismo es la guiñada, uno de los parámetros de control. Por ello, es necesario que el sisteam sea capaz de efectuar este movimiento.

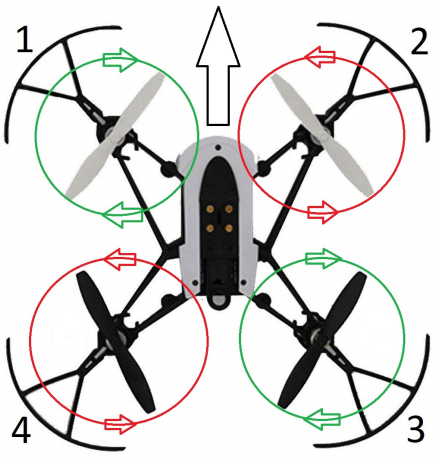


Figura 4.4: Esquema de las orientaciones de giro de los rotores

Teniendo en cuenta la configuración de giros, este movimiento se controla aumentando la velocidad de giro de los rotores de una dirección y disminuyendo los de la contraria. Así se tiene el mismo vector de fuerzas resultante sobre el CdG y un momento de guiñada controlado.

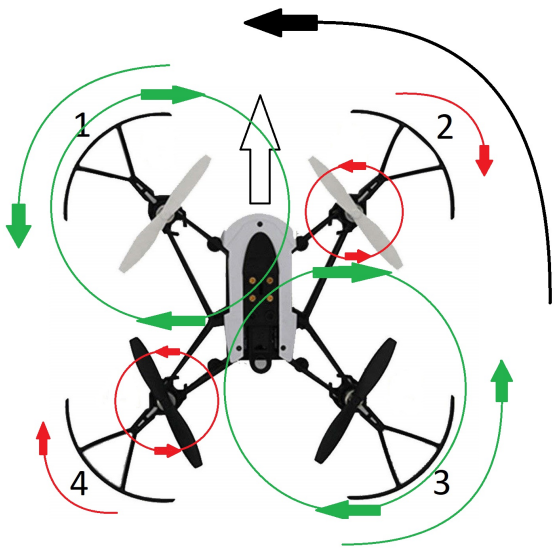


Figura 4.5: Balance de momentos para maniobra de guiñada en sentido antihorario

Además, la necesidad de realizar movimientos de guiñada explica por qué los rotores que giran en la misma dirección deben estar en posiciones opuestas. Si ambos estuviesen en el mismo lado, por ejemplo, el izquierdo, y se realizase este movimiento, al aumentar

la sustentación de los rotores del lado izquierdo se generaría un momento de alabeo indeseado.

El resto de los movimientos de control se basan en el mismo principio, si se desease realizar un desplazamiento hacia la derecha, el ejemplo anterior sería la solución, siendo esta vez el alabeo el momento buscado.

Figura 4.6: Balance de momentos para maniobra de alabeo, resultando en un desplazamiento hacia la derecha en el plano horizontal. Los rotores que aumentan su velocidad son la pareja de la izquierda (1 y 4) y los que disminuyen son los de la derecha (2 y 3)

El mecanismo para realizar el movimiento de cabeceo es el mismo pero las parejas que aumentan y disminuyen su velocidad de giro son las delanteras y traseras según la dirección de desplazamiento.

Esta configuración de giros y la disposición de los rotores hace que el empuje, el alabeo, el cabeceo y la guiñada puedan ser controlados de forma independiente unos de otros. Esto significa que se puede gobernar un movimiento sin afectar a los demás.

Esta afirmación no es del todo cierta y la realidad muestra que todos los movimientos están acoplados por interacciones aerodinámicas. Pero se puede realizar la simplificación de forma segura ya que estas interacciones no son de gran magnitud.

Por lo tanto, se puede asumir que el vuelo de un cuadricóptero se gobierna mediante el control directo del empuje, del alabeo, del cabeceo y de la guiñada. Además, estos movimientos se pueden controlar simplemente con el aumento y/o disminución de las velocidades de giro de los rotores. De forma que las órdenes que se enviarán a cada uno

de los rotores serán una mezcla de cuatro parámetros, uno para cada tipo de movimiento de control y se podría expresar con las siguientes ecuaciones:

$$Rotor_{DelanteroDerecho} = Empuje_{cmd} + Guiñada_{cmd} + Cabeceo_{cmd} + Alabeo_{cmd} \quad (4.1)$$

$$Rotor_{DelanteroIzquierdo} = Empuje_{cmd} - Guiñada_{cmd} + Cabeceo_{cmd} - Alabeo_{cmd} \quad (4.2)$$

$$Rotor_{TraseroDerecho} = Empuje_{cmd} - Guiñada_{cmd} - Cabeceo_{cmd} + Alabeo_{cmd} \quad (4.3)$$

$$Rotor_{TraseroIzquierdo} = Empuje_{cmd} + Guiñada_{cmd} - Cabeceo_{cmd} - Alabeo_{cmd} \quad (4.4)$$

Estas ecuaciones forman el algoritmo de mezcla de los rotores (*Motor Mixing Algorithm* MMA) que nos permite pasar de los parámetros de control a las velocidades de cada rotor.

4.2 Modelado Matemático

En esta sección se pretende sintetizar el modelo matemático no lineal de un cuadricóptero en base al trabajo de Karahan et al. (17) y Stevanovic et al. (33).

Se considera que el dron es un cuerpo rígido. Un cuerpo se clasifica como rígido cuando la distancia entre los puntos que lo forman se mantiene siempre constante. Las fuerzas y momentos que son aplicados sobre el cuerpo no alteran el tamaño y forma del cuerpo de forma significativa, es decir, no se producen deformaciones (4).

También se van a considerar dos sistemas de referencia:

- Sistema de coordenados fijo en el CdG del dron. *Body-fixed frame*
- Sistema de coordenados inercial fijo a la Tierra. *Earth reference frame*

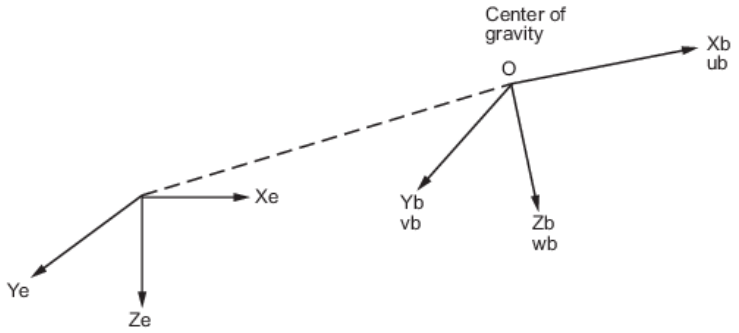


Figura 4.7: Sistemas de coordenadas. A la izquierda *Earth reference frame* a la derecha *Body-fixed frame*

Para pasar de un sistema a otro se hace uso de los llamados ángulos de Euler que sirven para indicar la orientación del sistema centrado en el mini dron respecto al inercial. En este caso los ángulos de Euler son el alabeo, cabeceo y guiñada definidos por $\eta = [\phi\ \theta\ \psi]^T$.

4.2.1 Cinemática y Dinámica de rotación

La dinámica de rotación de un cuerpo rígido en el sistema de referencia del CdG del dron se expresa como

$$I\dot{\omega} + \omega \times (I\omega) = \tau \tag{4.5}$$

donde $I = diag(I_x, I_y, I_z)$ es la diagonal de la matriz de inercia, $\omega = [p\ q\ r]^T$ es el vector de velocidad angular y $\tau = [\tau_\phi\ \tau_\theta\ \tau_\psi]^T$ es el vector de momentos generado por los rotores.

y la cinemática de rotación se expresa como

$$\dot{\eta} = \Omega_B \omega \tag{4.6}$$

donde Ω_B matriz de transformación del sistema centrado en el dron al inercial, está definida como

$$\Omega_B = \frac{1}{\cos\theta} \begin{bmatrix} \cos\theta & \sin\phi\sin\theta & \cos\phi\cos\theta \\ 0 & \sin\phi\cos\theta & -\sin\phi\cos\theta \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \tag{4.7}$$

4.2.2 Cinemática y Dinámica de traslación

La dinámica de traslación del cuerpo rígido en el sistema centrado en el cuerpo se define como

$$m\dot{v} + \omega \times (mv) = F \tag{4.8}$$

donde m es la masa del cuadricóptero, $F = [F_x\ F_y\ F_z]^T$ es el vector de fuerzas y $v = [u\ v\ w]^T$ es el vector de velocidad lineal del dron.

La ecuación de cinemática del cuerpo rígido es

$$\dot{\xi} = Rv \tag{4.9}$$

donde $\xi = [x\ y\ z]^T$ es el vector de posición sistema de coordinadas inercial y R es la matriz de rotación definida por ($c\theta \equiv \cos\theta$, $s\theta \equiv \sin\theta$)

$$R(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \tag{4.10}$$

$$R(\theta) = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \tag{4.11}$$

$$R(\psi) = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.12}$$

$$R = R(\phi, \theta, \psi) = R(\phi)R(\theta)R(\psi) \tag{4.13}$$

$$R(\phi, \theta, \psi) = \begin{bmatrix} c\theta c\psi & c\theta s\psi & -s\theta \\ s\theta s\phi c\psi - s\psi c\phi & s\psi s\theta s\phi + c\psi c\phi & s\phi c\theta \\ s\theta c\phi c\psi + s\psi s\phi & s\psi s\theta c\phi - c\psi s\phi & c\phi c\theta \end{bmatrix} \tag{4.14}$$

4.2.3 Fuerzas y momentos

Cada rotor al girar a una velocidad angular de Ω_i produce una fuerza $f_i = k_i\Omega_i^2$ donde $i = 1, 2, 3, 4$ ya que hay cuatro rotores, k_i es una constante positiva. Estas cuatro fuerzas como hemos visto en los apartado anteriores son equivalentes a nivel de control al empuje $F = [0\ 0\ F_z]^T$ y a los momentos alabeo, cabeceo y guiñada $(\tau_\phi, \tau_\theta, \tau_\psi)$.

Por lo tanto, la distribución de control de los cuatro rotores que generan las fuerzas y momentos es

$$\begin{bmatrix} F_z \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -l & 0 & l & 0 \\ 0 & l & 0 & -l \\ c & -c & c & -c \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} \tag{4.15}$$

donde l es la distancia desde los rotores al centro de gravedad y c es una constante conocida que relaciona la fuerza y el momento. De esta forma, si se conoce el empuje y los momentos podemos calcular la fuerza de cada rotor y viceversa.

4.2.4 Modelo dinámico simplificado

Durante el vuelo de un cuadricóptero, si se considera que no hay grandes cambios de velocidad lineal en el plano XY, los ángulos de alabeo y cabeceo se pueden aproximar a 0. Esto resulta en que la matriz Ω_B sea igual a la matriz de identidad y que la derivada de los ángulos de Euler $\dot{\eta}$ sean iguales a la velocidad angular ω. También se considera que no hay fuerzas externas.

Bajo estas suposiciones, el modelo dinámico de un cuadricóptero se puede expresar bajo las siguientes ecuaciones de movimiento

$$m\ddot{x} = (\cos\psi \sin\theta \cos\phi + \sin\psi \sin\phi)F_z \tag{4.16}$$

$$m\ddot{y} = (\sin\psi \sin\theta \cos\phi - \cos\psi \sin\phi)F_z \tag{4.17}$$

$$m\ddot{z} = -mg + (\cos\phi \sin\theta)F_z \tag{4.18}$$

$$I_x\ddot{\phi} = (I_y - I_z)\dot{\theta}\dot{\psi} + \tau_\phi \tag{4.19}$$

$$I_y\ddot{\theta} = (I_z - I_x)\dot{\phi}\dot{\psi} + \tau_\theta \tag{4.20}$$

$$I_z\ddot{\psi} = (I_x - I_y)\dot{\phi}\dot{\theta} + \tau_\psi \tag{4.21}$$

Capítulo 5

Simulación y modelo de vuelo

Ahora que se conoce el funcionamiento físico de un cuadricóptero así como su modelado matemático se presentará el modelo del Parrot Mambo en Simulink. En él se verán todos los subsistemas que en su conjunto permiten simular un dron. Además, el subsistema de control de vuelo también se utiliza para cargar en el Parrot Mambo los algoritmos de vuelo desarrollados.

Estudiar y comprender este modelo ayudará en la tarea de comprender el funcionamiento general de los drones. En cada subsistema se plasmará su cometido, su funcionamiento y como interactúa con los demás, así se obtendrá una perspectiva general que facilitará el diseño de sistemas de control.

También se hablará más en profundidad de la metodología de trabajo utilizada, el diseño basado en modelos.

5.1 Simulink y el diseño basado en modelos

La herramienta *software* principal de trabajo es Simulink, en esta sección se pretende abordar sus características y tratar el diseño basado en modelos (*model-based design*).

Simulink es un entorno de programación visual que trabaja dentro del entorno Matlab. Se basa en el modelado de sistemas lineales, no lineales, continuos o discretos que sirven como simulación de sistemas reales. Por tanto, se trata de una plataforma perfecta para el diseño de prototipos, estudios de parámetros y exploración de conceptos.

Simulink trabaja con diagramas de bloques, lo cuales realizan funciones específicas sobre su entrada y el resultado es propagado por su salida. Existen gran variedad de bloques por lo que se puede simular casi cualquier modelo, además cada *Toolbox* añade bloques específicos de su campo.

La propuesta más interesante de Simulink es la posibilidad de realizar el proceso de

diseño mediante el uso de modelos. Cuando se está trabajando en el desarrollo de un sistema muchas cuestiones de diseño solo pueden ser respondidas mediante el trabajo sobre un prototipo. Con este prototipo se realiza un trabajo de *ensayo y error* para comprobar si las decisiones de diseño son las correctas. Si no lo son o si se producen cambios en los requerimientos del sistema, el prototipo perderá gran parte de su funcionalidad retrasando el proyecto y aumentando los costes.

Por ello muchos equipos de desarrollo en diferentes industrias diseñan modelos de sus proyectos primero, de esta forma disponen de una base sobre la que realizan simulaciones, y como resultado de estas soluciones obtienen resultados preliminares, comprueban escenarios, tendencias de diseño etc. Esto les permite mejorar el producto, reducir costes y reducir los problemas de desarrollo. (27)

El flujo de trabajo cuando se trabaja con el diseño basado en modelos se puede resumir de la siguiente manera

- **Preproceso**. Los primeros pasos consisten en plantear los objetivos del proyecto, así como los requerimientos técnicos y las decisiones de diseño.
- **Modelado**. Una vez está claro aquello que se quiere desarrollar se acude a un *software* como Simulink para desarrollar el modelo del proyecto. Esto permite adaptarse rápidamente a posibles cambios de diseño.
- **Simulación**. Con el modelo ya preparado se pueden realizar simulaciones de funcionamiento con distintos parámetros y en distintos escenarios de trabajo.
- **Integración y generación de código**. En el diseño basado en modelos no es necesario escribir el código ni los algoritmos de control por programación tradicional, el propio modelo diseñado genera el código que se puede integrar en el *hardware*.
- **Evaluación y verificación**. La fase final del desarrollo consiste en comprobar el funcionamiento de tu modelo una vez implementado y verificar el correcto funcionamiento. Los resultados pueden no ser exactamente los mismos ya que el modelado suele asumir simplificaciones.

5.2 Modelo de vuelo

En este trabajo se utiliza el diseño basado en modelos utilizando como base el proyecto *parrotMinidroneCompetition*. Este proyecto está incluido en la *Toolbox Parrot Minidrones Support from Simulink* y presenta un modelado del Parrot Mambo listo para ser implementado en el *hardware*. Este proyecto se utilizará como la base sobre la que trabajar en la mejora de controladores e implementación de algoritmos. Por ello en este Capítulo se presentarán y explicaran todos los componentes que lo conforman.

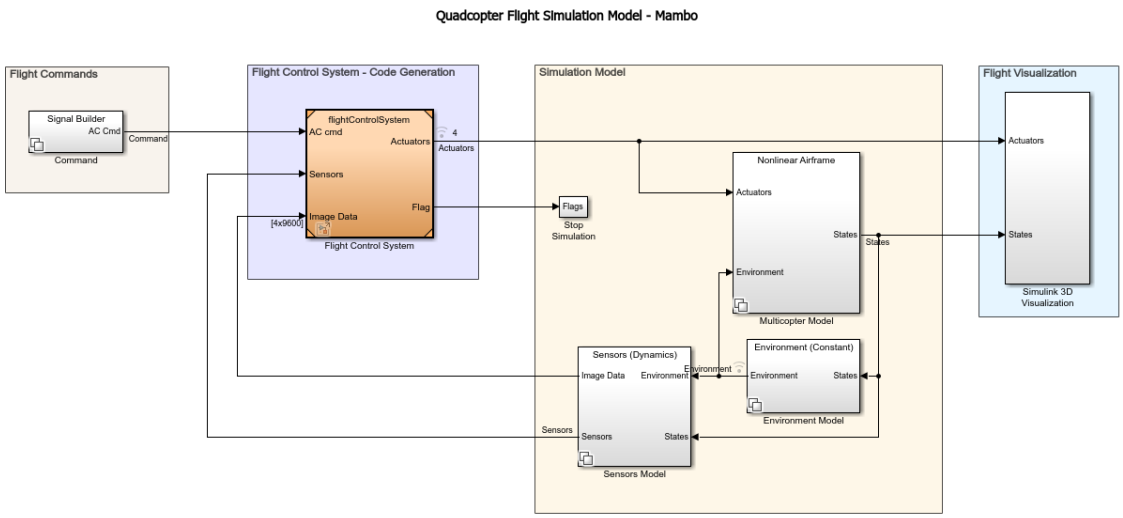


Figura 5.1: Modelo de simulación del Parrot Mambo en Simulink

En la Figura 5.1 se puede ver el modelo de simulación del Mambo. Existen cuatro bloques diferenciados que en este caso se denominarán subsistemas, ahora se procederá a explicar el propósito y funcionamiento de cada uno.

Es conveniente comentar que el proyecto al abrirse carga todos los parámetros necesarios para su funcionamiento. Estos parámetros son las matrices de inercia, masa, las constantes ambientales como presión o temperatura, valores de calibración etc.

5.2.1 Comandos de vuelo. *Flight Commands*

El primer subsistema a comentar es el que se encuentra en la esquina superior izquierda en color beis. Se puede ver que su nombre es comandos de vuelos (*Flight Commands*) y está compuesto por un único bloque con el nombre de constructor de señales (*Signal Builder*). No tiene entradas y solo tiene una salida de nombre Comandos (*Commands*).

Este subsistema se encarga de producir las órdenes de vuelo al resto del sistema, es decir, da las indicaciones de posición y orientación del dron deseada a lo largo del tiempo. Como no posee entradas, este subsistema es independiente del resto, es decir, que mediante este subsistema no se puede generar sistemas de control realimentados.

Este subsistema ofrece varios métodos de control, que se pueden observar en la Figura 5.2.

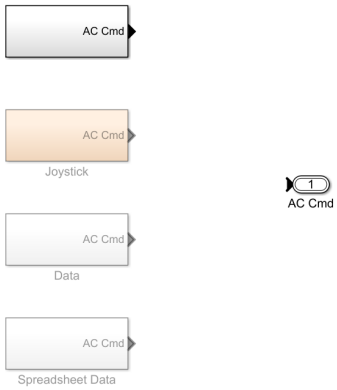


Figura 5.2: Opciones de comandos de vuelo

Existen cuatro opciones diferentes, la primera no posee nombre y es la seleccionada de forma predeterminada. Su contenido se puede ver en la Figura 5.3.

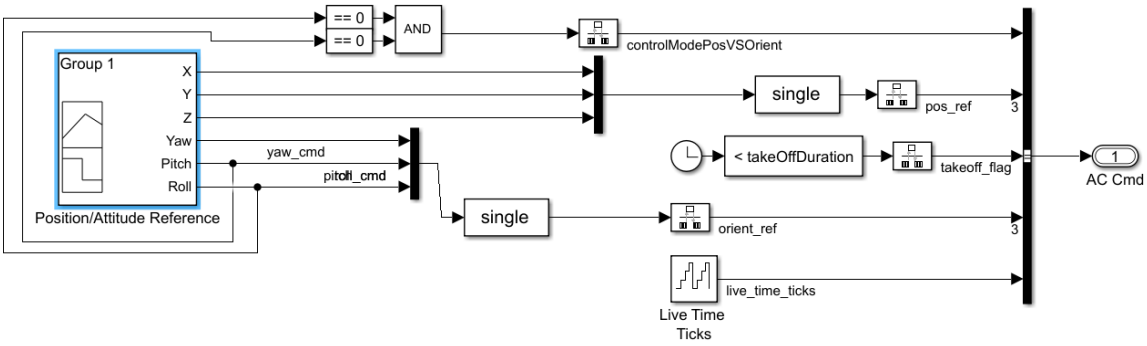


Figura 5.3: Dentro del primer bloque de comandos de vuelo

El bloque del extremo izquierdo es el *Signal Builder* que genera las señales del movimiento del dron. Se generan seis señales, tres corresponden a la referencia de posición (X, Y, Z) y las otras tres a la referencia de actitud (Yaw, Pitch, Roll).

Cada grupo de estas señales se agrupan en tres para formar los vectores *pos_ref* y *orient_ref*, además se pueden ver otras señales como son un conteo de ticks de simulación y otro de una bandera lógica que marca el despegue (*take-off*). Al final todas ellas se agrupan en un canal de información (también llamado *bus*) para salir del subsistema.

El resto de las opciones vistas en la Figura 5.2 funcionan de la misma manera pero cambian en la forma de generar los comandos de posición y orientación. El segundo bloque *Joystick* se usa cuando se quiere controlar el mini dron directamente desde unos mandos de vuelo.

Finalmente la tercera y cuarta opción se utilizan cuando los comandos están alojados

en archivos, archivos *.mat* (formato de guardado de datos de Matlab) para la tercera opción y *.xlsx* (formato utilizado en hojas de cálculo como Excel) para la cuarta.

Este subsistema solo es usado durante las simulaciones y no es cargado al dron cuando se va a trabajar con él. Dado que para la aplicación propuesta es necesario un control en bucle utilizando datos de los sensores este subsistema pierde significado. Sin embargo, no es completamente inútil ya que sirve para crear los canales de información que serán usados más tarde.

5.2.2 Sistema de control de vuelo. *Flight Control System - Code Generation*

En este subsistema se aloja toda la lógica que encargada de determinar las órdenes de control en tiempo real. También incluye el sistema de procesamiento y análisis de imagen. Por ello, este es el único subsistema que se carga en el *hardware*.

Si se entra dentro del subsistema se puede ver que está dividido en el sistema de control y el de procesamiento de imágenes. Del segundo se hablará más adelante, ahora se va a abordar el bloque de sistema de control (*Control System*).

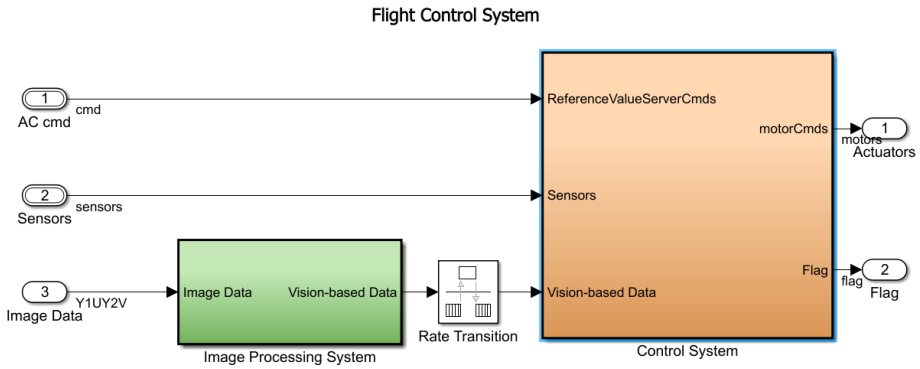


Figura 5.4: Dentro del bloque *Flight Control System - Code Generation*

Como entradas del sistema de control se tiene: el canal de información de control de vuelo proveniente del subsistema anterior (*AC cmd*), los datos de la sensorización embarcada (*Sensors*) y los del procesamiento de imágenes (*Vision-based Data*). Como salida están las órdenes a los actuadores (*motorCmds*) y al sistema de banderas lógicas (*Flag*). Este último se encarga de detener el vuelo cuando existen errores o es inestable. El bloque *Rate Transition* sirve para igualar el ratio de transmisión de información entre la cámara (60Hz) y el mini dron (200Hz).

En la Figura 5.5 se muestra la composición del sistema de control, se pueden encontrar 4 bloques principales:

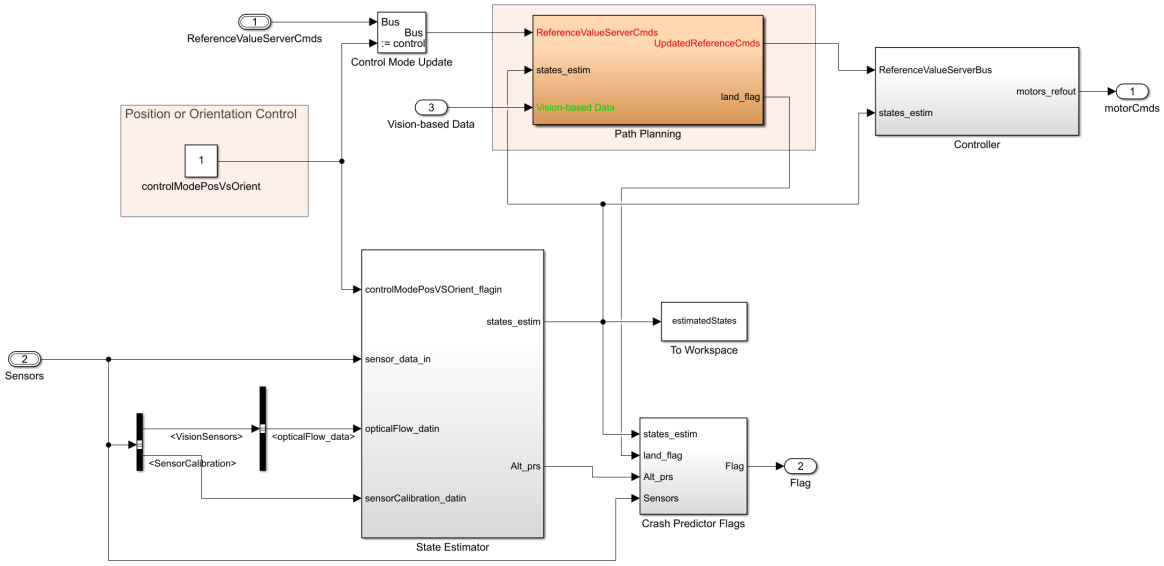


Figura 5.5: Dentro del bloque *Control System*

- *State Estimator*. Recibe información de los sensores y la usa para estimar el estado del dron, esto es recogido y propagado por un canal de información denominado *states_estim* por su salida.

  Está formado por un bloque inicial de preprocesamiento que toma todos los datos provenientes de los sensores y los adecua para su uso en los bloques posteriores. Estos bloques posteriores como se puede ver se dedican a estimar la posición (*XY Position Estimator*), la altura (*Altitude Estimator*) y la orientación (*Attitude Estimator*) como se puede ver en la Figura 5.6.

  A modo de ejemplo en la Figura 5.7 se muestra el bloque *Attitude Estimator*, que utiliza las lecturas de los giroscopios y acelerómetros del IMU para estimar las variaciones de en los ángulos de *Yaw*, *Pitch* y *Roll*. Estas estimaciones se realizan mediante integradores y filtros de Kalman.

- *Path Planning*. Aloja la lógica de vuelo y da los comandos de vuelo. En este bloque se desarrollará la lógica de seguimiento de trayectorias que será explicada más tarde. Su salida se denomina *UpdatedReferenceCmds* ya que actualiza los comandos de vuelo.

- *Crash Predictor Flags*. Este bloque es el encargado de activar una bandera de error que detenga de inmediato el vuelo del mini dron. Las condiciones para que salte la bandera son varias: si el dron gana demasiada velocidad o se aleja del punto de despegue más de 10 metros. También salta la bandera si los sensores fallan y el dron deja de poder determinar su estado. Su lógica se puede ver en la Figura 5.8

- *Controller*. Este bloque es el encargado de comparar los procedentes del *Path Planning* con el estado estimado (sus dos entradas) y mediante controladores

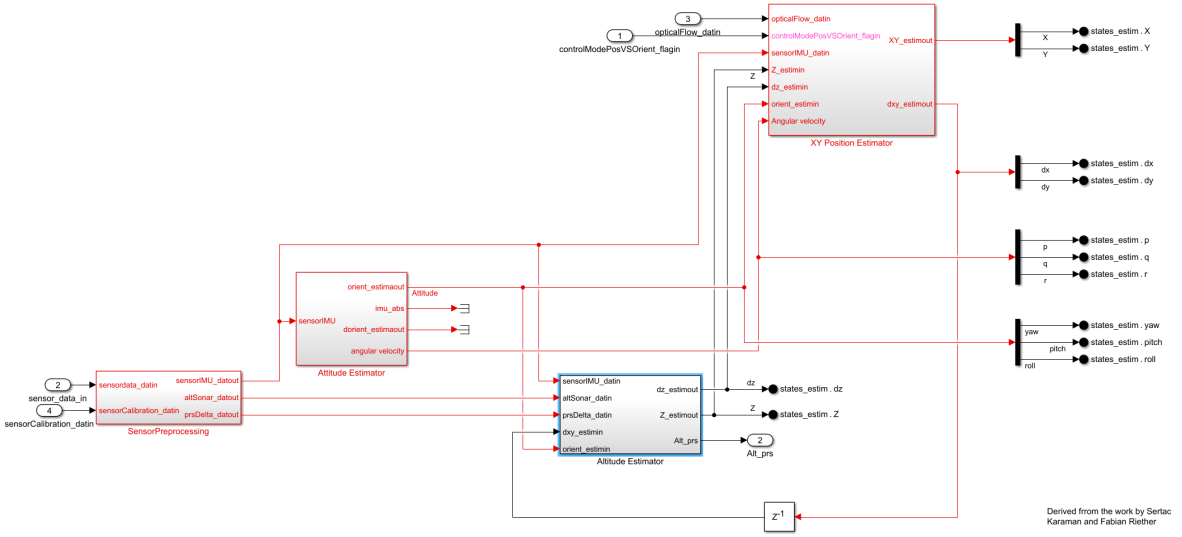


Figura 5.6: Dentro del bloque *State Estimator*

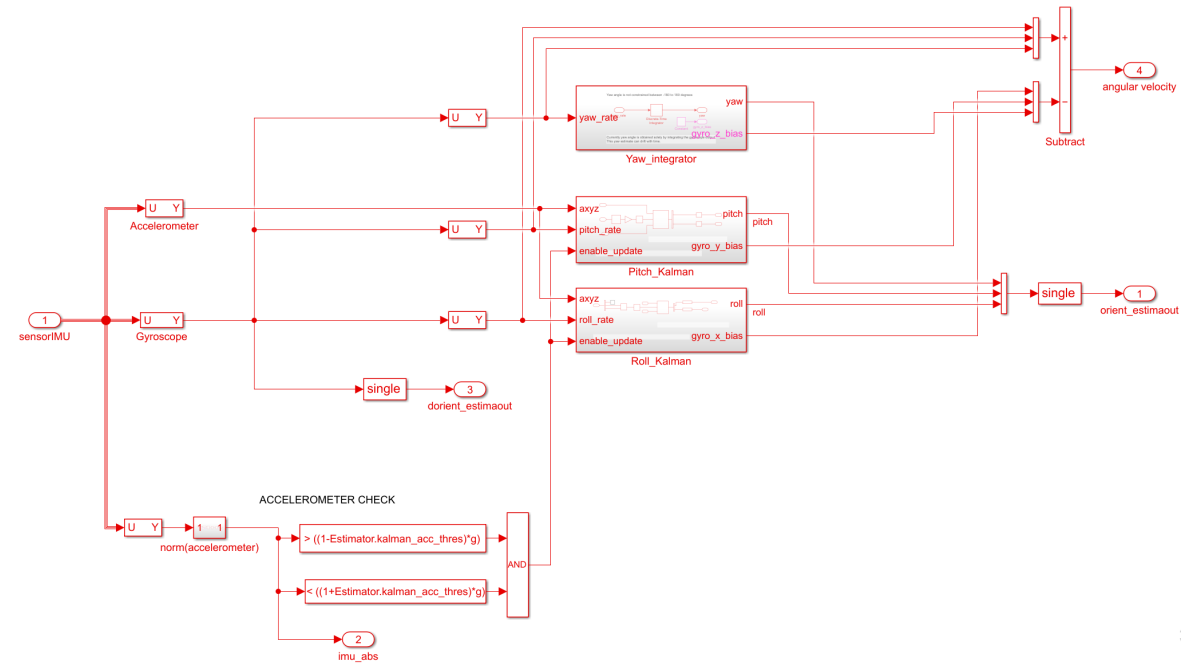


Figura 5.7: Dentro del bloque *Attitude Estimator*

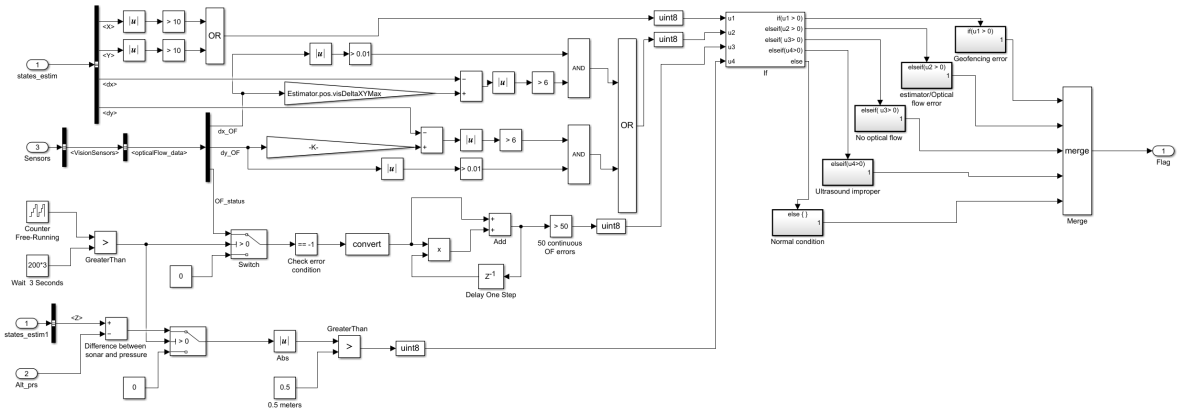


Figura 5.8: Dentro del bloque *Crash Predictor Flags*

PID determina las órdenes a los rotores para alcanzar el estado deseado. Es el mismo concepto que el tratado en el Capítulo 4, Figura 4.1 donde se planteaba el problema de control. Se Hablará de este bloque con mayor profundidad en el siguiente capítulo.

5.2.3 Modelo de simulación *Simulation Model*

Este subsistema se encarga de la simulación del dron como un cuadricóptero con 6 GdL así como de su entorno y sus sensores. Naturalmente este subsistema no se carga al dron para los ensayos de campo, pero es esencial en el proceso del diseño ya que es el encargado emular los sistemas durante las simulaciones.

Si se regresa a la Figura 5.1 se observará que está compuesto por 4 bloques. El primero denominado *Multicopter Model* emula al Parrot Mambo en cuanto a que lo considera un cuadricóptero con 6 GdL. Se observa que este bloque toma las órdenes a los actuadores e información del entorno y genera el estado resultante simulado. Se puede observar que tiene el título de *Nonlinear Airframe*, es decir, que un modelado no lineal. Como se puede ver en la Figura 5.9 se tiene a disposición dos modelados: uno lineal y uno no lineal.

En el modelo no lineal se encuentran dos bloques principales (Figura 5.10). El primero *AC model* que modeliza los actuadores del mini dron y segundo *6DOF (Quaternion)* que modela el movimiento de un cuerpo rígido con 6 grados de libertad.

El bloque *6DOF (Quaternion)* proviene de *Aerospace Toolbox* y resume todo el modelado matemático de un cuadricóptero en un solo bloque. Con las fuerzas y momentos generadas por los actuadores en el centro de gravedad y que en este caso provienen del bloque *AC model* determina la posición, velocidad y aceleración del dron.

El bloque *AC model* toma como entrada los comandos a los actuadores, la información

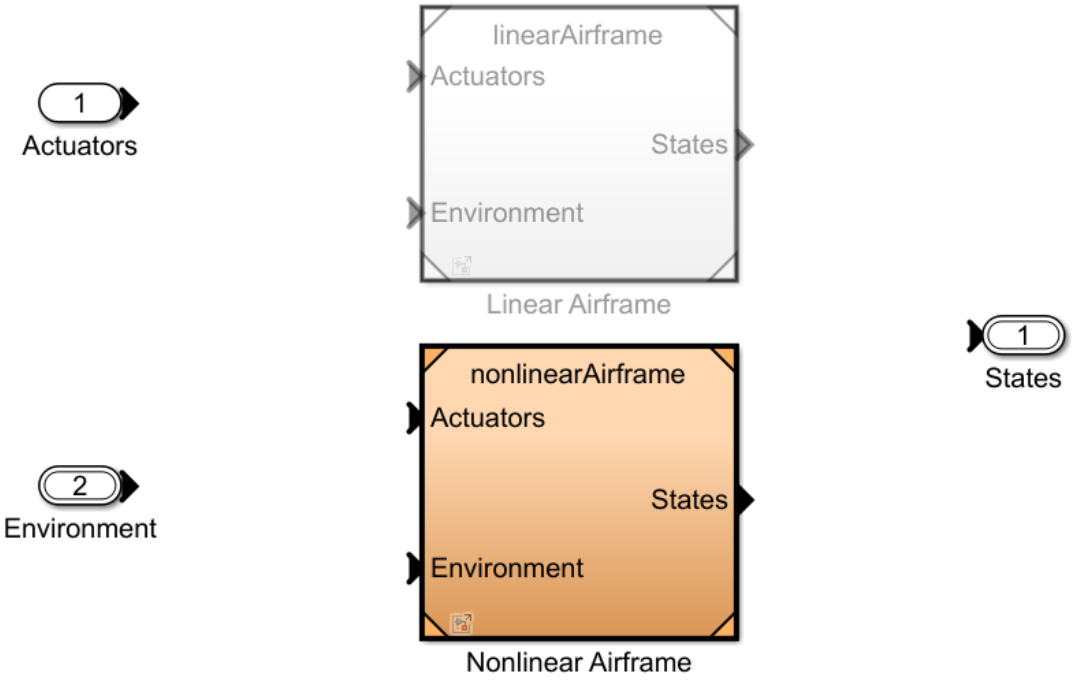


Figura 5.9: Dentro del bloque *Multicopter Model*

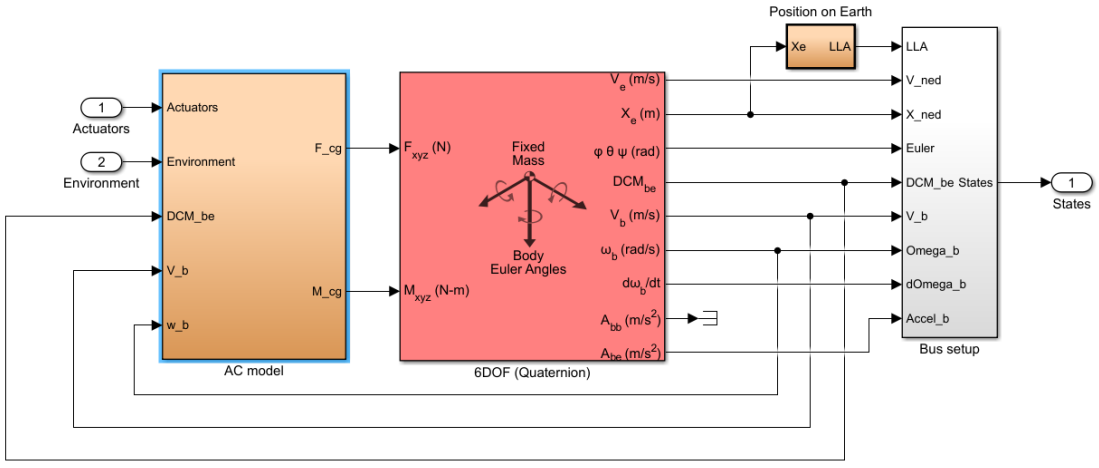


Figura 5.10: Dentro del bloque *Nonlinear Airframe*

del entorno e información de la velocidad del dron. La salida son las fuerzas y momentos que producen los rotores, en la siguiente Figura se puede ver su composición.

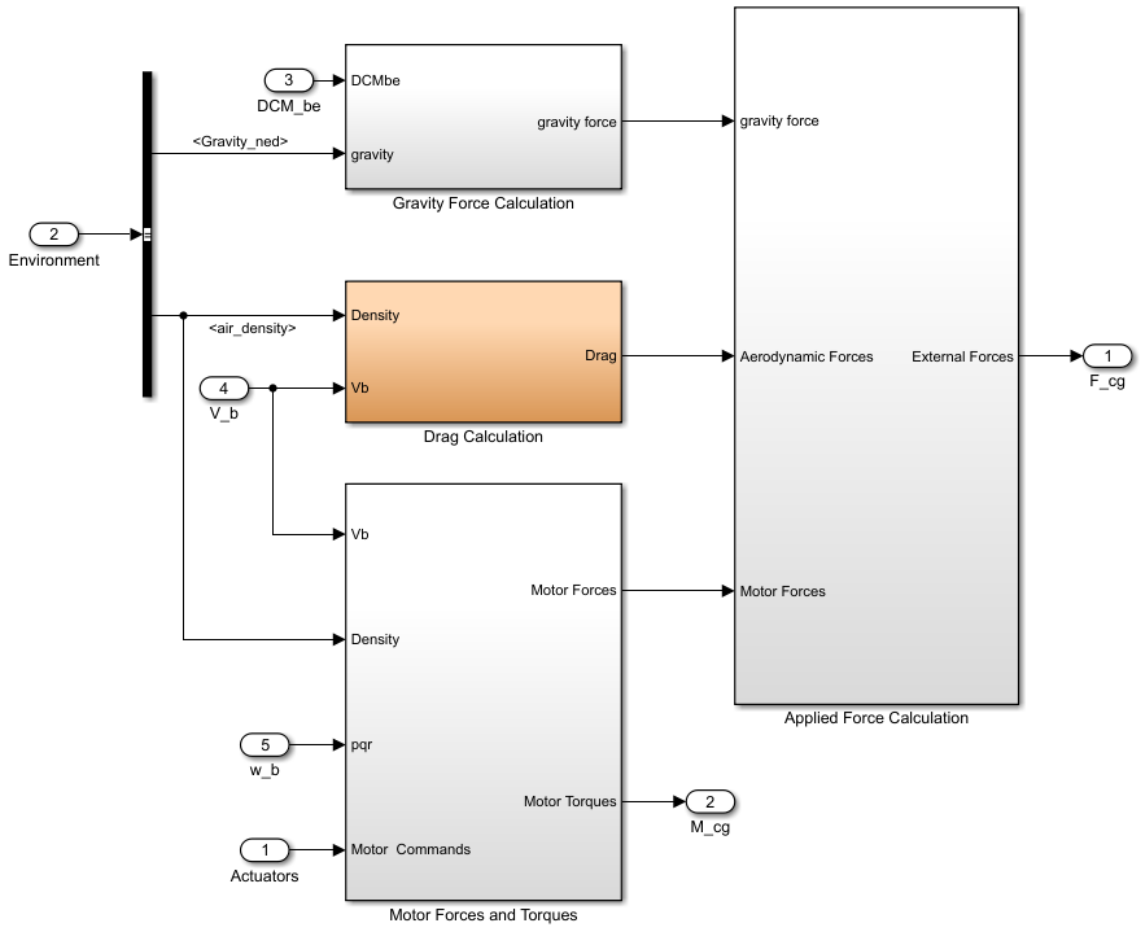


Figura 5.11: Dentro del bloque *AC model*

Hay cuatro bloques principales:

- ***Gravity Force Calculation***. Este bloque se encarga de transformar el vector de gravedad al sistema de referencia del dron mediante multiplicación matricial.

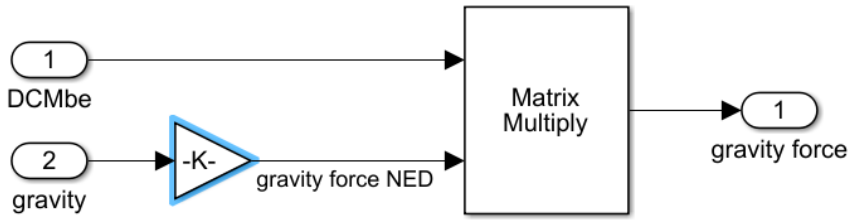


Figura 5.12: Dentro del bloque *Gravity Force Calculation*

- ***Drag Calculation***. Calcula el *drag* usando la velocidad del dron, la densidad del aire, el coeficiente de *drag* propio del dron y sus dimensiones. El coeficiente de *drag* ha sido estimado utilizando un modelo CAD del minidron.

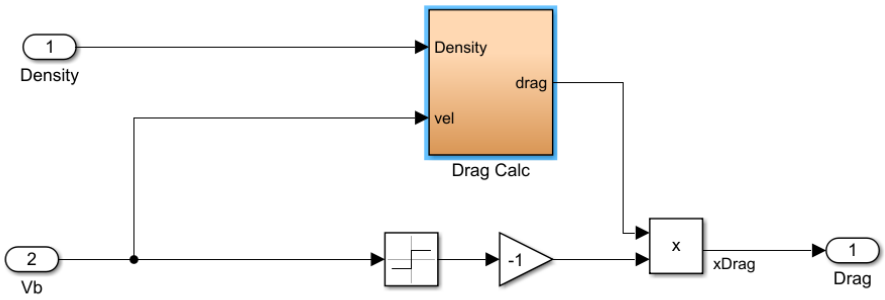


Figura 5.13: Dentro del bloque *Drag Calculation*

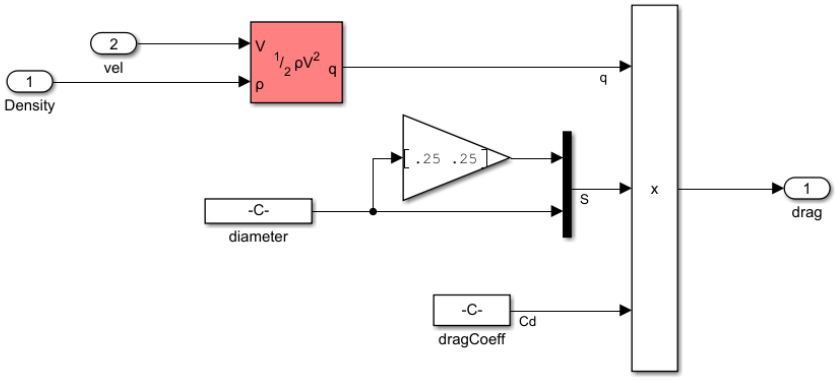


Figura 5.14: Dentro del bloque *Drag Calc*

- ***Motor Forces and Torques***. Usando la comandos a los actuadores determina las fuerzas y momentos que estos generan.

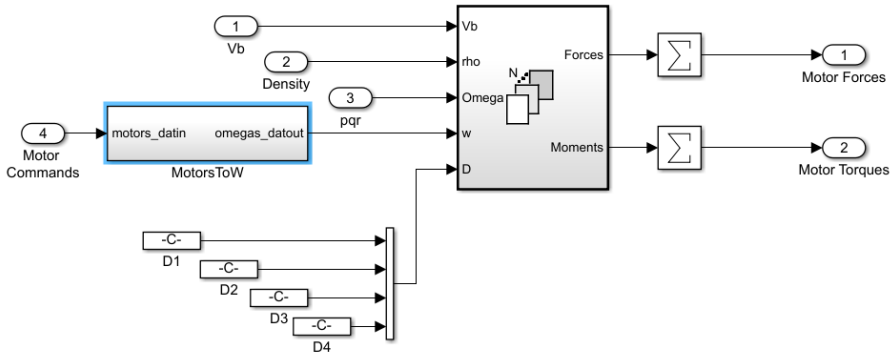


Figura 5.15: Dentro del bloque *Motor Forces and Torques*

- ***Applied Force Calculation***. Realiza el balance de fuerzas sumando las salidas de los bloques anteriores

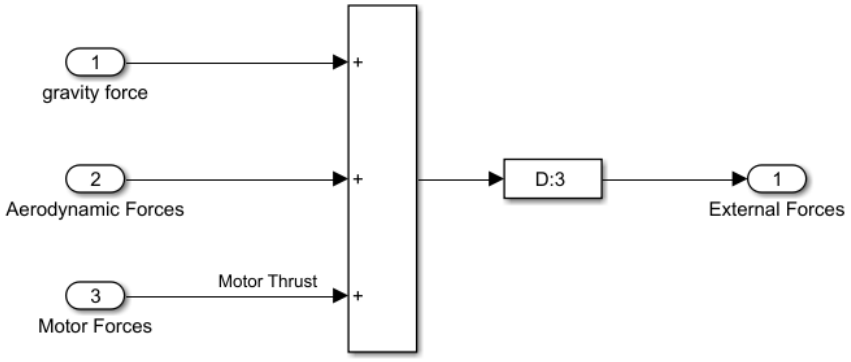


Figura 5.16: Dentro del bloque *Applied Force Calculation*

El siguiente bloque a tratar en el subsistema de simulación es el encargado de la simulación del entorno (*Environment Model*) de la Figura 5.1. Contiene dos opciones, una constante y otra variable con la altura. En este caso dado que la altura no varía en gran medida es seguro utilizar el modelo constante.

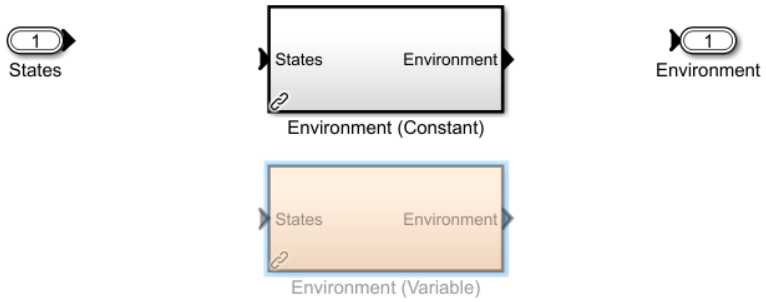


Figura 5.17: Dentro del bloque *Environment Model*

Dentro del modelado constante se puede ver como se genera un canal de información con los datos del vector de gravedad, parámetros atmosféricos y campo magnético. Asume los valores a nivel del mar.

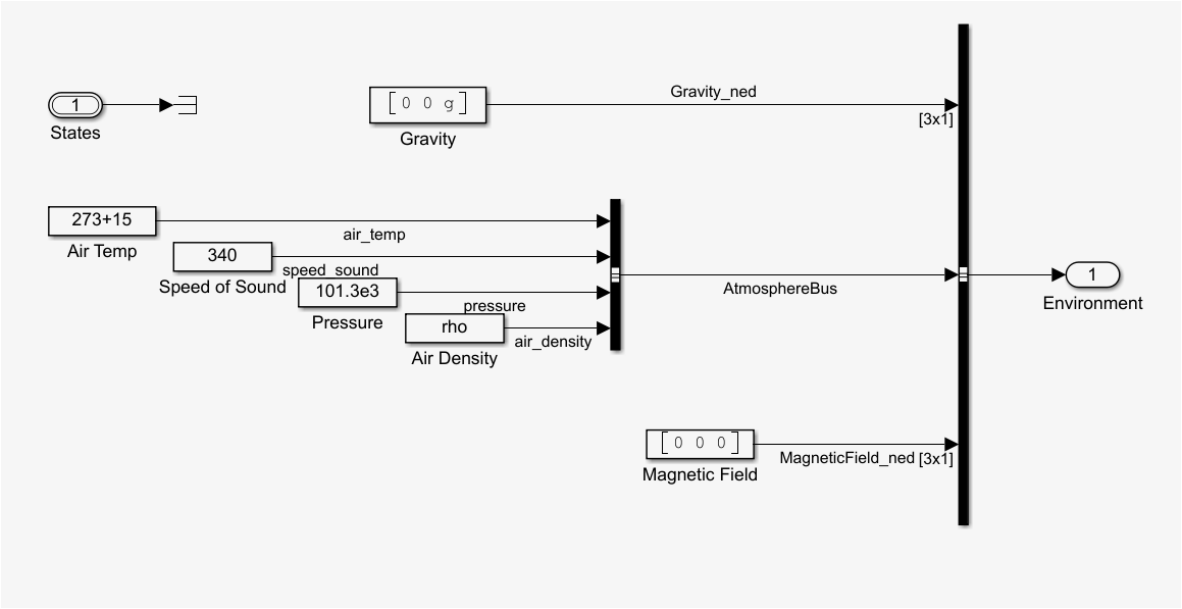


Figura 5.18: Dentro del bloque *Environment (Constant)*

Para el simulado de los sensores se utiliza el bloque *Sensor Model* de la Figura 5.1 que también ofrece dos opciones. *Feedthrough* y *Dynamics*, el segundo intenta simular los sensores de forma más fidedigna y por ello incluye ruido a las señales, mientras que el primero no lo hace. Para la simulación se usa el segundo ya que se busca los resultados más similares posibles al sistema real.

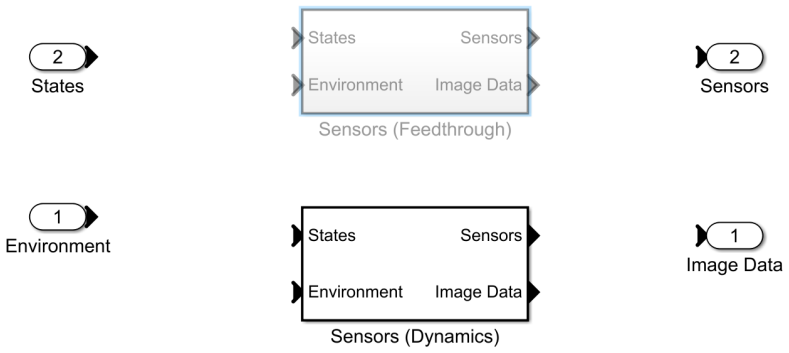


Figura 5.19: Dentro del bloque *Sensor Model*

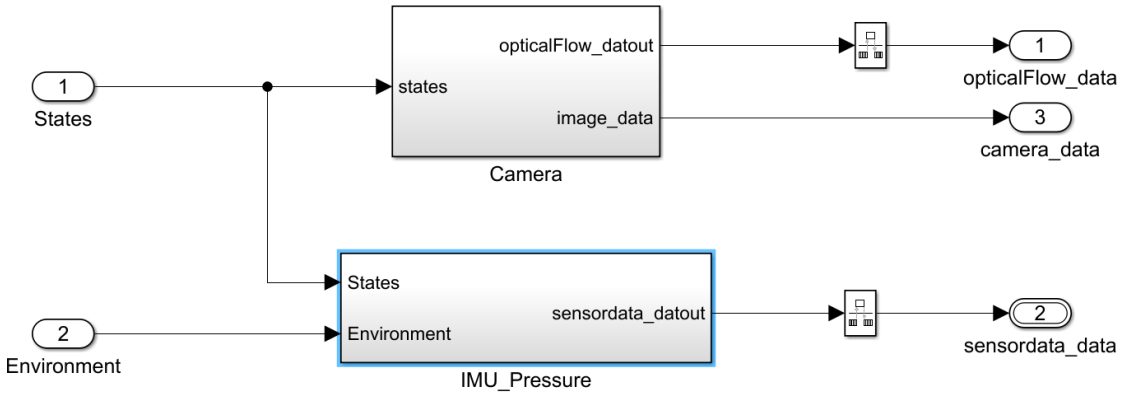


Figura 5.20: Dentro del bloque *Sensors (Dynamics)*

Se puede observar como un bloque se encarga de extraer información de la cámara y el otro de la IMU y de los sensores de presión.

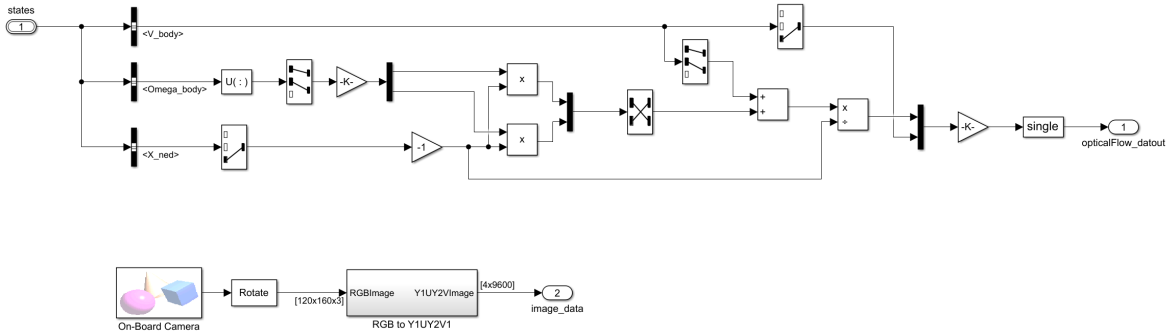


Figura 5.21: Dentro del bloque *Camera*

El bloque de la cámara produce los datos del flujo óptico y las propias imágenes como

se puede ver en la Figura *Camera*

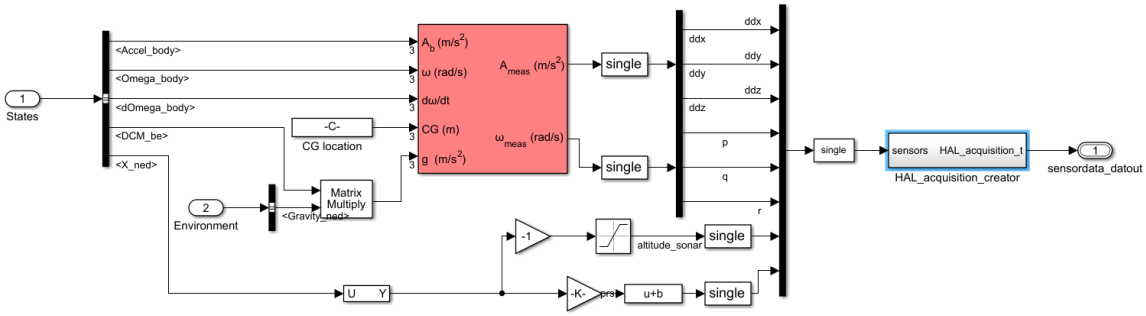


Figura 5.22: Dentro del bloque *IMU*

El bloque *IMU*, unidad de medición inercial, simula giroscopio y acelerómetro para determinar velocidades y aceleraciones. También incluye la simulación del sonar para la medición de la altura.

El último bloque del subsistema es la simulación de las banderas *Stop Simulation* (Figura 5.23), es muy sencillo simplemente se encarga de simular cuando una bandera se activa.

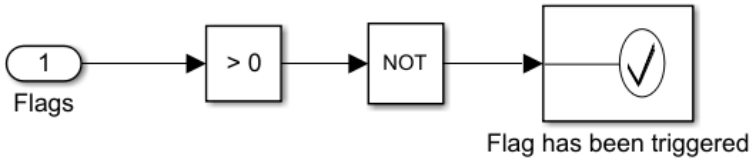


Figura 5.23: Dentro del bloque *Stop Simulation*

5.2.4 Visualización *Flight Visualization*

El último subsistema se encarga de realizar una representación en tiempo real de la simulación para poder visualizarla. Esto se realiza mediante la *Toolbox Simulink 3D Animation* que representa un modelo CAD del Parrot Mambo en la simulación.

En la Figura 5.24 se puede observar cómo toma los datos de los actuadores y del estado simulado del dron para realizar la representación visual. Incluye dos bloques *Extract Flight Instruments* y *Simulink 3D*, el segundo realiza la representación visual en sí y el primero representa los parámetros de vuelo mediante instrumentos.

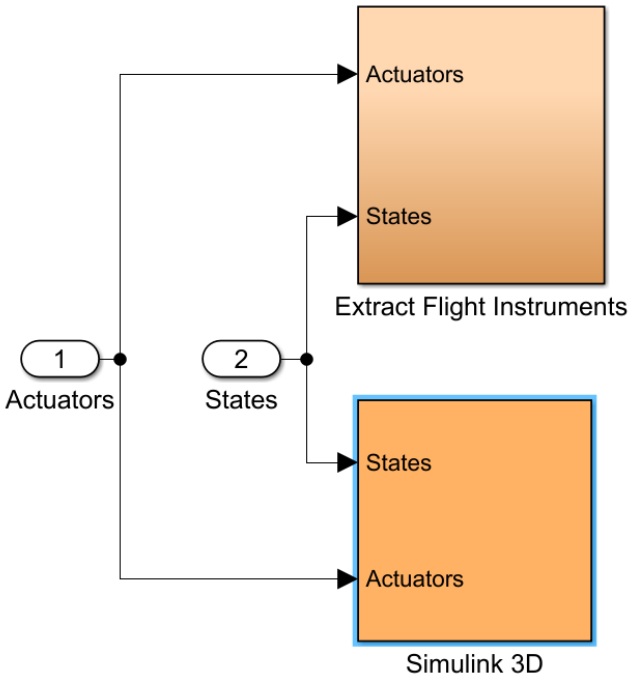


Figura 5.24: Dentro del bloque *Simulink 3D Animation*

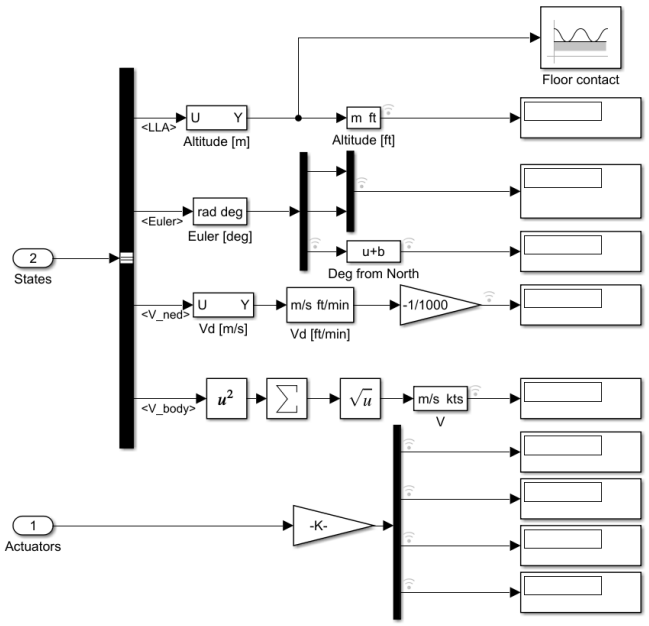


Figura 5.25: Dentro del bloque *Extract Flight Instruments*

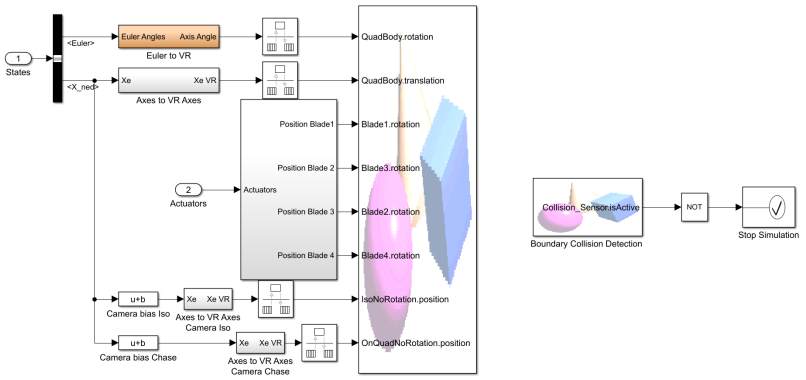


Figura 5.26: Dentro del bloque *Simulink 3D*

5.3 Simulación de Vuelo y Carga de código

En esta sección se presentará el método para realizar vuelos simulados y también como cargar el código desarrollado en el Parrot Mambo.

5.3.1 Simulación de vuelo

Para el trabajo en esta sección se hará uso de otro proyecto ofrecido por la *Toolbox Parrot Minidrones Support from Simulink* llamado *asbQuadCopter*.

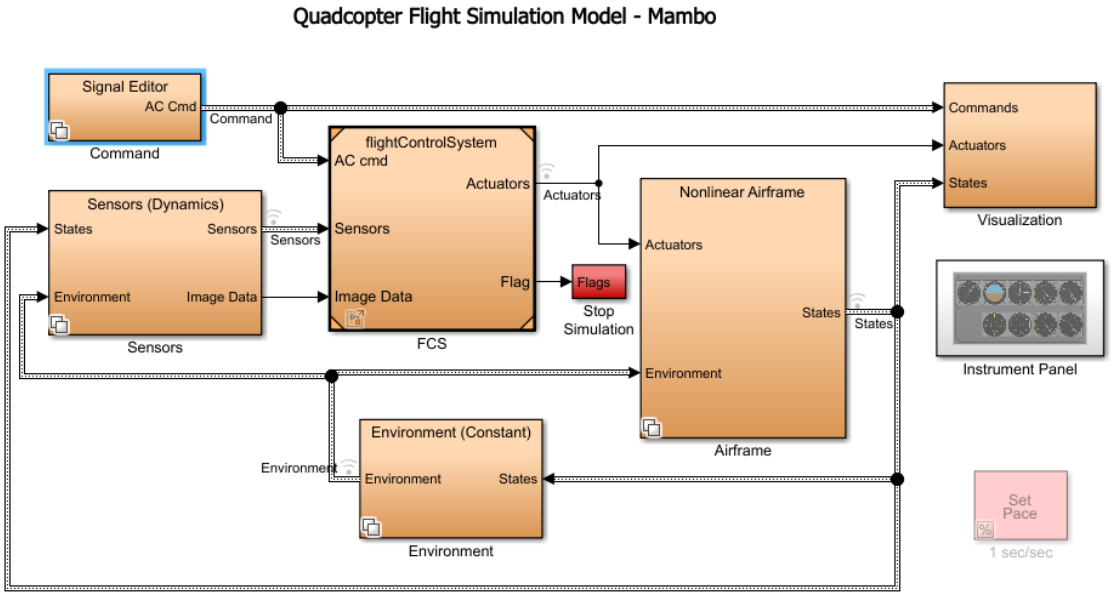


Figura 5.27: Proyecto *asbQuadCopter*

Este proyecto es muy similar al mostrado anteriormente, los subsistemas son los mismos, pero con un orden diferente. Este proyecto está diseñado para simular un vuelo donde el dron despegará y se mantendrá a una altura constante de 1 metro. Se puede modificar para que realice cualquier otro tipo de vuelo, pero como ejemplo el predeterminado es suficiente.

Para controlar los parámetros de la simulación se usará el menú *Simulation* de Simulink

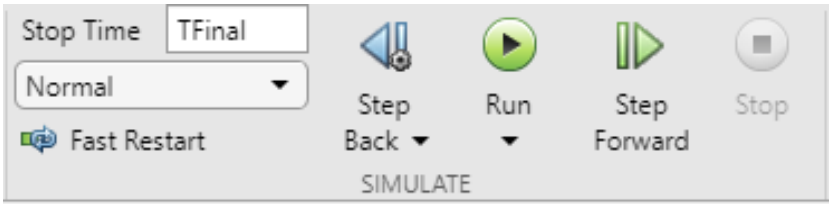


Figura 5.28: Menú *Simulation*

Para comenzar la simulación se hará clic en *Run*.

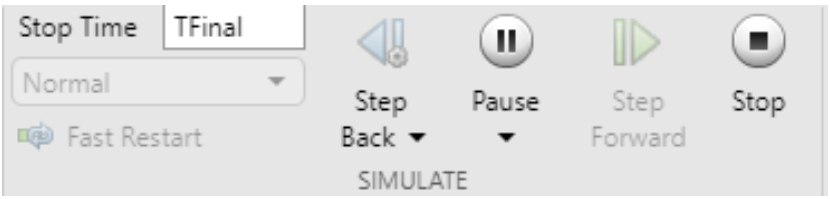


Figura 5.29: Menú *Simulation* durante la simulación

Durante la simulación podemos pausar la mediante *Pause* o detenerla completamente con *Stop*. También aparecerá una ventana con la representa visual de la simulación donde se puede observar el Parrot Mambo volando y una ventana con paneles de instrumentos que nos indica la altura, las revoluciones de los rotores etc.

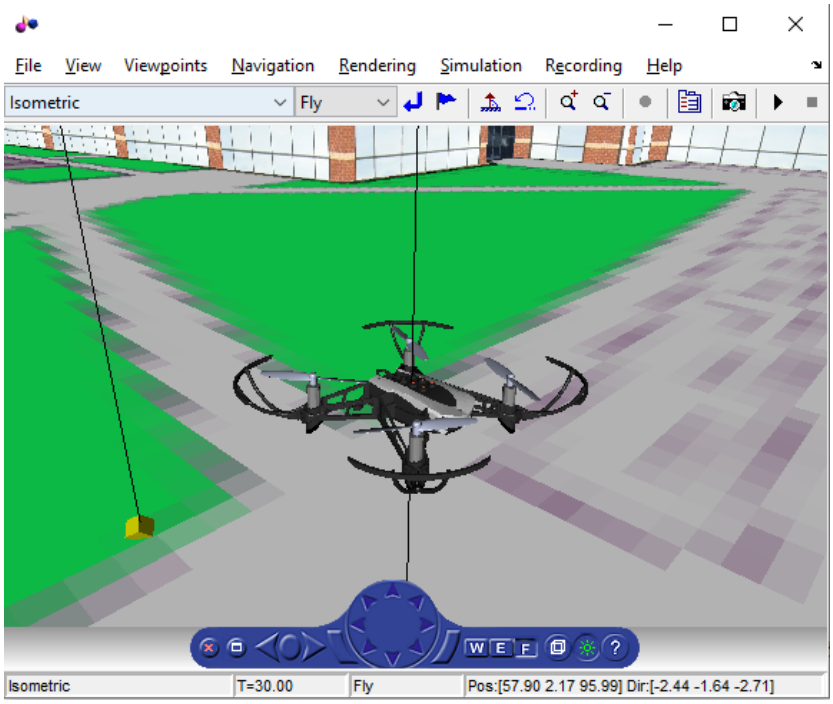


Figura 5.30: Representación de la simulación

Figura 5.31: Panel de instrumentos durante la simulación

Este proyecto además incluye un bloque que registra todos los parámetros de vuelo durante la simulación, *Logging* (Figura 5.32). El registro se realiza mediante la creación de variables en la memoria de Matlab llamadas *struct*, en este caso se guarda información del estado estimado del dron (*estim*), de los comandos de vuelo (*cmd*), calibración (*calib*), de la información de los sensores (*optical*, *sensor*), de la actuación de los rotores (*motor*) y de la posición de referencia (*posref*). Esto se puede observar en la Figura 5.33.

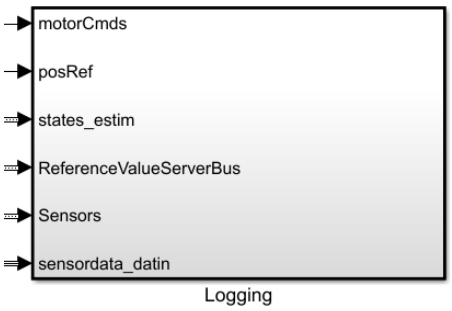


Figura 5.32: Bloque *Logging*

5.3.2 Carga de código

Para cargar el código de vuelo en el Parrot Mambo se hace clic derecho en el bloque *flightControlSystem* y se selecciona *Open As Top Model*. Como se ha explicado antes solamente se carga este bloque al *hardware* ya que el resto son para la simulación. En la ventana que se abre se navega hasta el menú *hardware*.

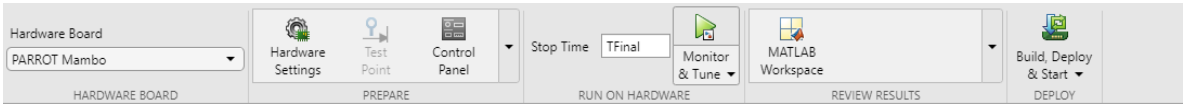


Figura 5.34: Menú *hardware* en Simulink

En el menú se puede observar como el *hardware* elegido es Parrot Mambo, para cargar el código se hace clic en la opción de *Monitor & Tune*. Tras la carga se abrirá el menú de control de vuelo.

Figura 5.35: Menú de control de vuelo

En este menú se puede elegir la potencia con el *slider PowerGain*, también se puede

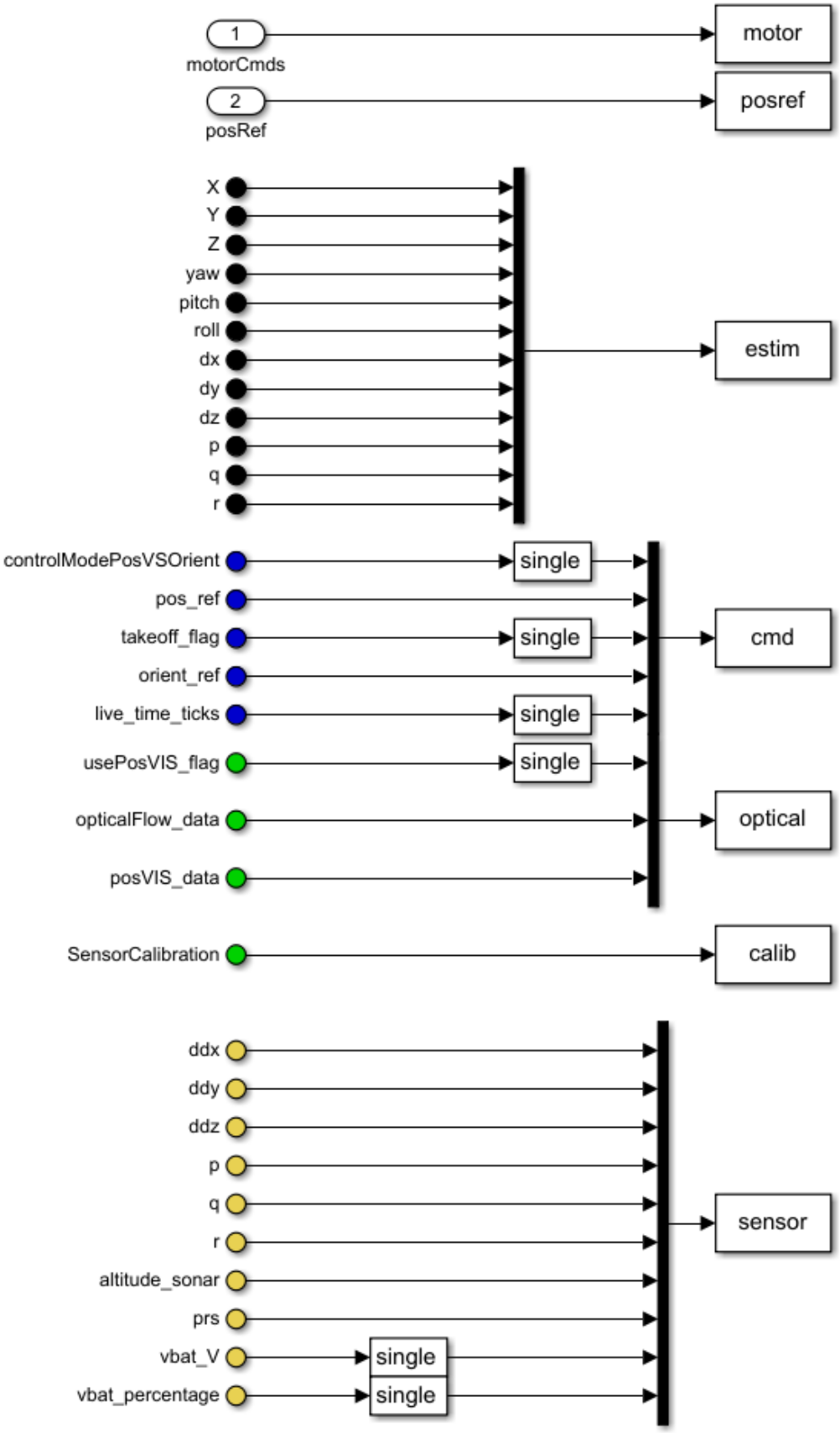


Figura 5.33: Dentro del bloque *Logging*

elegir utilizar el tiempo de simulación para el tiempo de vuelo. Para comenzar se hace clic en el botón verde *Start*.

Durante el vuelo el menú cambiará y mostrará la opción de *Stop* para para el vuelo. También nos informa de las opciones elegidas en el menú anterior y del estado del vuelo.

Figura 5.36: Menú de control de vuelo durante el vuelo

Cuando el vuelo acabe o lo detengamos el menú volverá a cambiar. Será similar al primero pero ahora añade los botones de *Flight Log* y *MAT File*. Ambos botones generan un fichero con los datos del último vuelo, el primero lo genera en extensión .txt y el segundo en .mat. También indicará si el vuelo se ha completado correctamente como en este caso, pero en caso de error también lo informará.

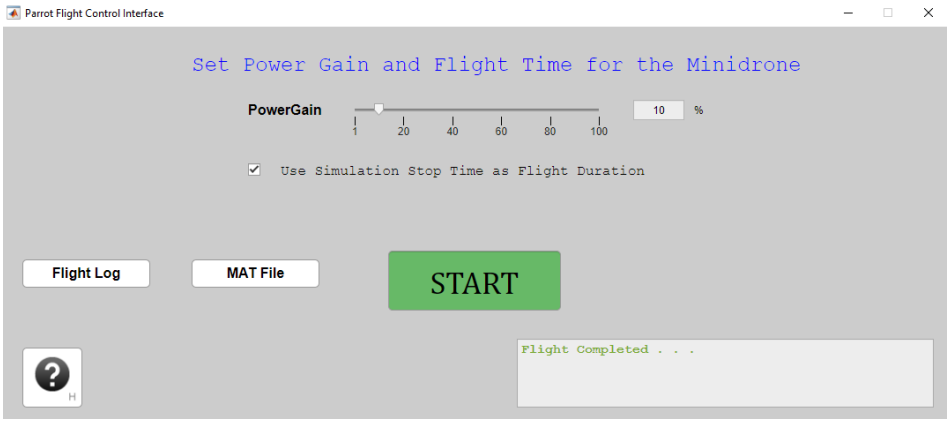


Figura 5.37: Menú de control de vuelo después del vuelo

Capítulo 6

Diseño de parámetros de control

En el Capítulo 4 se trató el problema de control de un cuadricóptero y de cómo era necesario un controlador que, conociendo el estado deseado y el estimado, comandase los rotores del mini dron. En el capítulo anterior se ha visto el bloque *Controller* que se dedica a ello, se encuentra dentro del subsistema de control de vuelo, toma como entrada los valores de referencia deseados y el estado estimado para producir los comandos directos a los motores.

Antes de mostrar el funcionamiento interno de dicho bloque primero se va a plantear cómo funciona el bucle de control utilizando, como ejemplo, la maniobra de vuelo a punto fijo a 0.7 metros de altura.

En este Capítulo se describirá el proceso llevado a cabo para diseñar los parámetros de control, así como los resultados en función de unos objetivos de diseño

6.1 Bucle de Control

Si se recuerda problema de control, se tenía que un dron recibía los comandos de los cuatro rotores y como resultado el dron desarrollaba una trayectoria de vuelo mediante una serie de fuerzas y momentos. En este caso dicha trayectoria es el despegue vertical para luego mantenerse en vuelo a punto fijo y la cuestión es cómo comandar los rotores para obtener dicho resultado. Basado en el trabajo de Brian Douglas en (8).

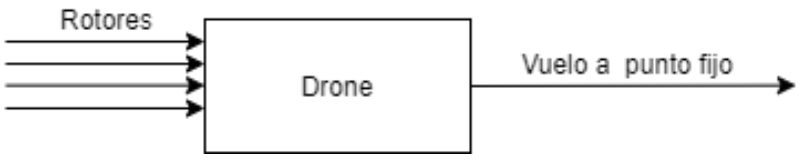


Figura 6.1: Esquema control básico

Ya que las velocidades de rotación de los motores no son parámetros evidentes a la hora de trabajar con movimientos en el espacio tridimensional se utilizaba el algoritmo de mezcla de los rotores (MMA) (Ecuaciones 4.1, 4.2, 4.3 y 4.4) para poder trabajar con empuje, alabeo, cabeceo y guiñada que son los parámetros de control.

Debido a la naturaleza de la maniobra propuesta se va a trabajar inicialmente solamente con el empuje. Se supone que el mini dron se encuentra paralelo al suelo, por lo que aumentar el empuje lo hará subir y reducir lo hará bajar. La altura a la que se encuentra el dron es conocida gracias a las sensorización del dron y también existe una referencia que indica la altura deseada (en este caso 0.7 metros).

La altura de dron y la deseada se restan, obteniendo el error en la altura. Este valor se provee a un controlador de altura que se encarga de aumentar o disminuir el empuje en función del error que le llega. Esto es un controlador PID y en este capítulo se tratará su ajuste.

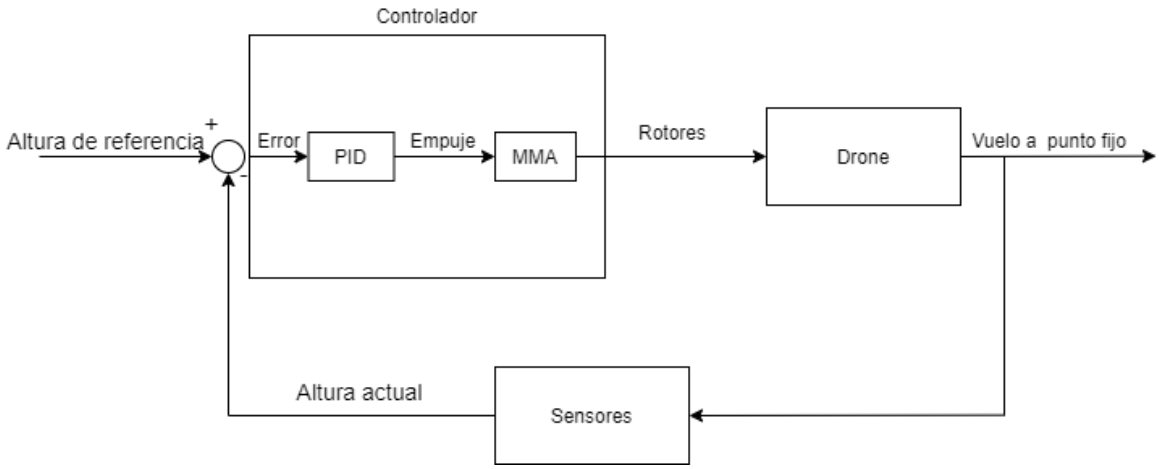


Figura 6.2: Esquema de control para la altura

Como se mostró en el Capítulo 4 el empuje, alabeo, cabeceo y guiñada se pueden controlar de forma independiente, por tanto en el controlador del minidron existirá un PID para cada uno de ellos.

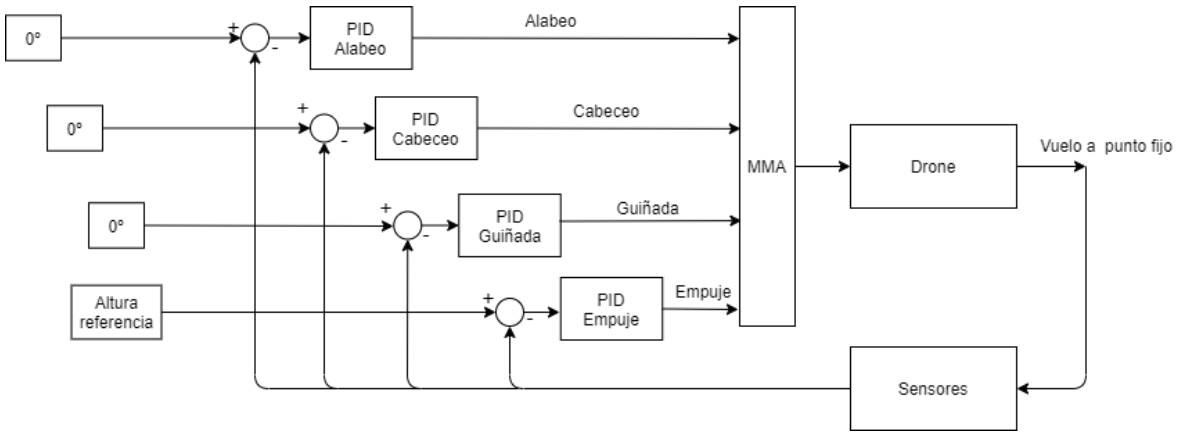


Figura 6.3: Esquema de control para la altura añadiendo controladores de alabeo, cabeceo y guiñada

En este caso la referencia de estos ángulos es 0 ya que el dron está equilibrado y no hay desplazamientos en el plano horizontal (XY).

Si una racha de viento afecta al mini dron, esto desestabilizará los ángulos de control y los controladores se encargarán de hacer que vuelvan a 0. Se podría suponer que el control es satisfactorio, sin embargo, durante esta desestabilización el mini drones se moverá en el plano horizontal y con la configuración actual el mini dron no corregiría este error de posición.

Por ello es necesario actualizar el controlador de forma que se tenga en cuenta la posición en el plano horizontal (XY). El alabeo y cabeceo son los ángulos que controlan dicho movimiento e inicialmente se ha supuesto que para esta misión deben ser 0, pero ya se ha visto que se puede dar el caso donde esto no es cierto.

Por lo tanto, resulta más interesante que el control se realice mediante una posición de referencia con las coordenadas X e Y. Ahora no se especifican los ángulos de cabeceo y alabeo de forma directa, sino que se hace de forma indirecta a través de la posición de referencia. Por ello los controladores de estos dos ángulos se deben mantener, para seguir corrigiendo la diferencia entre los ángulos estimados y los deseados.

En la Figura 6.4 se muestra un esquema del controlador ya completado. Se puede ver que ahora la referencia son las coordenadas X e Y que en este caso se suponen 0 que representan el punto de despegue. Esta referencia va a un controlador de posición que, primero, realiza un cambio de referencia entre el sistema inercial al fijo al dron y después mediante 2 controladores PID pasa del error de posición a los valores deseados de alabeo y cabeceo. Estos valores finalmente van sus controladores como hacían en el esquema anterior.

De esta forma los controladores del desplazamiento horizontal y los controladores de los ángulos de alabeo y cabeceo están acoplados al igual que su dinámica.

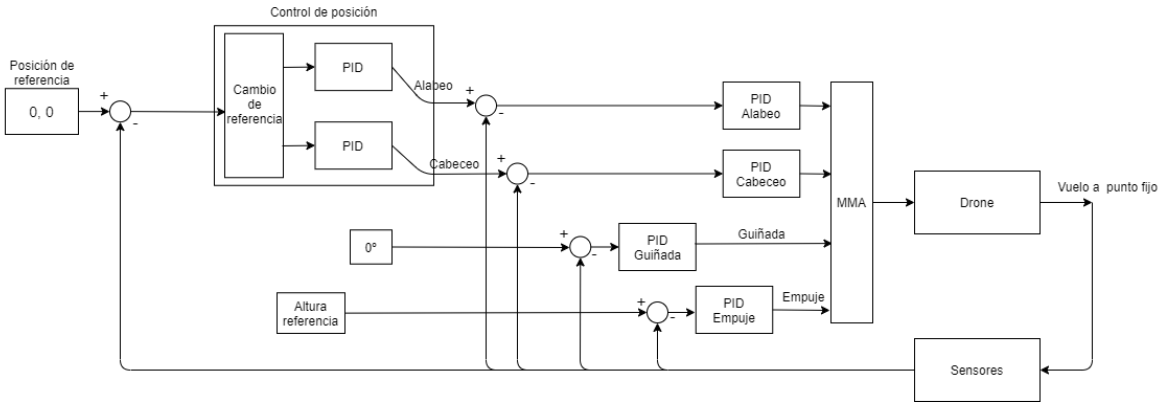


Figura 6.4: Esquema de control propuesto completado

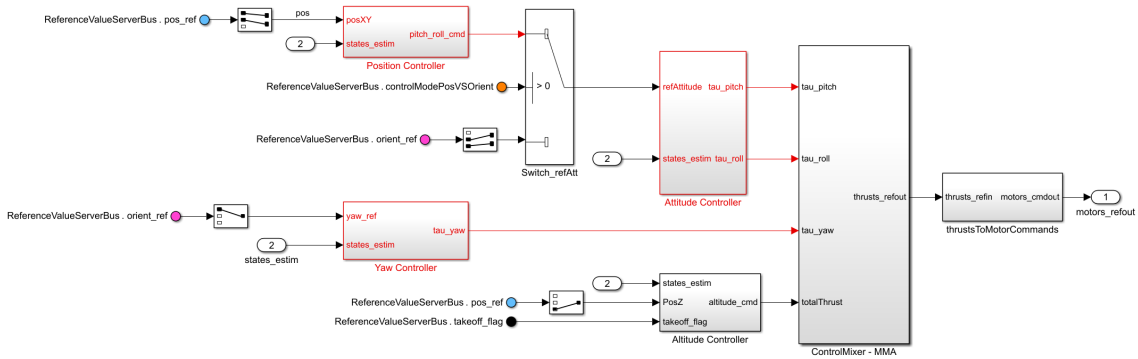


Figura 6.5: Esquema de control en el modelo del Parrot Mambo de Simulink

Este esquema de control ahora si es correcto, pero para que funcione de forma satisfactoria es necesario que los 6 controladores PID están bien calibrados. En la siguiente sección se llevará a cabo esta tarea.

La Figura 6.5 muestra el esquema de control dentro del bloque *Controller* que se encuentra dentro del subsistema de control de vuelo del proyecto explicado en el Capítulo anterior. Como se puede observar es muy similar al esquema mostrado. No se muestra como un control cerrado, aunque si lo es, ya que las entradas provienen de canales de información que se ven influidas por los cálculos aquí realizados.

También se observa cómo hay 4 bloques de controladores: altura, posición, guiñada y orientación, el de posición alimenta al de orientación como se había explicado antes. Como entrada de cada bloque se tiene el estado estimado marcado con un 2 que vienen del estimador de estados y además los valores de referencia procedentes del controlador.

Las Figuras 6.6, 6.7, 6.8 y 6.9 muestran el control de guiñada, altura, orientación y posición respectivamente.

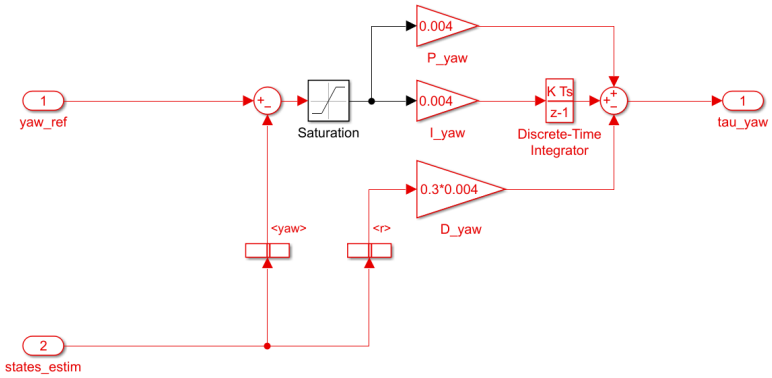


Figura 6.6: Dentro del bloque *Yaw Controller*

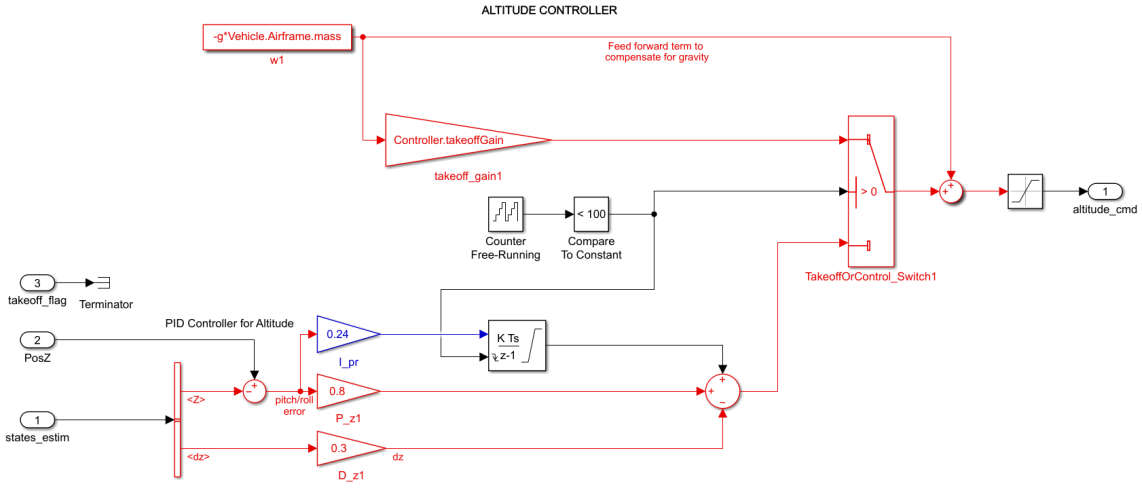


Figura 6.7: Dentro del bloque *Altitude Controller*

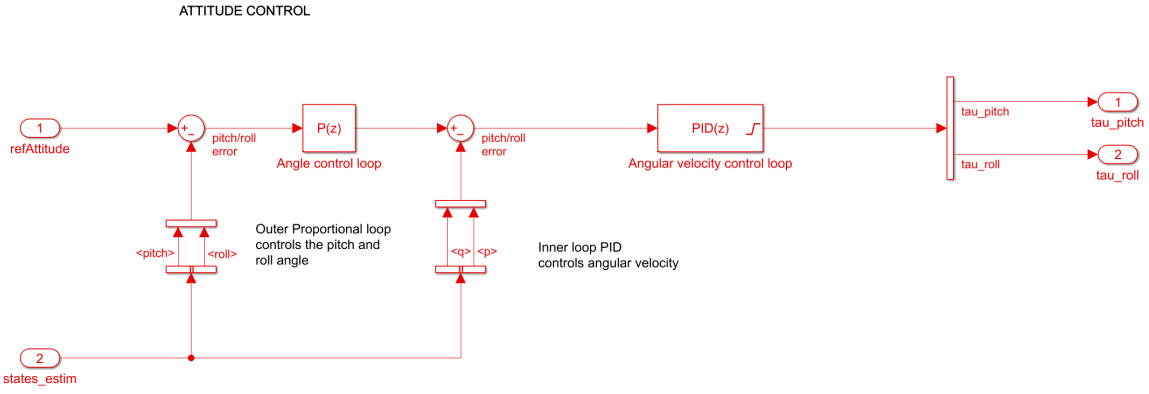


Figura 6.8: Dentro del bloque *Attitude Controller*

El control de alabeo y cabeceo, posee dos bucles uno externo que controla los ángulos y uno interno que controla la velocidad angular.

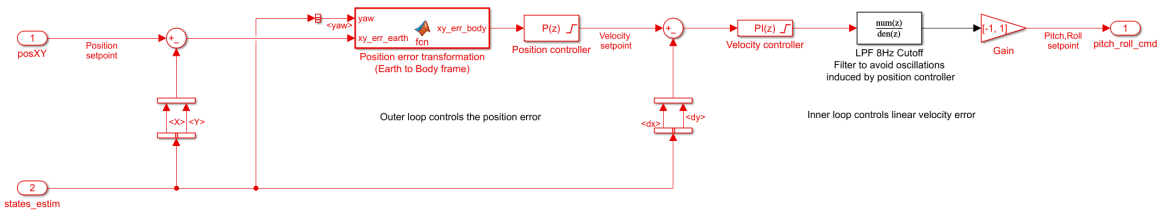


Figura 6.9: Dentro del bloque *Position Controller*

El control de posición, como el anterior posee bucle externo e interno. El primero control realiza el cambio de referencia y controla el error de posición y el segundo controla la velocidad lineal.

Además, si se regresa a la Figura 6.5 se ve como todos los comandos de control terminan en el bloque *Control Mixer - MMA* que contiene el algoritmo de mezcla de los rotores MMA visto anteriormente. Este recibe los valores de empuje, alabeo, cabeceo y guiñada y los transforma a comandos de velocidad de rotación de los rotores (Figura 6.10).

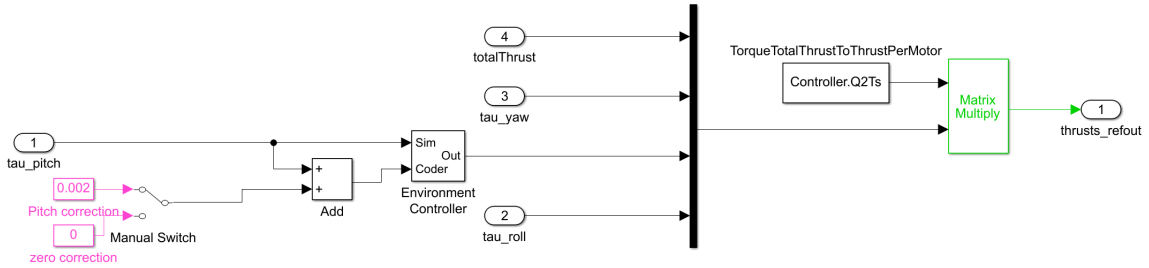


Figura 6.10: Dentro del bloque *Control Mixer - MMA*

6.2 Linealización

Ahora que el esquema de control está claro se va a realizar el ajuste de los controladores PID previamente mostrados. Los controladores PID están compuestos por tres parámetros distintos llamados ganancias: proporcional (Kp), integral (Ki) y derivativo (Kd), no todos los controladores son PID, pueden ser solamente P, PI o PD según que parámetros se utilicen.

Un mal ajuste de los PID pude hacer inestable el vuelo y provocar colisiones que pongan en peligro el *hardware*. Como ya se dispone de un modelo del Parrot Mambo que se

puede simular, una idea lógica seria realizar los ensayos de ajuste de las ganancias en el simulador.

Esta idea presenta un inconveniente, el modelado del Parrot Mambo no es perfecto (ningún modelado de un sistema físico lo es) y los resultados de las simulaciones no serán idénticos a los ensayos reales. Sin embargo, el modelo es suficientemente bueno como para ser usado de plataforma inicial de ajuste, pero en algún momento se deberá realizar una verificación de los resultados mediante ensayos en el *hardware*.

Para poder ajustar controladores PID se requiere trabajar sobre un sistema lineal. Aquí surge un problema ya que el modelado actual es no lineal, ya que incluye operaciones lógicas, sistemas condicionales y ecuaciones no lineales.

Por ello, primero se debe llevar a cabo una linealización del sistema, con el objetivo de obtener un modelo lineal. El procedimiento para ello será la eliminación o adaptación de los sistemas no lineales presentes en el modelo.

De esta forma se dispondrá de un modelo lineal, que nos permite utilizar las herramientas de ajuste de controladores y otro no lineal que se utilizará para realizar simulaciones y comprobar dichos ajustes.

Siguiendo el ejemplo utilizado anteriormente, se mostrará el proceso para linealizar el bucle de control de la altura para una misión de despegue y vuelo a punto fijo.

Primero, se eliminarán los controladores PID de posición, guiñada y orientación y se les dará un valor fijo de 0 a sus respectivos momentos ya que, como se ha visto en el ejemplo anterior, bajo las suposición de que no hay ningún agente desequilibrante como el viento es correcto usar esta configuración.

El siguiente paso consiste en realizar la suposición de que el ruido y la dinámica de los sensores no afecta a la estimación de la altura del dron. Esto significa que el dron conoce con total precisión a que altura se encuentra en cada instante. Para llevar cabo esta suposición, la entrada de estado estimado del controlador de altura será sustituida por el valor procedente directamente del modelo del cuadricóptero.

La entrada de valor de referencia en el controlador ya no vendrá determinada por la lógica de vuelo, sino que se le dará valor constante.

Para comprobar el ajuste primero se va a observar el control actual de la altura con el modelo no lineal, para ello se realizará la maniobra de despegue para mantenerse en vuelo punto fijo a 0.7 metros. El resultado se puede observar en la Figura 6.11.

Como se puede observar el sistema se estabiliza en los 0.7 metros, pero al principio sube hasta casi los 0.9, esto es lo que se conoce como *overshoot* y como se verá más delante en el ajuste busca eliminarlo.

El modelo linealizado se puede ver en la Figura 6.12.

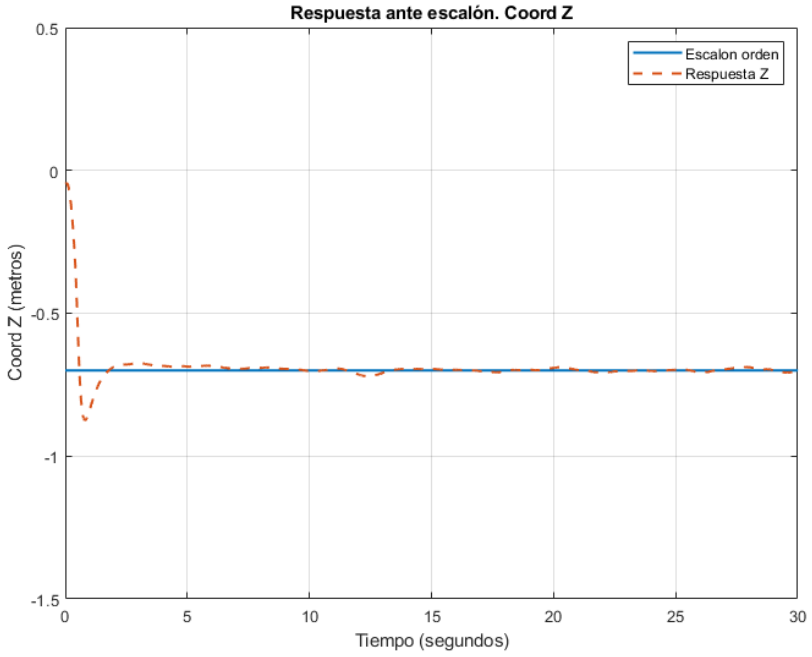


Figura 6.11: Respuesta del sistema no lineal. Debido al sistema de referencia, el eje Z negativo es hacia arriba

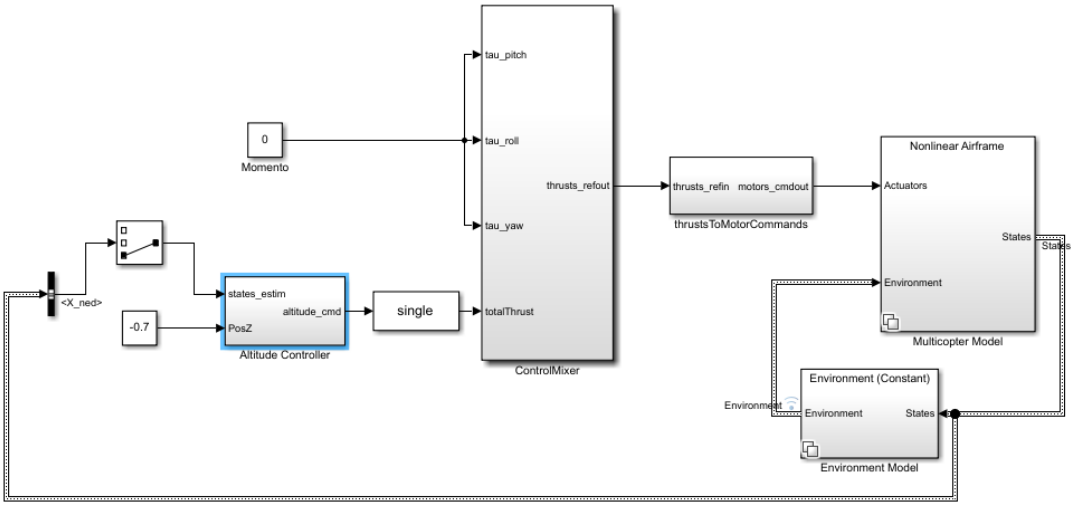


Figura 6.12: Modelo linealizado

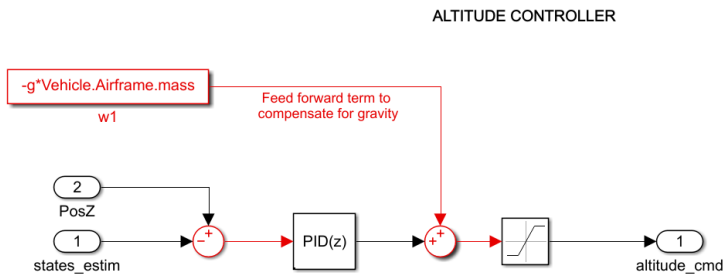


Figura 6.13: Dentro del bloque *Altitude controller*

En el modelo linealizado solo ha quedado el controlador de altitud, el MMA, el modelo del cuadricóptero y el ambiental. El sistema de visualización también ha sido eliminado ya que los resultados se estudiarán mediante los datos extraídos de la simulación y no se necesario la visualización del ensayo.

El nuevo controlador de altura se observa en la Figura 6.13 donde los bloques de las ganancias ahora se han sustituido por un bloque PID. Esto permite hacer uso de las herramientas de ajuste lineal como se verá a continuación. En la Figura 6.14 se puede ver el panel de configuración del bloque PID, aquí es donde se encuentran las ganancias P,I,D.

En la parte inferior se puede ver un apartado llamado *Automated tuning* que se podría traducir como ajuste automático. Pulsar el botón *Tune...* abrirá la aplicación *PID Tuner* (Figura 6.15) que es la herramienta que se va a utilizar en este caso para el ajuste del PID.

En línea discontinua se pude ver la respuesta con los valores configurados actualmente, como se puede ver es muy similar a la vista en la Figura 6.11. En línea continua nos muestra la respuesta de la configuración que estamos modificando mediante la aplicación que se actualizará cada vez que se altere uno de los parámetros de control. Gracias a esta herramienta se puede ajustar el controlador de una manera visual y rápida.

Una vez se haya configurado la respuesta se tomarán los datos de la ganancias proporcionales, derivativas e integrales y se llevaran al modelo no lineal para comprobar su desempeño en la simulación. El resultado con los nuevos parámetros de ajuste del controlador se puede ver en la Figura 6.16

Si se compara con la Figura 6.11 se puede observar que el *overshoot* se ha eliminado y el sistema tarda un poco más en alcanzar la altura de diseño.

Para diseñar los controladores se seguirán los siguientes criterios:

- **Estabilidad**. Siempre debe ser el criterio principal en el ajuste de cualquier con-

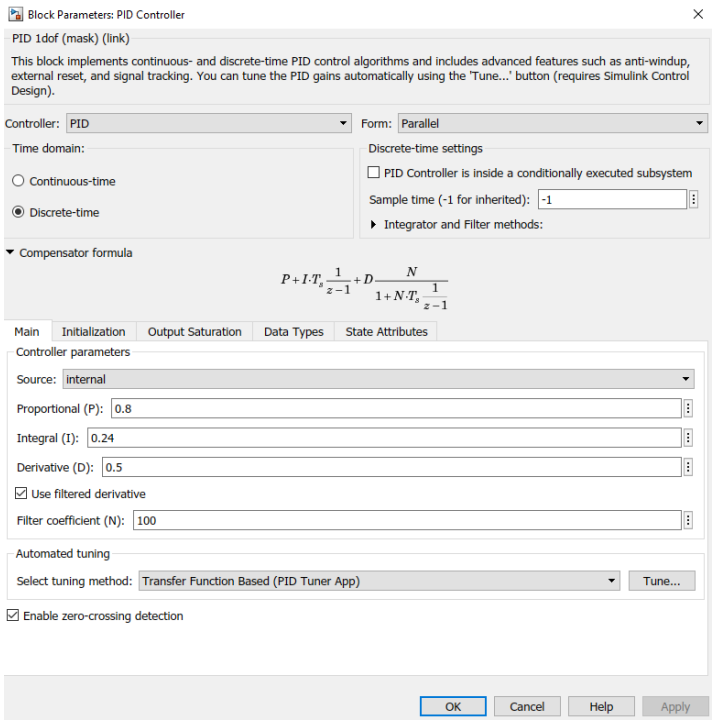


Figura 6.14: Panel de configuración del bloque PID

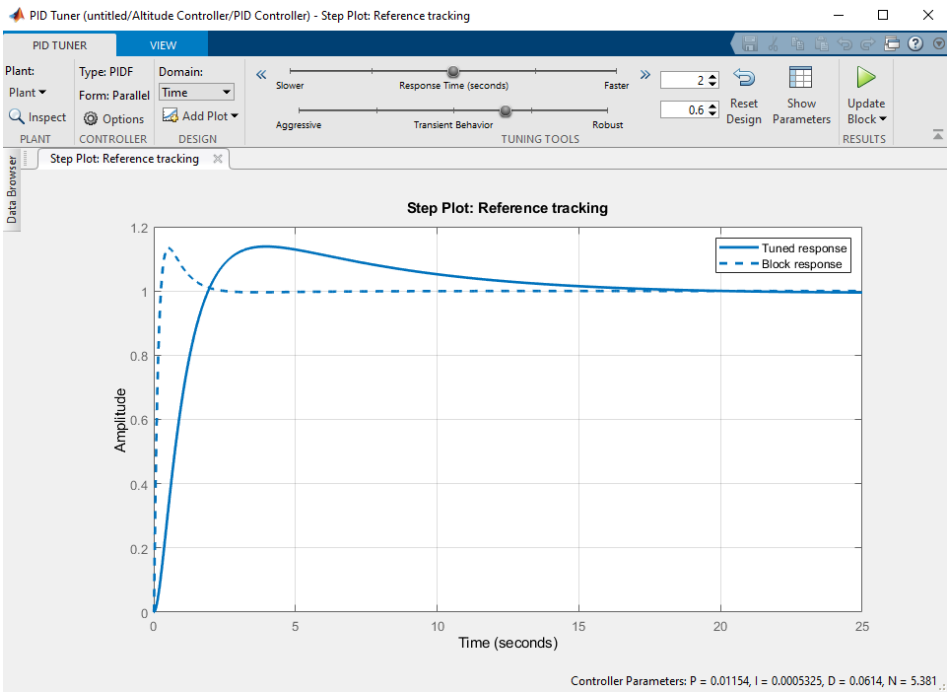


Figura 6.15: Aplicación PID Tuner. En el menú superior podemos ver las opciones de control para el ajuste del PID, se puede elegir la planta simulada, el tipo de controlador, el dominio y dos *slider* que ajustan la respuesta en base al tiempo de respuesta y el periodo transitorio. Abajo a la derecha se pueden ver las ganancias del PID para la configuración elegida.

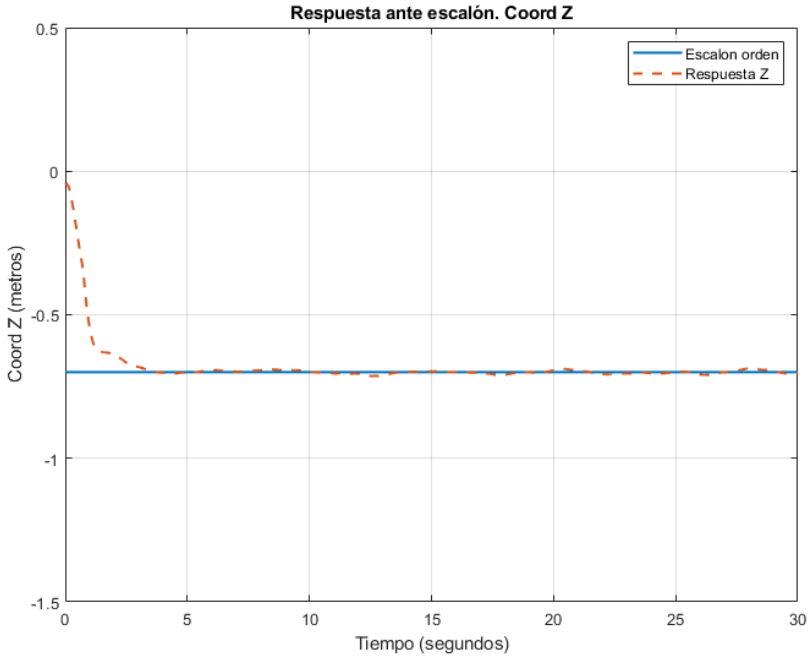


Figura 6.16: Respuesta del sistema no lineal con los nuevos parámetros de ajuste

trolador. Un sistema es estable cuando alcanza y mantiene en el tiempo el estado deseado.

- **Reducción del *overshoot***. Es el objetivo de diseño principal, disminuir o eliminar el *overshoot*, esto que significa que el sistema no se excede ante el estímulo. En este caso es importante para la aplicación donde son preferibles los movimientos lentos y precisos a los rápidos e inexactos.

- **Reducción de la oscilación**. Muy similar al objetivo anterior, la oscilación introduce mucha inexactitud temporal en la respuesta y en este caso lo mejor es evitarla.

- **Tiempo de respuesta contenido**. El tiempo de respuesta se puede definir como el tiempo que tarda el sistema en alcanzar el valor final. Como consecuencia de los objetivos anteriores el sistema tenderá a tener un tiempo de respuesta alto, el objetivo aquí es que este no se dispare.

6.2.1 Resultados Simulación

En este apartado se van a mostrar los resultados del ajuste de los controladores PID en la simulación. Para ello se introducirá una excitación al sistema en forma de escalón y se observará la respuesta del sistema. El escalón se introducirá en la dirección

Z (despegue/desplazamiento vertical), dirección X (movimiento frontal), dirección Y (movimiento lateral) y cambio de guiñada.

Cada excitación se realizará en un ensayo independiente a los demás.

La respuesta se observará mediante una serie de gráficas, en ellas se mostrará la respuesta junto al escalón para los casos donde es introducido. A su derecha se mostrará su velocidad, que será lineal en caso de ser un desplazamiento o angular en el caso de un cambio de orientación.

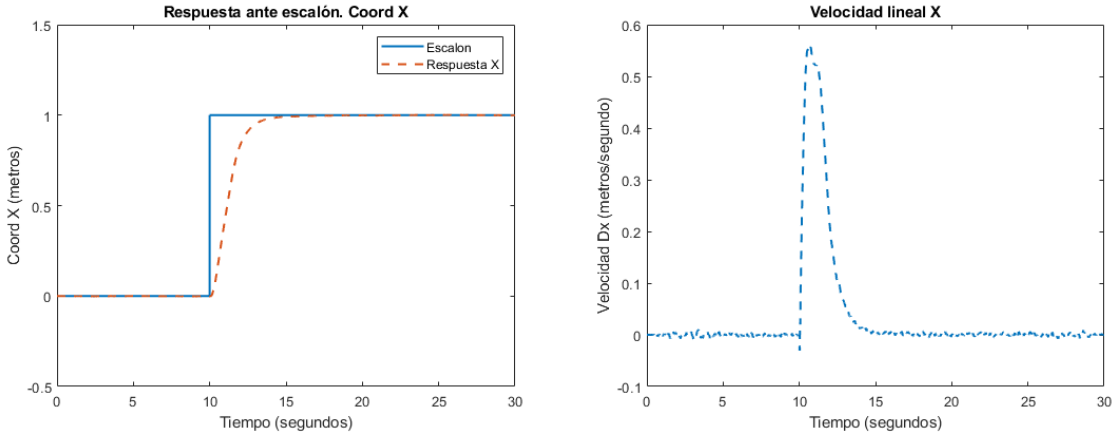


Figura 6.17: Respuesta ante escalón en el eje X. A la izquierda la posición X, a la derecha su velocidad lineal

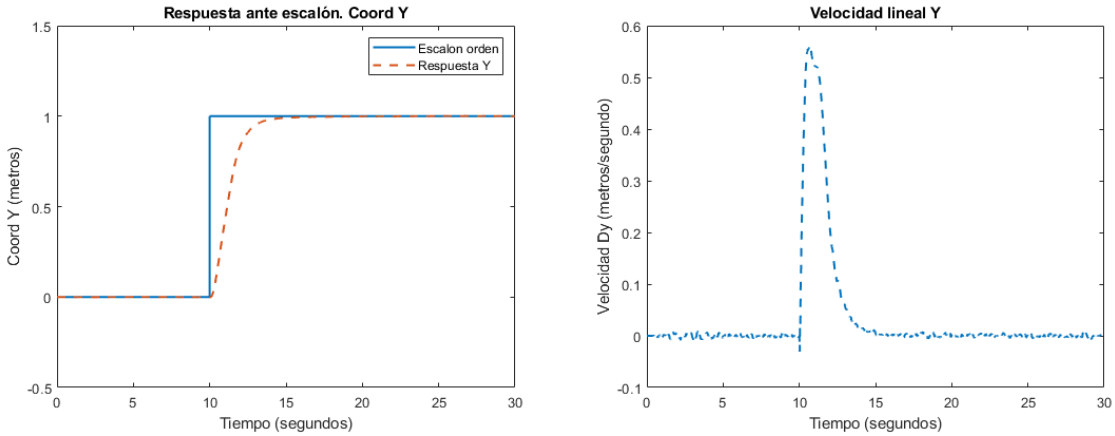


Figura 6.18: Respuesta ante escalón en el eje Y. A la izquierda la posición Y, a la derecha su velocidad lineal

Las respuestas en los ejes X e Y son iguales ya que comparten los mismos controladores

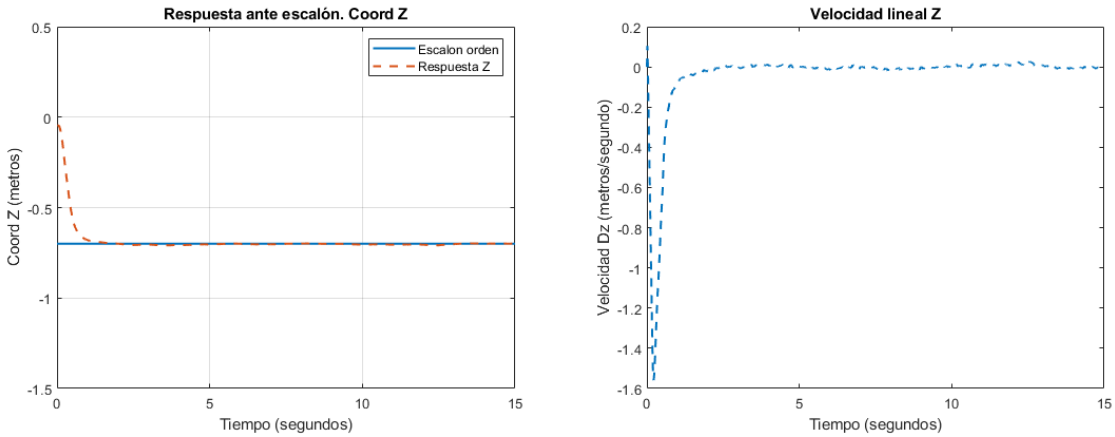


Figura 6.19: Respuesta ante escalón en el eje Z. A la izquierda la posición Z, a la derecha su velocidad lineal

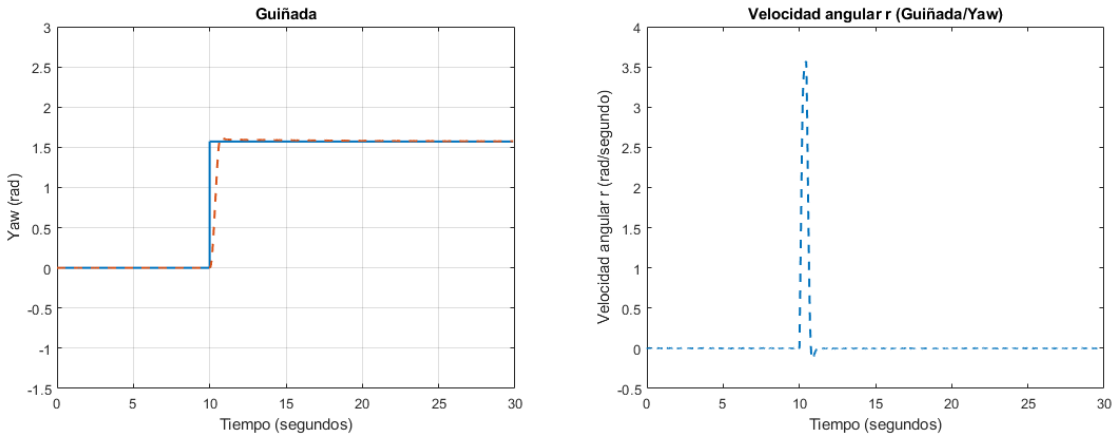


Figura 6.20: Respuesta ante escalón en el ángulo de guiñada. A la izquierda la guiñada, a la derecha su velocidad angular

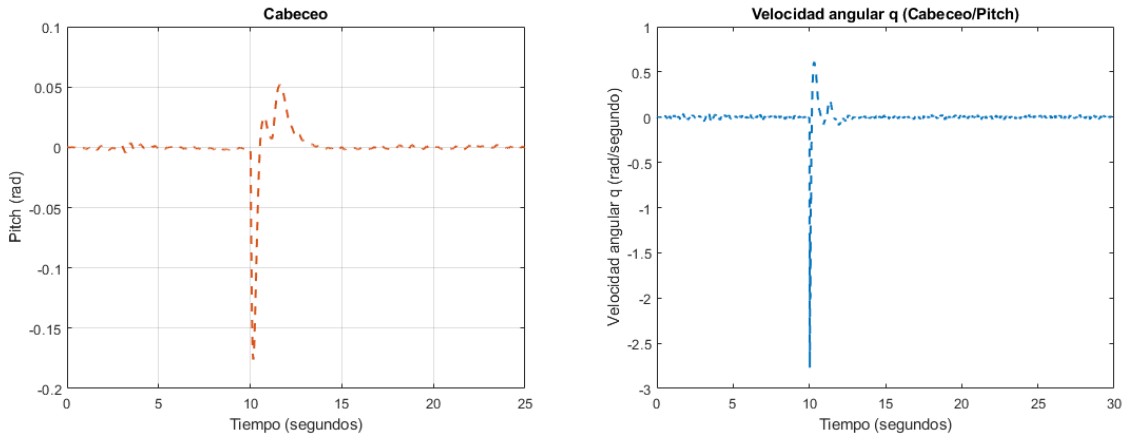


Figura 6.21: Respuesta ante escalón en el eje X. A la izquierda el cabeceo, a la derecha su velocidad angular

Las gráficas de respuesta del cabeceo y alabeo se han obtenido de los ensayos de movimiento en el eje X e Y respectivamente.

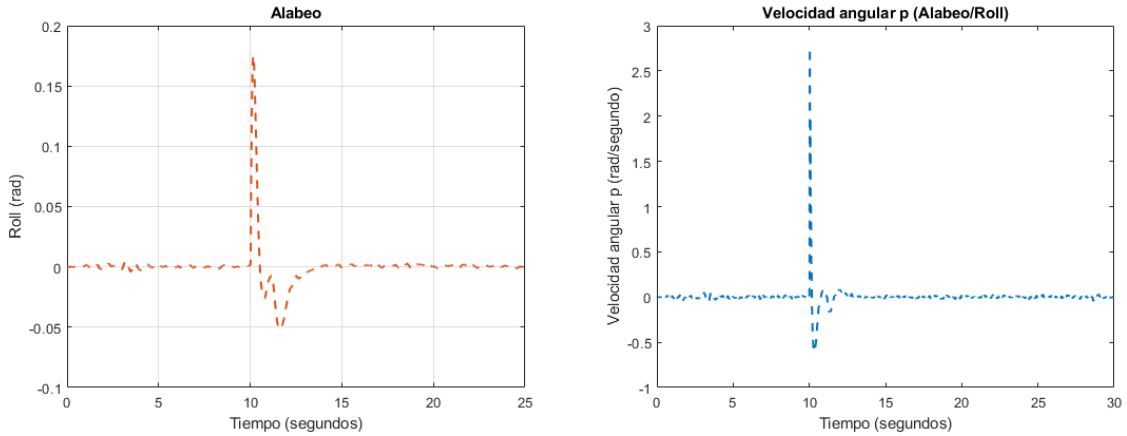


Figura 6.22: Respuesta ante escalón en el eje Y. A la izquierda alabeo, a la derecha su velocidad angular

En las gráficas se puede observar que se han logrado los objetivos de diseño. También muestran como el comportamiento simulado del Parrot Mambo es bastante ideal, el mejor ejemplo de ello es la respuesta en la guiñada, en ella se observa como el sistema se ajusta al escalón de forma casi perfecta.

Esto no resulta un gran problema ya que el ajuste realizado en simulador no es el definitivo y hace falta verificar estos comportamientos en el *hardware*. Este procedimiento ha permitido obtener unos valores de ajuste preliminares que se sabe que tienden hacia los objetivos deseados, ahora falta realizar un ajuste más refinado mediante pruebas de campo.

6.3 Ensayos de circuito en el *hardware*

Para verificar los ajustes del simulador y elegir una configuración final se ha diseñado un ensayo en el que el Parrot Mambo realizará un circuito. El seguimiento del circuito se realizará mediante una serie de *waypoints*.

Las consideraciones del ensayo son las siguientes

- El dron comenzará el recorrer el circuito a los 5 segundos de despegar.
- El circuito se seguirá a una altura constante de 0.7 metros.
- El dron cambiará su guiñada para orientarse siempre en la dirección del siguiente *waypoint*.
- Cuando se complete el circuito el dron aterrizará.
- Existe un radio de giro de 0.3 metros, esto significa que el Parrot Mambo comenzará a viajar al siguiente *waypoint* cuando alcance una distancia de 0.3 metros del *waypoint* que esté siguiendo en ese momento.
- Habrán 6 *waypoints*, un dibujo esquemático se muestra en la Figura 6.23 con las coordenadas de posición de cada *waypoint* (X,Y)

Para llevarlo a cabo se usará como base el proyecto de Simulink *parrotMinidroneCompetition*, este es el mismo que el usado en el Capítulo 5. Las indicaciones de movimiento se realizarán mediante el bloque *Path Planning* (Figura 6.24) del subsistema de control de vuelo.

En la Figura 6.25 se muestra la lógica de vuelo para realziar el recorrido. En el recuadro naranja superior se observa la lógica de aterrizaje, debajo está el bloque *Waypoint follower* que genera las señales de posición y orientación conforme se va realizando el recorrido. Mediante bloques *To Workspace* se extraen los datos de posición y guiñada al ser estos los parámetros de estudio.

En la Figura 6.26 se muestra el bloque *UAV Waypoint Follower* que recibe los datos de la posición y orientación actual desde los sensores mediante la entra *Pose*. También recibe la serie de *waypoints* del recorrido y un parámetro llamado *look ahead distance* que marca la velocidad a la que se desplazamiento.

Tiene varias salidas, las que se utilizan en este caso son 3. La primera es *LookaheadPoint* que va marcando el siguiente punto que seguir. La segunda *DesiredYaw* es la orden de guiñada que también se actualiza en cada tramo entre *waypoints* y la última *Status* marca el estado de finalizado cuando termina el recorrido.

Esta última variable de control va a la lógica de aterrizaje y le indica que comience dicha maniobra. Finalmente los valores de orientación y guiñada se van alimentado al canal de comandos de referencia conforme avanza el recorrido.

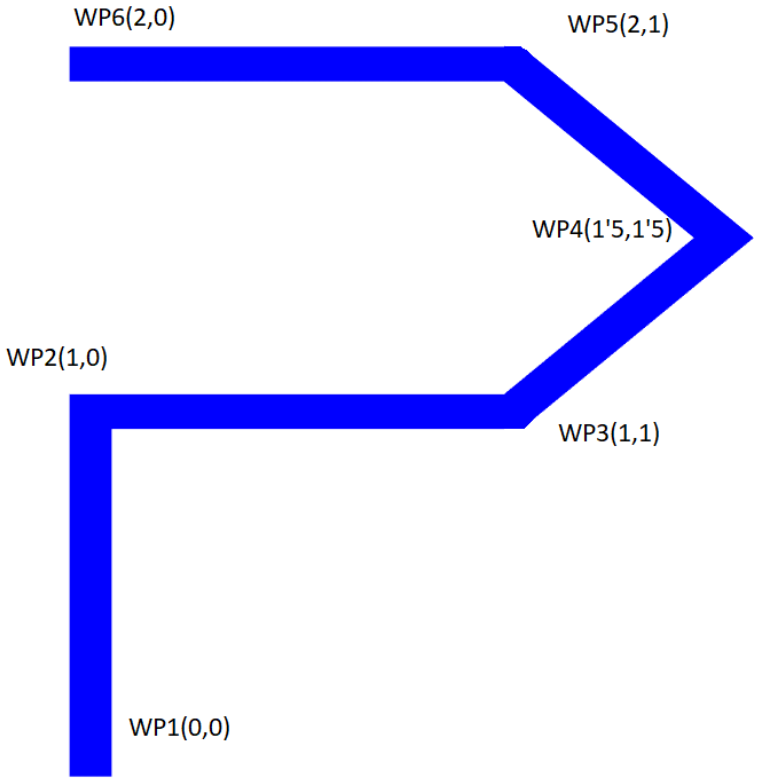


Figura 6.23: Esquema del circuito del ensayo

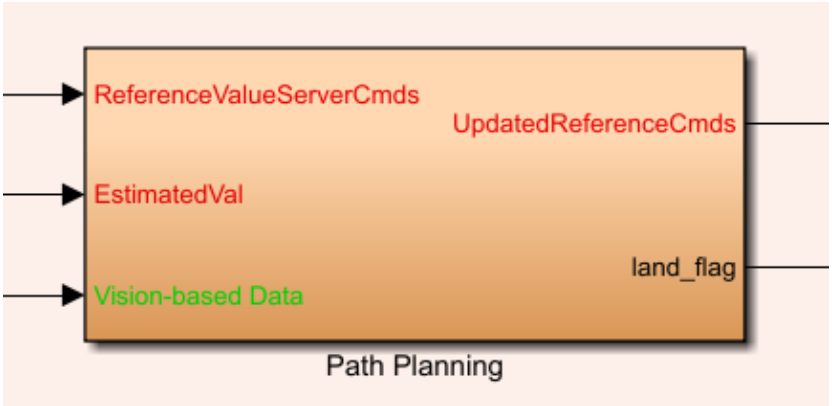


Figura 6.24: Bloque que aloja las órdenes de vuelo

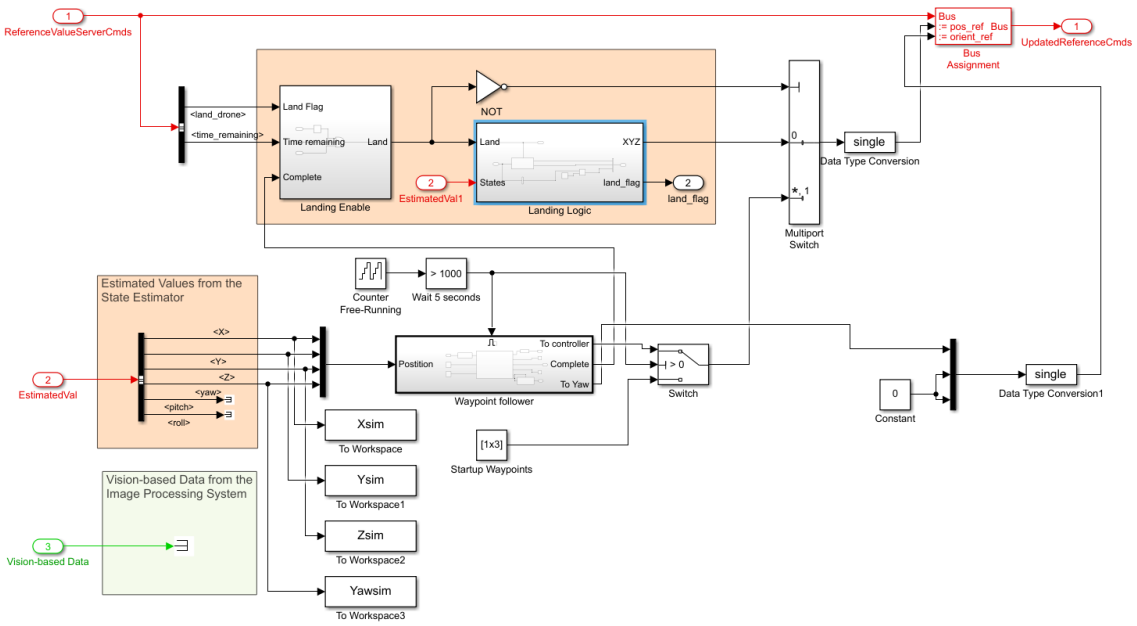


Figura 6.25: Dentro del bloque *Path Planning*

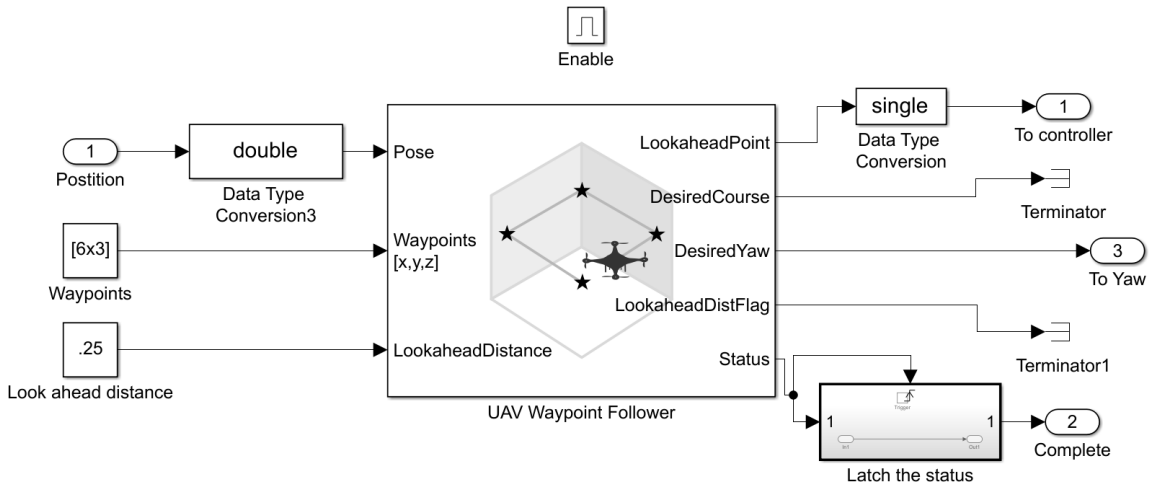


Figura 6.26: Dentro del bloque *Waypoint follower*

6.4 Resultados

Tras verificar que el ajuste de los controladores procedente del simulador es estable en el *hardware* se procede a realizar ensayos con el circuito explicado.

Estos ensayos servirán para realizar un ajusta más fino con el objetivo de obtener el mejor ajuste para vuelos con el *hardware*, ya que la configuración óptima en el simulador no tiene por qué ser la mejor en el Parrot Mambo. Este ajuste más fino se llevará a cabo mediante la variación de los parámetros de ajuste en cada ensayo, esto se hará de forma sistemática con todos los parámetros disponibles. A partir de estos cambios se estudiará si mejoran el desempeño del Parrot Mambo o, por el contrario, lo empeoran.

Las variaciones realizadas son arbitrarias y sus tendencias se han elegido en función de los resultados observados conforme se iban obteniendo. Sin embargo, como norma general las variaciones iniciales de cada parámetro consistían en aumentos y disminuciones del rango de 20 %-50 %. Según la influencia de estos cambios preliminares, se decidían las variaciones posteriores.

Para comprobar el desempeño de cada vuelo se toma los datos de posición y guiñada en cada vuelo, estos se compararán entre ellos mediante el uso de gráficas. Para determinar el ajuste definitivo se usarán estos datos, pero también se tendrá en cuenta la observación del vuelo realizada durante el mismo.

Se dispone de los datos registrados de un total de 57 ensayos, cada uno con un ajuste distinto como ya se ha comentado. A modo de muestra los resultados de algunos de ellos se pueden encontrar en el Apéndice B. El número total de ensayos es mucho mayor, pero en aquellos casos donde se veía de forma evidente que el vuelo era deficiente o directamente inestable no se tomaban sus datos.

Las siguientes figuran muestran los resultados del ensayo de la configuración final elegida.

Las Figuras 6.27 y 6.28 muestra de forma clara que el recorrido del dron se corresponde con el circuito propuesto, verificando así que la programación de vuelo es correcta. Además, durante el despegue se realizan movimientos en el plano XY, estos son debidos a que el despegue desestabiliza la posición del dron. Por ello se ha programado de forma que el recorrido comienza 5 segundos después de despegar para que el sistema tenga tiempo de regresar a la posición inicial antes de comenzar el recorrido.

El radio de giro de 0.3 metros se aplica también para el último *waypoint*, por ello el recorrido acaba a 0.3 metros antes del punto donde debería acabar.

Las Figuras 6.29 y 6.30 muestran la evolución en las coordenadas horizontales. La inestabilidad durante el despegue se presenta principalmente en el eje X. Los resultados muestran como los movimientos en el plano horizontal son suaves y están en línea con el recorrido

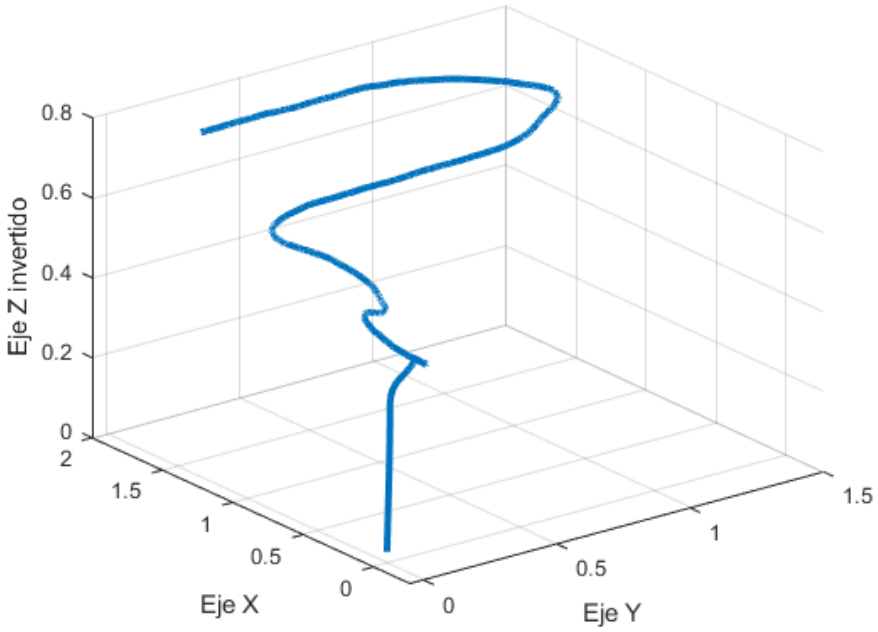


Figura 6.27: Gráfico 3D del recorrido del circuito con la configuración final

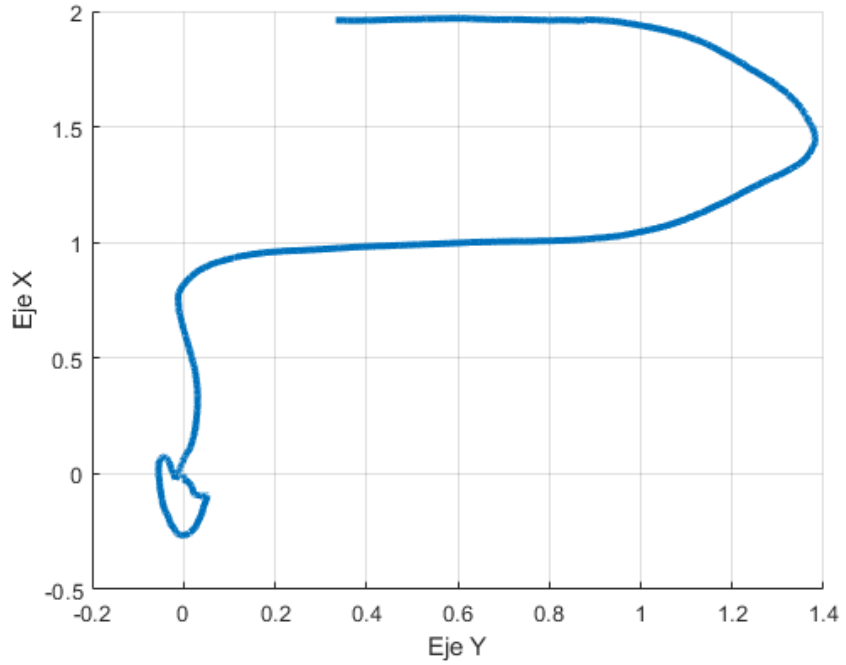


Figura 6.28: Gráfico del movimiento horizontal durante el recorrido del circuito con la configuración final

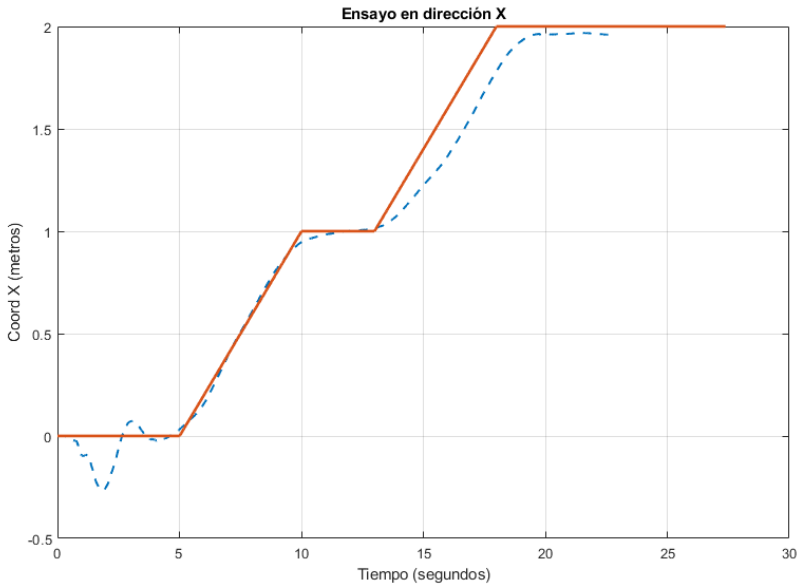


Figura 6.29: Movimiento en el eje X a lo largo del tiempo del ensayo. En naranja representación del recorrido y el resultado del ensayo en azul

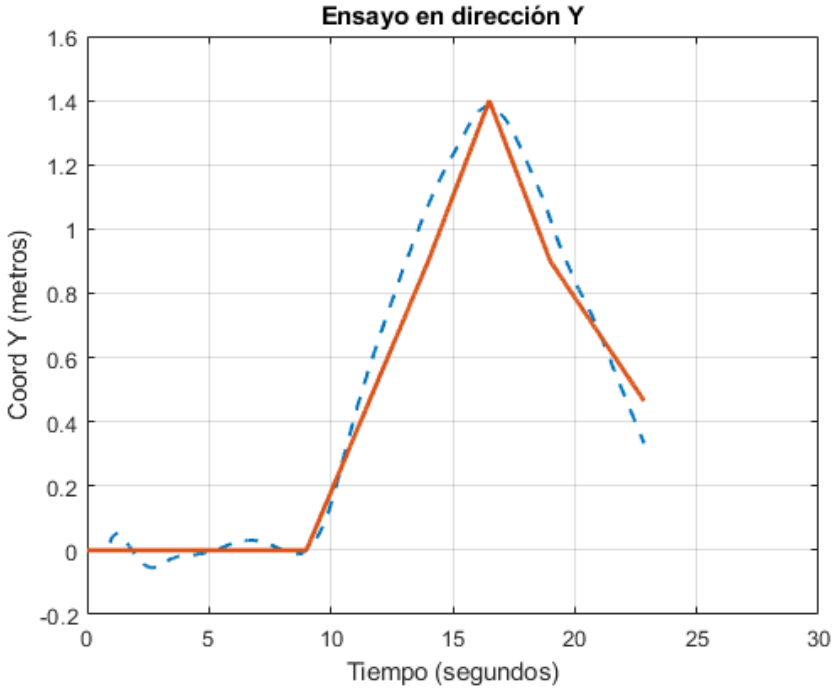


Figura 6.30: Movimiento en el eje Y a lo largo del tiempo del ensayo. En naranja representación del recorrido y el resultado del ensayo en azul

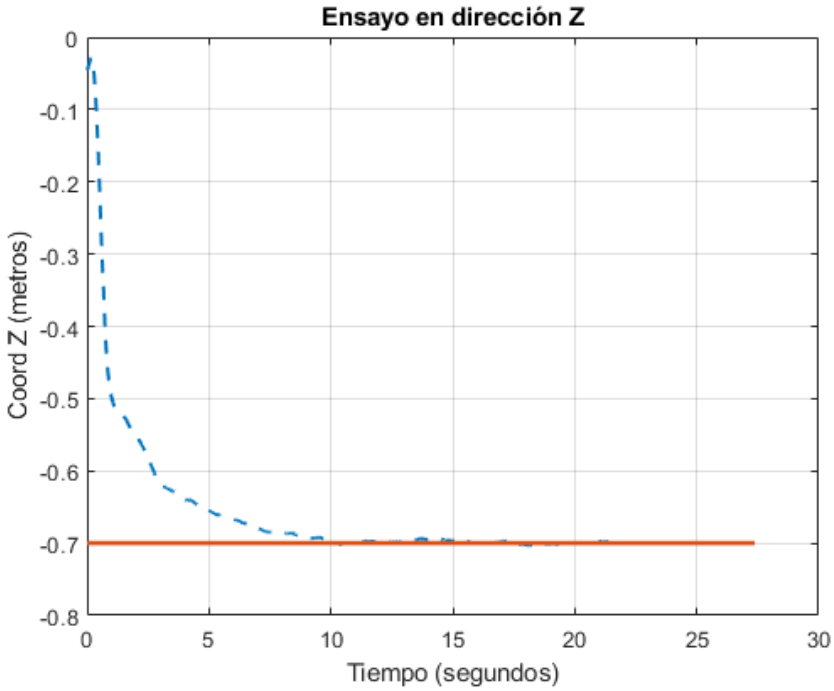


Figura 6.31: Movimiento en el eje Z a lo largo del tiempo del ensayo. En naranja representación del recorrido y en azul el resultado del ensayo

La Figura 6.31 muestra la evolución en la coordenada Z, se observa como tarda en alcanzar la altura 0.7 metros, pero luego se mantiene estable. Esto no supone un problema ya que el movimiento vertical es el menos importante en la aplicación objetivo y mientras se alcance la altura de diseño y se mantenga se considera un control satisfactorio.

La Figura 6.32 muestra la evolución de la guiñada, el ensayo demuestra que se realizan demasiados cambios de guiñada en un lapso pequeño. Por ello, se observa que el sistema no llega a estabilizarse antes de que lleguen el siguiente cambio. A pesar de este fallo en el diseño del circuito se puede observar como el sistema busca las guiñadas de diseño.

Se puede concluir que el ajuste de los controladores final es satisfactorio. El vuelo del Parrot Mambo es estable y capaz de recorrer el circuito programado de forma suave y precisa. Esta configuración será la utilizada durante el desarrollo de la aplicación, así como de sus ensayos en el *hardware*.

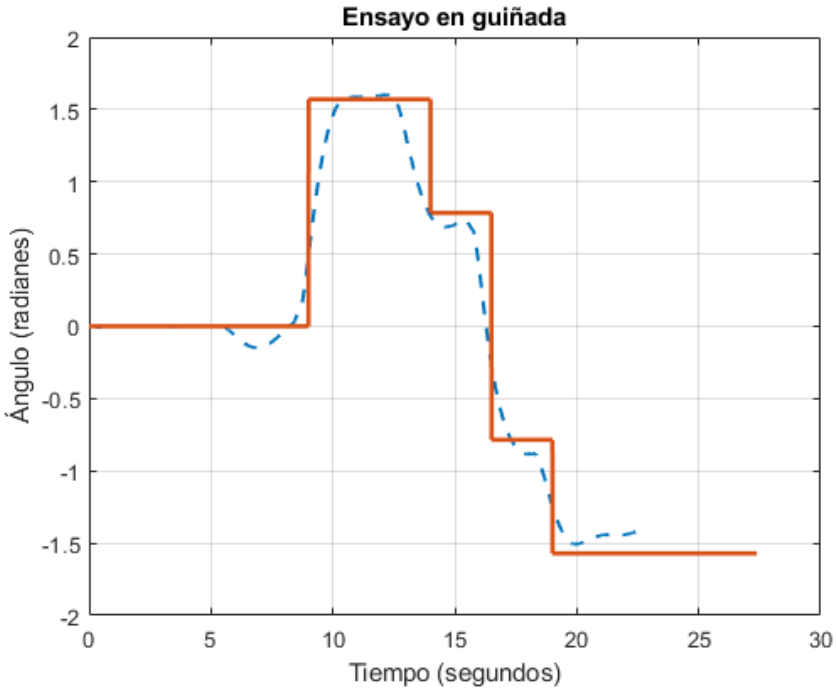


Figura 6.32: Gráfico de la evolución de la guiñada durante el recorrido del circuito con la configuración final. En naranja representación del recorrido y en azul el resultado del ensayo

Capítulo 7

Aplicación al seguimiento de trayectorias mediante realimentación visual

Como ya se comentó en la introducción, uno de los subojetivos del proyecto es desarrollar una aplicación donde el Parrot Mambo se guíe mediante una trayectoria marcada en el suelo.

Para desarrollar la aplicación se hará uso del proyecto *parrotMinidroneCompetition* explicado en el Capítulo 5. Se trabajará principalmente en dos bloques: el primero es el bloque *Image Processing System* donde se procesará y analizará la imagen y el segundo es el bloque *Path Planning* que calculará los movimientos del dron para el seguimiento de la trayectoria.

La aplicación se probará primero en el simulador para comprobar su funcionamiento, cuando este sea satisfactorio, se procederá a implementar la aplicación en el *hardware*.

7.1 Procesamiento de imagen

El objetivo de este bloque es tomar la imagen de la cámara, procesarla para aislar la trayectoria marcada en el suelo y finalmente analizarla para que permita orientar al Parrot Mambo.

En la Figura 7.2 se observan 4 subsistemas principales. El primero a la izquierda en color verde recibe la imagen desde la cámara y mediante el bloque *PARROT Image Conversion* la transforma al modelo de color RGB. Sus 3 salidas son los tres colores del modelo: rojo (R), verde (G) y azul (B) van directo al siguiente subsistema.

El subsistema de color azul claro se encarga del procesamiento de la imagen. La idea

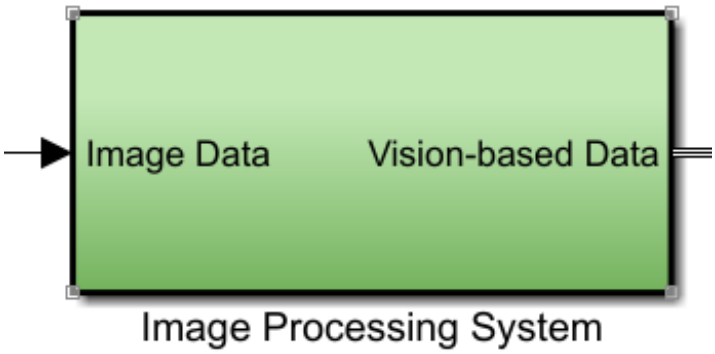


Figura 7.1: Bloque para el procesamiento de imagen

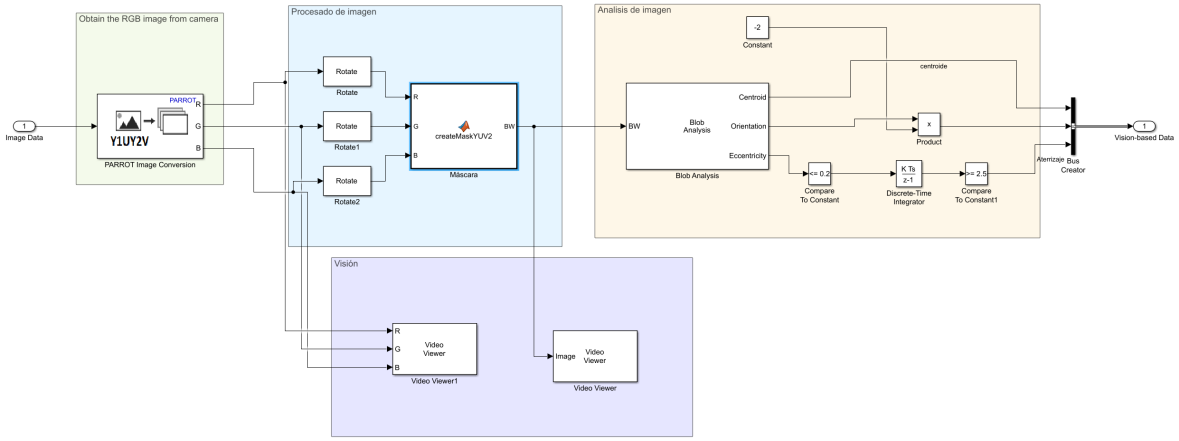


Figura 7.2: Dentro del bloque *Image Processing System*

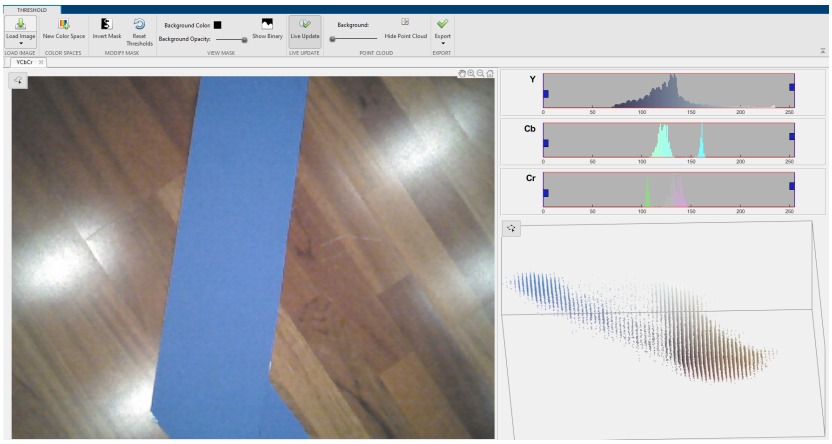


Figura 7.3: Aplicación *Color Thresholder*. Imagen original

es tomar la imagen en esquema RGB y transformarla al código binario (blanco/negro), la trayectoria detectada se mostrará en blanco y el resto en negro.

Se puede observar que la imagen primero se rota y luego entra en el bloque *Máscara*, la rotación se explicará en el siguiente subsistema. El bloque *Máscara* se encarga de la transformación de la imagen y alberga un código en lenguaje Matlab que se encarga de ello.

Este código se ha obtenido mediante el uso de la aplicación *Color Thresholder* proporcionada por la *Toolbox Image Processing*. *Color Thresholder* permite segmentar imágenes en función de su color. Permite el uso de 4 esquemas de colores: RGB, HSV, YUV y $L^*a^*b^*$ (20).

En este caso se ha utilizado el modelo YUV ya que es el que ha dado los mejores resultados durante las pruebas realizadas. En el modelo YUV la Y representa luminancia (color blanco/negro) y U/V la crominancia (información respecto al color). La luminancia se ajusta para evitar o disminuir el efecto de los reflejos y la crominancia para la diferenciación del color azul.

Primero se toma una imagen de la trayectoria marcada en el suelo con la cámara del dron a la altura que se va a ensayar. Esta imagen se carga en la aplicación *Color Thresholder* como se puede ver en la Figura 7.3.

A la izquierda se aprecia la imagen cargada y a su derecha los parámetros de ajuste de YUV (en la aplicación se denomina YCrCb pero es el mismo modelo de color).

En la Figura 7.4 se aprecia como se han eliminado los reflejos del suelo mediante el ajuste del parámetro Y.

Por último se ajustan los parámetros U y V para segmentar el color azul (Figura 7.5). El botón del menú superior *Export* genera un código Matlab que procesa las imágenes con la configuración elegida en la aplicación.

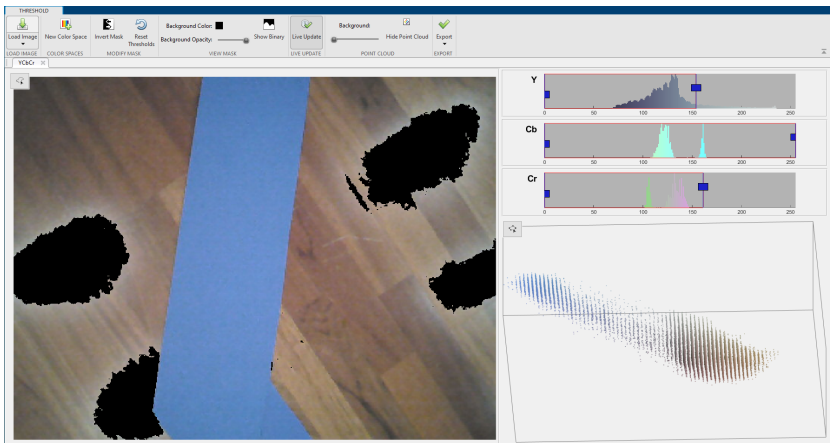


Figura 7.4: Aplicación *Color Thresholder*. Imagen sin reflejos

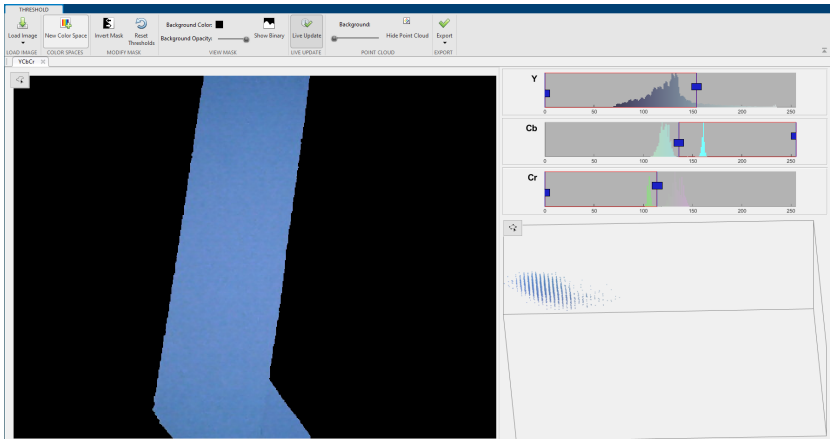


Figura 7.5: Aplicación *Color Thresholder*. Imagen con el color azul segmentado

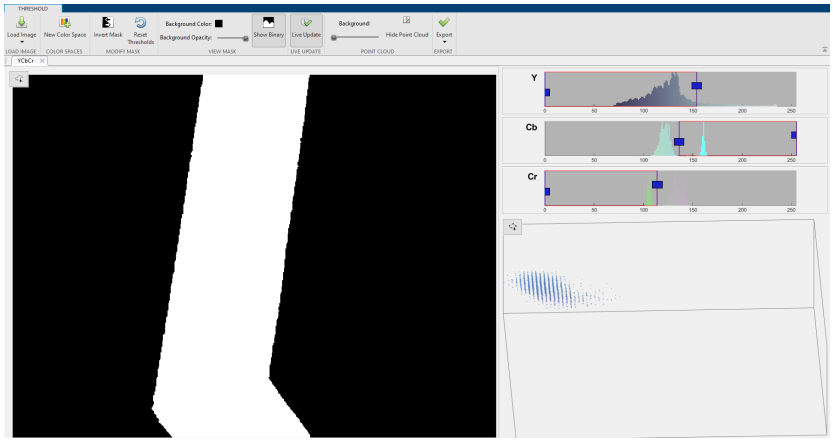


Figura 7.6: Aplicación *Color Thresholder*. Imagen binaria

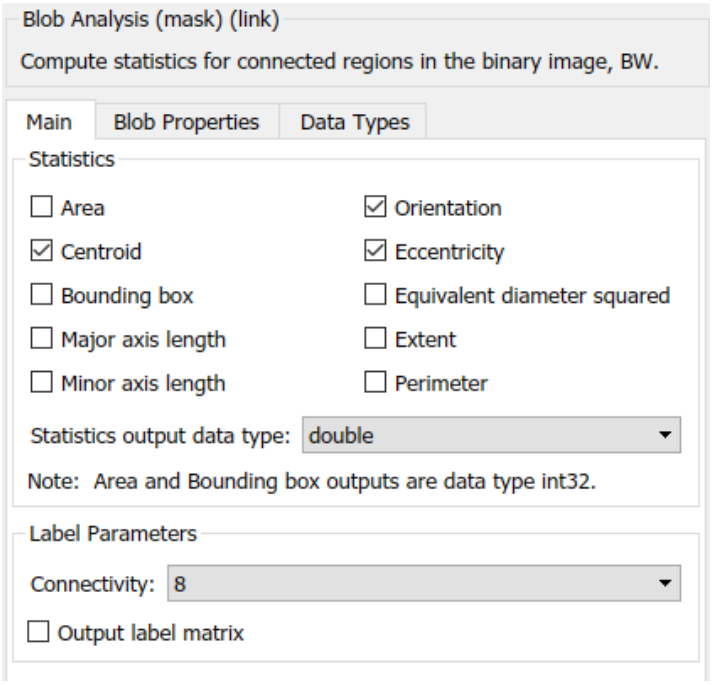


Figura 7.7: Menú configuración bloque *Blob Analysis*. Parámetros de análisis

El resultado final se observa en la Figura 7.6, es una imagen binaria donde se ha separado completamente la trayectoria marcada del fondo facilitando mucho su análisis.

El tercer subsistema es de color amarillo de la Figura 7.2 se encarga de analizar la imagen para orientar Parrot Mambo, en este caso se extrae la orientación, el centroide y la excentricidad de la trayectoria.

El análisis de la imagen se realiza mediante la técnica de visón por computador *Blob Analysis*. Esta técnica detecta regiones continuas en una imagen binaria y determina parámetros de estos.

Los parámetros a analizar son muy variados: el centroide, el área, excentricidad, perímetro etc. como se observa en la Figura 7.7. Esta técnica se suele utilizar cuando existe varias regiones que detectar, pero en este caso se especifica al análisis que solo debe detectar una región. En la Figura 7.8 se aprecia como se ha especificado como 1 el número máximo de regiones, así como el límite superior e inferior del número de píxeles que lo conforman.

La rotación que se había realizado en el subsistema anterior junto con la multiplicación por -2 sirven para transformar la orientación obtenida del bloque *Blob Analysis* a la misma referencia que el sistema fijo en el dron.

Los parámetros que se especifican como salida en este caso son el centroide, la orientación y la excentricidad. La orientación sirve para determinar la guiñada que debe tener el Parrot Mambo.

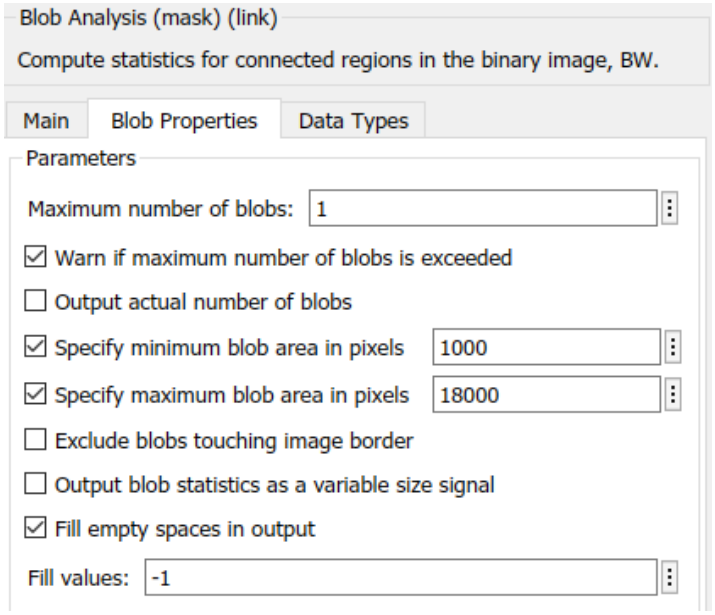


Figura 7.8: Menú configuración bloque *Blob Analysis*. Propiedades de la región

La rotación que se había realizado en el subsistema anterior junto con la multiplicación por -2 sirven para transformar la orientación obtenida del bloque *Blob Analysis* a la misma referencia que el sistema fijo en el dron.

Si se compara el centroide de la región detectada con el de la imagen se puede determinar como de desplazado está el dron del centro de la trayectoria. En el siguiente sección se mostrará como se utiliza.

La excentricidad se puede definir como el grado de similitud con una circunferencia. Las líneas rectas de la trayectoria tienen una excentricidad cercana a 1 y el círculo de aterrizaje es casi 0. Esto se usa para determinar cuándo se está sobrevolando el punto de aterrizaje.

El último subsistema es el que se encuentra en la parte inferior en color morado, en el se encuentran dos bloques que reciben las imágenes y las muestran en tiempo real en sendas ventanas. Una muestra la imagen sin procesar y la otra procesada en el modelo binario.

7.2 Algoritmo seguimiento de trayectoria

Una vez se ha detectado la trayectoria y se conocen sus parámetros, solo queda que el Parrot Mambo calcule sus movimientos para el seguimiento de esta. Esta lógica se implementa en el bloque *Path Planning* dentro del sistema de control de vuelo.

Se toma como entrada la información procedente del procesado de imagen y el canal

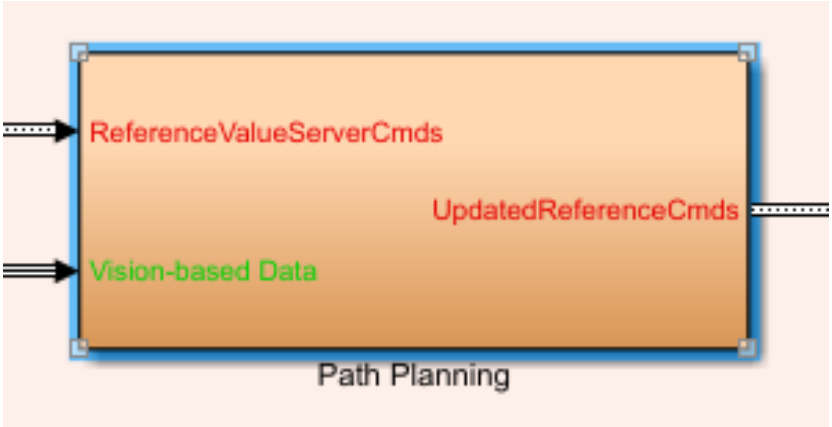


Figura 7.9: Bloque *Path Planning*

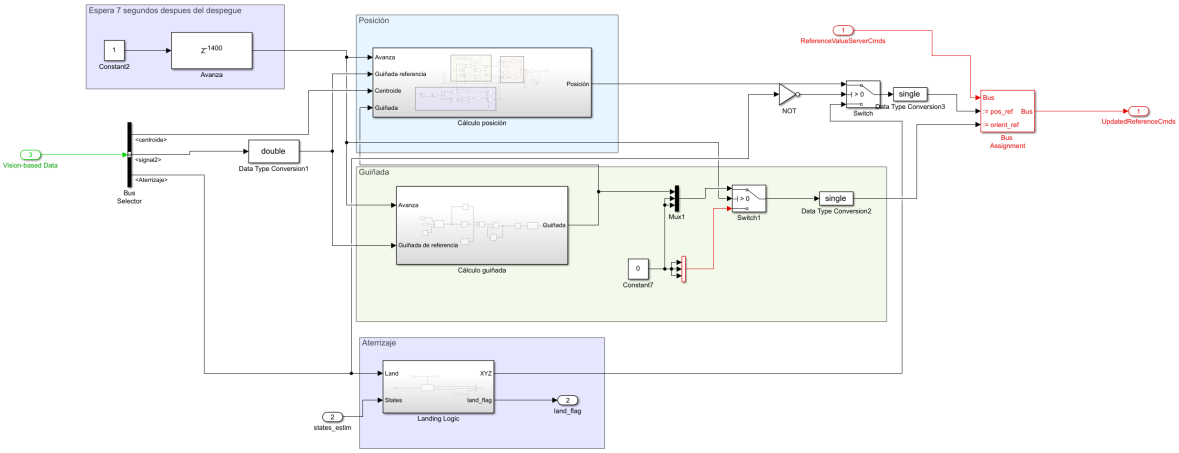


Figura 7.10: Dentro del bloque *Path Planning*. Algoritmo de guiado

de información para los comandos de vuelo.

En la Figura 7.10 se observan 4 subsistemas principales, el primero en la parte superior izquierda es el más sencillo y sirve para que el Parrot Mambo comience a a seguir los algoritmos 7 segundos después del despegue. Estos 7 segundos sirven para estabilizarse en la posición inicial tras el despegue, esto es necesario ya que en el capítulo anterior se observó cómo el sistema se desplazaba en el plano XY durante el comienzo del vuelo. Asimismo, se le otorga tiempo para alcanzar la altura de diseño.

Pasados los 7 segundos la señal *Avanza* se activa y alimenta al resto de subsistemas con un 1 lógico marcando así el comienzo del seguimiento de la trayectoria.

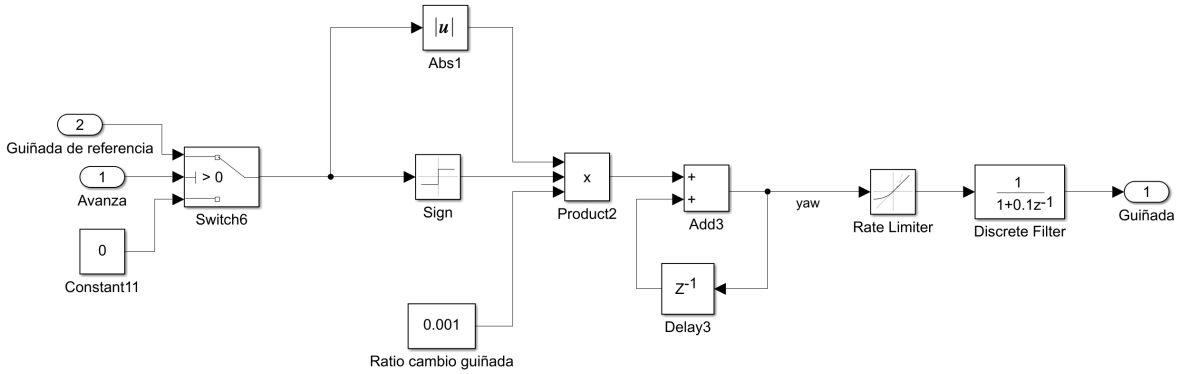


Figura 7.11: Dentro del bloque *Cálculo guiñada*

7.2.1 Cálculo de guiñada

El subsistema de color verde de la Figura 7.10 se encarga del cálculo de la guiñada. Está compuesto por un bloque llamado *Calculo guiñada* y un bloque *switch*. En el primero, como su nombre indica, se estima la guiñada que debe seguir el Parrot Mambo según la información procedente del análisis de imagen.

El bloque *switch* posee 3 entradas y 1 salida, su funcionamiento consiste en elegir cuál de las entradas se propagada por su salida, para ello la segunda entrada sirve de condición. En este caso la condición es la señal *Avanza* antes explicada, mientras no hayan pasado los 7 segundos se le indica al sistema que debe mantener sus ángulos de orientación en 0. Una vez pasados los 7 segundos, el bloque *switch* cambia su salida, indicándole al controlador la guiñada calculada.

Para el cálculo de la guiñada (Figura 7.11) primero se utiliza un bloque *switch* de la misma forma recién explicada. Mientras no hayan pasado los 7 segundos el sistema recibe un 0. Después, el bloque *switch* propagará la orientación procedente del procesamiento de imagen que en este caso se ha llamado *Guiñada de referencia*.

Acto seguido se calcula la guiñada que debe seguir el Parrot Mambo, para ello se toma el signo y el valor absoluto de la guiñada de referencia junto con un parámetro llamado *Ratio cambio guiñada*. Todos esto valores se multiplican y como resultado se tiene un valor de incremento de guiñada, que será positivo si el giro es hacia la derecha o negativo hacia la izquierda. Además, su valor absoluto será mayor si el sistema debe girar mucho o pequeño si debe hacerlo poco. El *Ratio cambio guiñada* se ha estimado mediante ensayos en el simulador, si su valor aumenta la velocidad de giro lo hace también y sucede lo propio al disminuir. El valor utilizado se ha elegido ya que representa un compromiso entre velocidad y control del sistema.

Mediante el uso de un bloque suma y un bloque *delay* los valores de incremento de guiñada se van acumulando de forma que se obtiene el valor de la guiñada final. Este sistema permite ir actualizando la guiñada en tiempo real conforme le va llegando

la información del procesado de la imagen. De esta forma si a través del análisis de imagen se detecta que la orientación del Parrot Mambo difiere de la orientación de la trayectoria, el sistema lo corregirá de manera automática.

Después del cálculo de la guiñada existen dos bloques que se encargan de asegurar que no se produzcan cambios muy bruscos. El primero, *Rate limiter*, impone un máximo a la variación de la señal entre pasos consecutivos y el segundo, *Discrete Filter* realiza un filtrado tipo IIR (*Infinite Impulse response*) que supone una carga computacional menor frente a otros filtros.

Finalmente, la guiñada calculada sale del subsistema hacia el bloque de cálculo de posición y hacia el controlador de vuelo.

7.2.2 Cálculo de posición

El bloque *Cálculo posición* es el de color azul claro en la Figura *dpath2*. Determina la posición que debe tener el Parrot Mambo conforme va realizando el recorrido. El movimiento se realiza por guiado de guiñada y se puede expresar con las siguientes ecuaciones

$$X^t = \Delta X + X^{t-1} \tag{7.1}$$

$$Y^t = \Delta Y + Y^{t-1} \tag{7.2}$$

$$\Delta X = C * \cos\psi \tag{7.3}$$

$$\Delta Y = C * \sin\psi \tag{7.4}$$

donde t marca un instante, ψ la guiñada y C es una constante de velocidad.

Al igual que el subsistema anterior la posición se calcula mediante incrementos que se van acumulando para dar el valor final, y el sistema solo comienza cuando la variable *Avanza* se activa a los 7 segundos. Esta suma se realiza en el subsistema de color beige de la derecha de la Figura 7.12.

También se incluyen filtros para evitar cambios bruscos como en el caso anterior y la altura se le indica de forma constante.

El guiado por guiñada consiste en que los desplazamientos se realizan exclusivamente mediante cabeceo. La guiñada marca la dirección de avance y mediante el cabeceo se realiza el desplazamiento. Por ejemplo si la guiñada es, 0 grados, el incremento en X será máximo y en Y nulo siguiendo así el sistema de referencia fijo en el dron.

Este guiado se encuentra en el subsistema de color verde. La constante de velocidad C antes mencionada, varía entre un valor rápido y uno lento. Para la elección de uno u otro se utiliza el valor de guiñada de referencia proveniente del procesado de imagen. Si

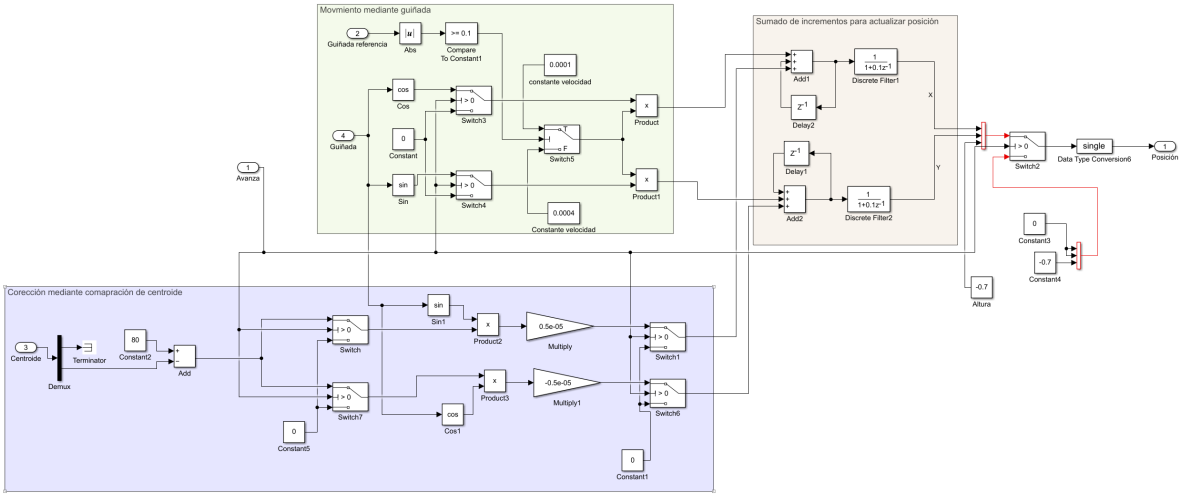


Figura 7.12: Dentro del bloque *Cálculo posición*

la guiñada de referencia tiene un valor alto significa que el sistema debe girar bastante y por tanto se le indica al sistema que utilice la constante de velocidad lenta, si por el contrario el dron está alineado con la trayectoria o solamente tiene que girar un poco se utiliza la constante rápida. Esta decisión de diseño es resultado de lo observado en pruebas de campo, donde si se usaba una constante de velocidad rápida de forma constante, el Parrot Mambo avanzaba demasiado antes de que le diese tiempo a girar. Este avance superior al necesario resultaba en la pérdida de visión de la trayectoria con la pérdida de control que esto supone.

De nuevo estas constantes se han determinado mediante experimentación y representan un acuerdo entre velocidad de desplazamiento y control.

Si el sistema de guiado marcado en color verde en la Figura 7.12 se basaba solamente en la guiñada y cabeceo, el sistema azul es complementario y sirve de corrección mediante alabeo. Para determinar esta corrección, el sistema compara la coordenada Y del centroide de la trayectoria con el valor constante de 80. Este valor representa el centro en el eje Y de la cámara. De esta forma, si la diferencia es grande significa que se debe realizar una corrección mayor que si es pequeño o nulo. Después, este valor se multiplica por el seno y el coseno de la guiñada, esto se hace porque el desplazamiento por alabeo depende de la orientación del dron en el sistema inercial. Así, por ejemplo, si la guiñada actual es de 90 grados en el sistema inercial, el alabeo será exclusivamente un movimiento en el eje X y por ello esta coordenada de desplazamiento usa el seno y la coordenada Y es el coseno.

También se hace uso de una constante que multiplica el valor de la corrección para adecuarse al resto del sistema. Por convención de signos del sistema, se usa el positivo para la coordenada X y el negativo para la Y.

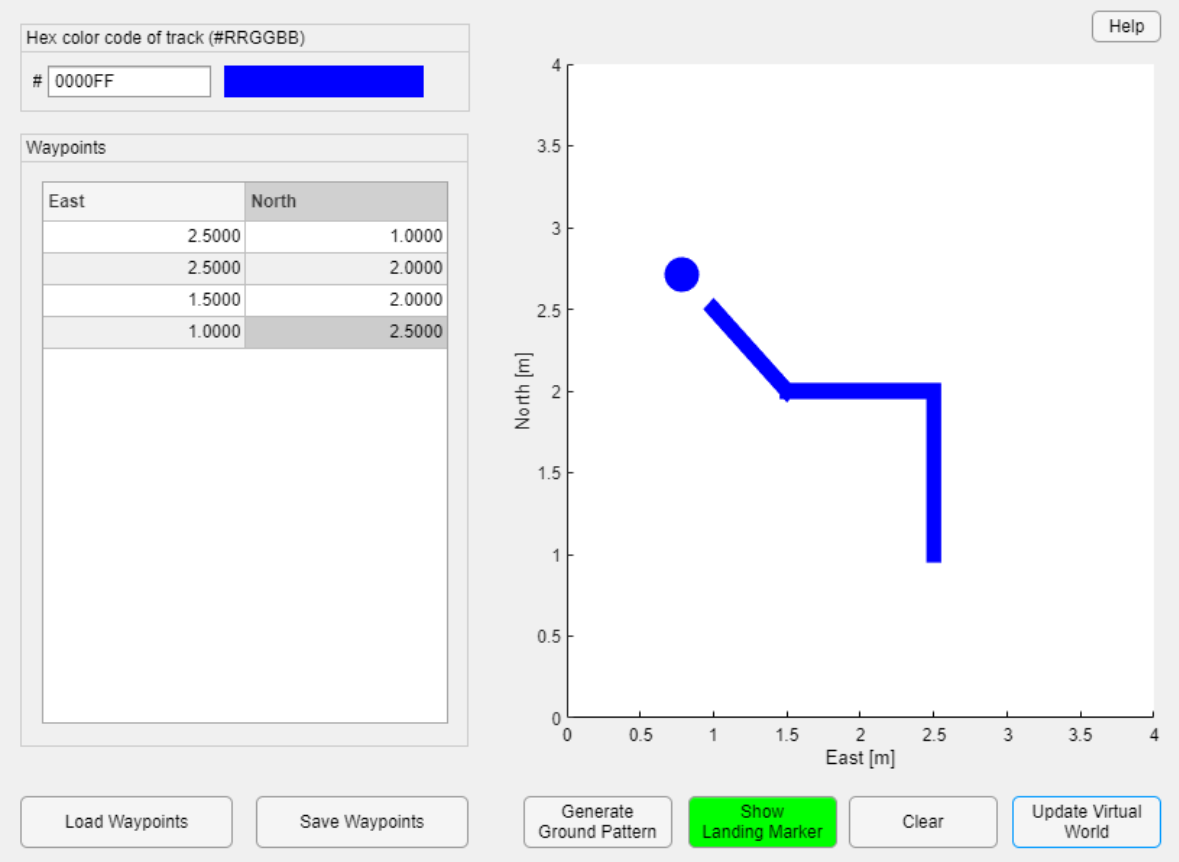


Figura 7.13: Aplicación *Track Builder*

7.3 Resultados del ensayo en simulación

Para verificar el funcionamiento de los algoritmos implementados se hace uso del simulador visto en el Capítulo anterior. Primero es necesario recrear una trayectoria en el entorno de simulación, para ello se hace uso de una de las funcionalidad del proyecto *asbQuadCopter*: *Track Builder*.

Se trata de una aplicación que mediante una interfaz sencilla permite diseñar trayectorias, se puede ver en la Figura 7.13.

Se puede elegir el color mediante código RGB hexadecimal, también se pueden elegir los *waypoints* del recorrido que se muestran de forma visual en la parte derecha. También permite guardar y cargar las trayectorias diseñadas mediante *Load Waypoints* y *Save Waypoints* así como la posibilidad de crear una marca de aterrizaje al final del recorrido. Finalmente para implementar el recorrido en el entorno del simulador se hace uso del botón *Update Virtual World*.

Para realizar la simulación se sigue el mismo procedimiento que el mostrado anteriormente. Su seguimiento se realiza mediante 3 ventanas: una nos muestra lo que ve la cámara del dron (Figura 7.14 izquierda), otra lo mismo pero tras procesar la imagen

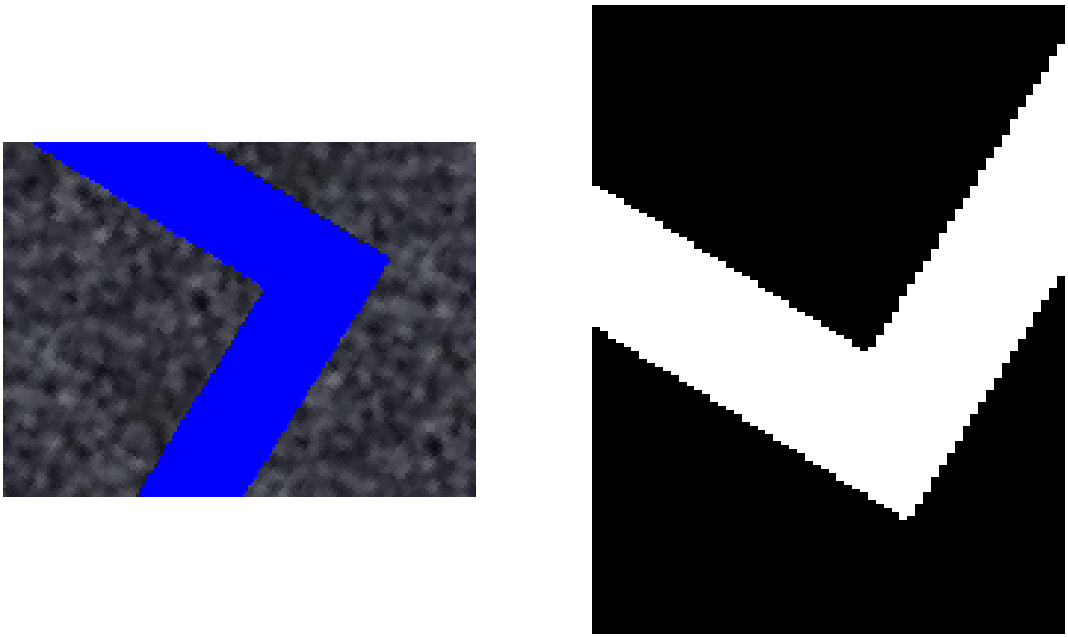


Figura 7.14: Cámara del dron durante de la simulación. A la izquierda la imagen original y a la derecha la procesada

(Figura 7.14 derecha) y la última nos muestra al dron navegando por el entorno de la simulación (Figura 7.15).

Para la presentación de resultados se va a seguir el mismo esquema que en el Capítulo anterior.

Las Figuras 7.16 y 7.17 muestran cómo se realiza el seguimiento de la trayectoria diseñada de forma casi perfecta. De esta forma se verifica que los sistemas de control mediante la realimentación visual diseñados son correctos.

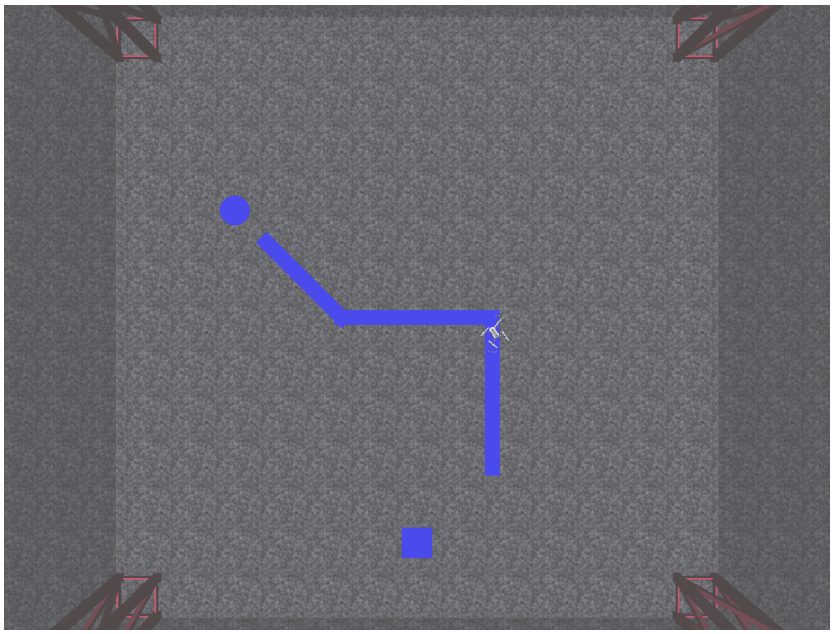


Figura 7.15: Simulación durante el ensayo

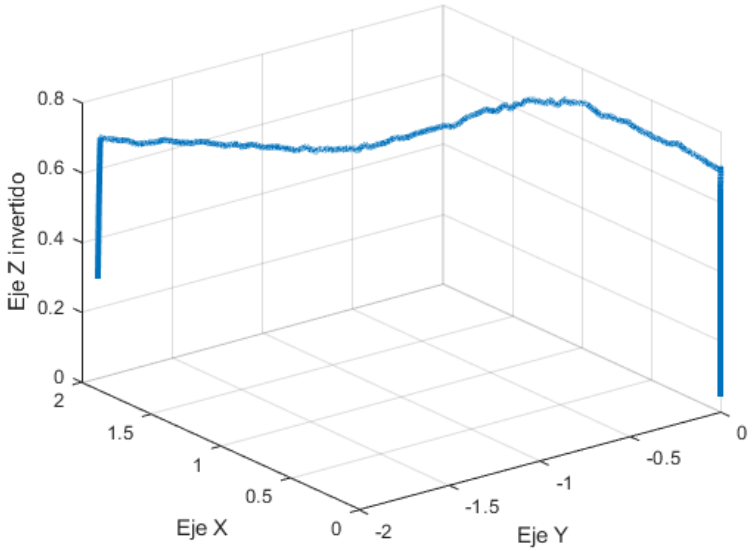


Figura 7.16: Gráfico 3D del recorrido durante la simulación del seguimiento de trayectoria

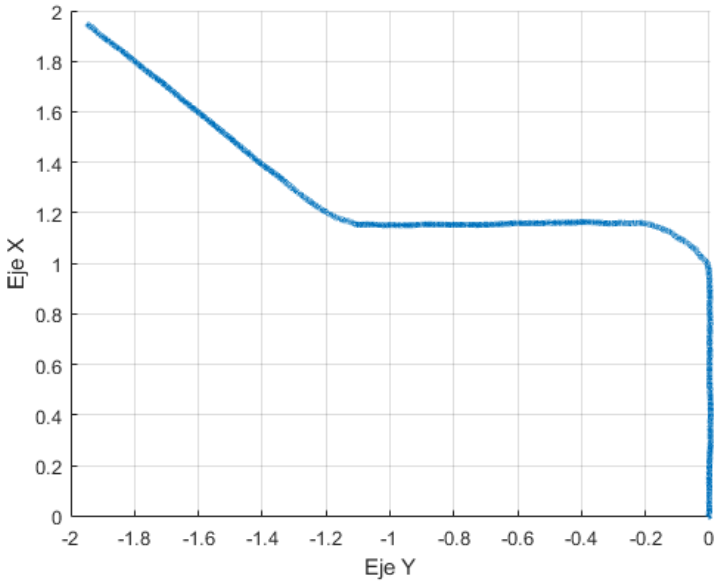


Figura 7.17: Gráfico del movimiento en el plano horizontal XY durante la simulación del seguimiento de trayectoria

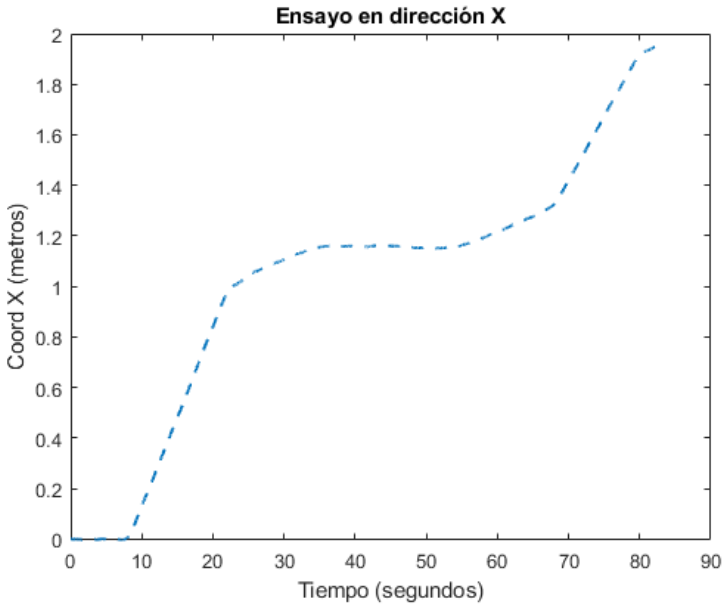


Figura 7.18: Movimiento en el eje X a lo largo del tiempo del ensayo de seguimiento de trayectoria en simulador

En la Figura 7.18 se observa de forma muy clara las dos constantes de velocidad utilizadas, en el primero tramo el incremento en X tiene una pendiente más pronunciada que durante el tramo siguiente donde se está realizando el giro.

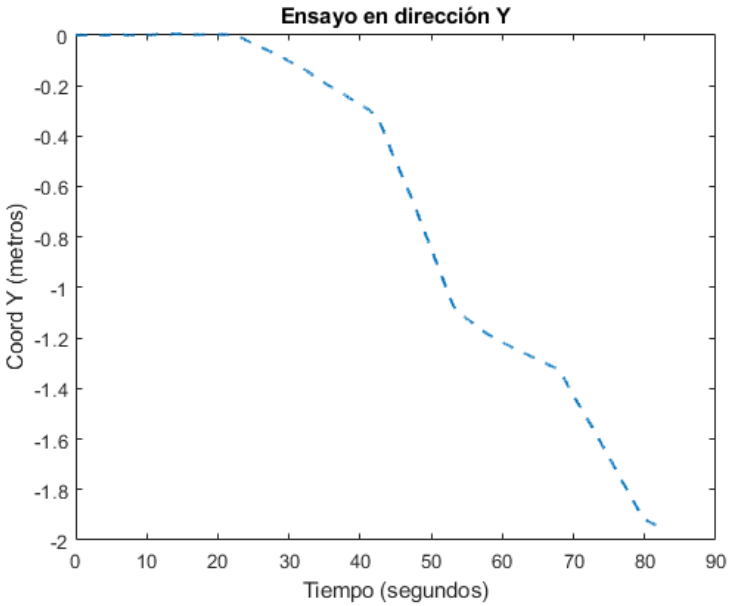


Figura 7.19: Movimiento en el eje Y a lo largo del tiempo del ensayo de seguimiento de trayectoria en simulador

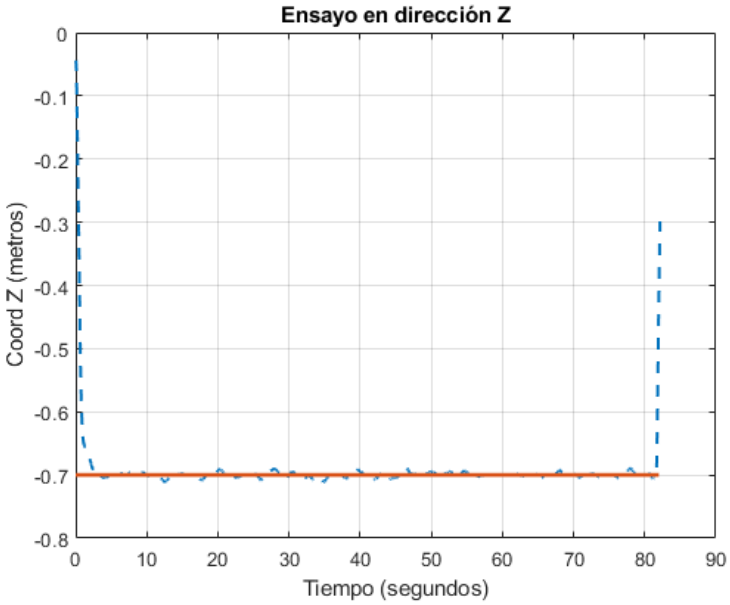


Figura 7.20: Movimiento en el eje Z a lo largo del tiempo del ensayo de seguimiento de trayectoria en simulador

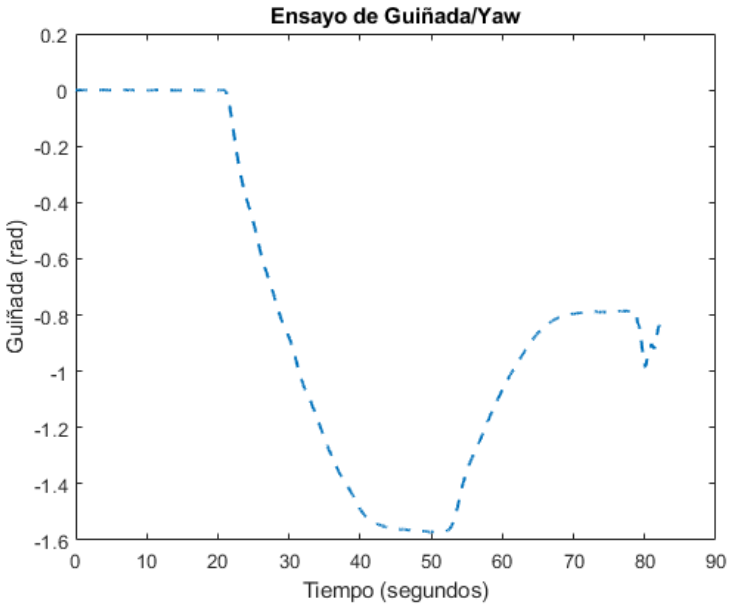


Figura 7.21: Evolución de la guiñada durante el ensayo de seguimiento de trayectorias en el simulador

La Figura 7.21 muestra de forma clara como la guiñada va variando a lo largo del recorrido. En la posición inicial su valor es 0, luego pasa hasta el valor cercano a -1.6 que, en realidad, es $-pi/2$ característico del giro de 90^{o} de la trayectoria definida. Finalmente vuelve a subir hasta entorno el valor -0.8 que es $-pi/4$ del giro final de 45^{o}.

7.4 Ensayo en *Hardware*

Para los ensayos en el *hardware* es necesario disponer de un entorno de trabajo adecuado, para ello se ha utilizado la habitación más diáfana posible. Esto es debido que cualquier obstáculo cercano el dron mientras vuela puede alterar las lecturas de sus sensores. Por ejemplo, si durante el vuelo el Parrot Mambo se acerca a una mesa, esta afectará a sus lecturas del sonar y como resultado el dron variará su altura de forma equívoca.

Es imprescindible realizar los ensayos en un entorno indoor, ya que debido al bajo peso del dron los entornos expuestos al viento u otras perturbaciones desestabilizan con gran facilidad la planta motora, haciendo los vuelos prácticamente incontrolables. Por lo cual, el diseño de la trayectoria está limitada en sus dimensiones a la habitación más grande disponible.

Para el modelado de la trayectoria en el entorno de ensayo se ha utilizado cartulina de color azul recortada. Se ha recortado de forma que los tramos sean rectos y uniformes entre ellos, así se mantienen las características de la trayectoria constantes durante todo el tramo. La cartulina se mantiene adherida al suelo mediante cinta aislante que

Figura 7.22: Montaje experimental

se coloca en la parte inferior. Se debe asegurar la adhesión completa en todo el recorte ya que si se dejan extremos sueltos las perturbaciones aerodinámicas producidas por el dron durante su vuelo pueden mover o levantar la cartulina.

La Figura 7.22 muestra una imagen del montaje experimental utilizado.

Como se puede ver la trayectoria diseñada es muy similar a la mostrada en los ensayos en simulador, pero no es exactamente la misma debido a limitaciones de espacio. La principal diferencia es que el segundo giro no es de 45º sino de 35º.

También se observan una serie de recortes menores en el entorno de la trayectoria. El porqué de ello está en la técnica de medición *optical flow* explicada anteriormente, los recortes ayudan a estimar mejor los desplazamientos que realiza el Parrot Mambo durante el recorrido.

Una vez se dispone del entorno experimental lo primero es asegurar que la toma, procesamiento y análisis de las imágenes en tiempo real es satisfactoria. Por ello en la Figura 7.23 se muestra un fotograma extraído de uno de los vuelos de ensayo, se muestra tanto la imagen real como la procesada.

Se puede observar cómo se ha realizado la segmentación de la imagen de forma casi perfecta. Tras el procesado, se ha eliminado de la imagen original el fondo de forma completa incluyendo los reflejos en el suelo, de esta forma se aísla la trayectoria.

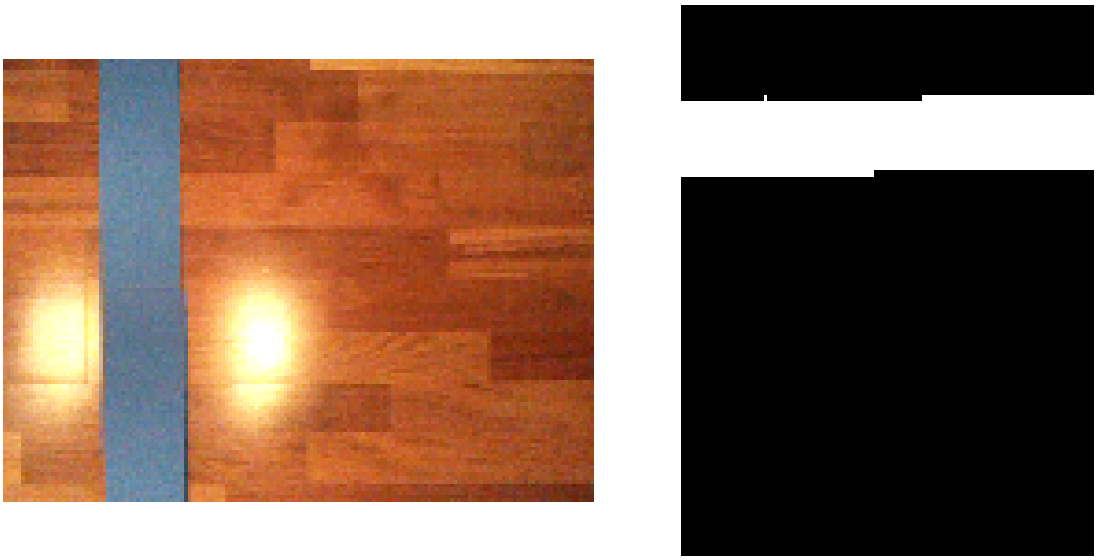


Figura 7.23: A la izquierda el fotograma directamente del cámara del dron, a la derecha el resultado de su procesamiento en tiempo real

7.5 Resultados del ensayo en *Hardware*

Para mostrar los resultados de los ensayos en *hardware* se seguirá el mismo esquema que el usado en los ensayos en simulación. Pero primero se va a mostrar que los algoritmos desarrollados son capaces de llevar a cabo la aplicación propuesta.

Para ello se ha grabado un vídeo que demuestra el Parrot Mambo es capaz de seguir la trayectoria diseñada en el entorno de trabajo, así como capaz de aterrizar en el circulo final de aterrizaje. Debido a la naturaleza del documento esto se muestra que mediante una serie de fotogramas procedentes del vídeo.

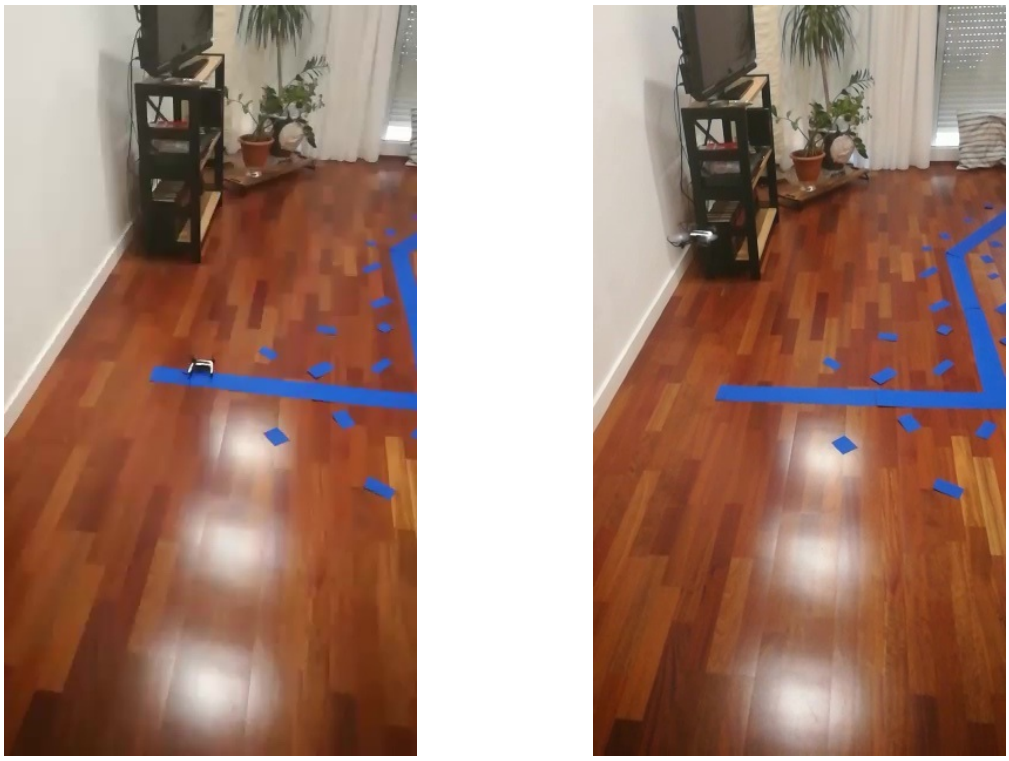


Figura 7.24: Primeros dos fotogramas del vídeo. El de la izquierda muestra la posición antes de despegar y a la derecha recién despegado

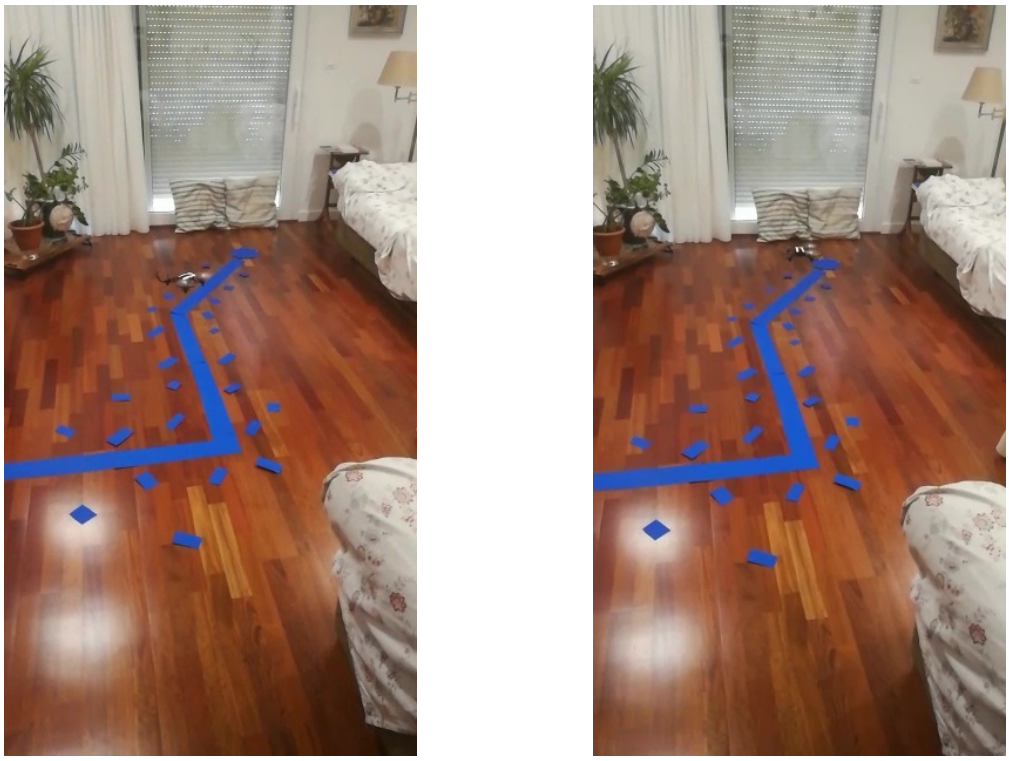


Figura 7.25: Tercer y cuarto fotograma del vídeo. El de la izquierda muestra como gira el Parrot Mambo al llegar al giro y a la derecha cuando se alinea con el segundo tramo

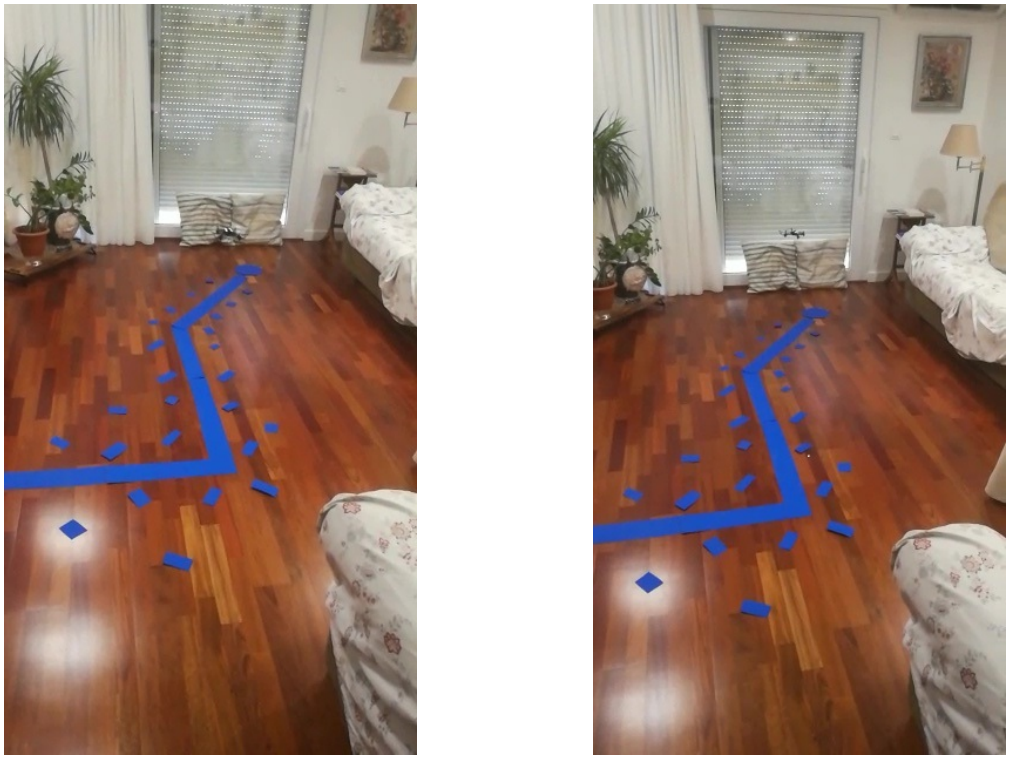


Figura 7.26: Quinto y sexto fotograma del vídeo. El de la izquierda muestra como ha avanzado por el segundo tramo y a la derecha muestra como se se alinea con el tercer tramo

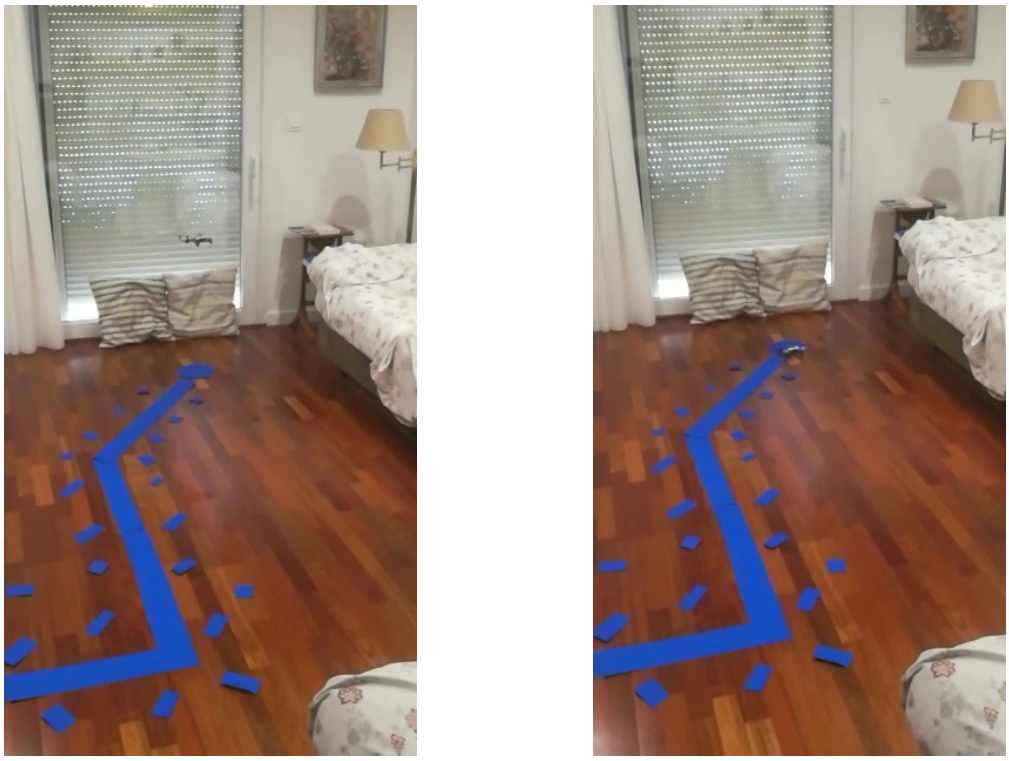


Figura 7.27: Séptimo y octavo fotograma del vídeo. El de la izquierda muestra se mantiene sobrevolando el círculo de aterrizaje y a la derecha el aterrizaje

Por tanto se demuestra que se ha llevado a cabo la aplicación propuesta y que se han cumplido los objetivos propuestos. Ahora se mostrarán las gráficas obtenidas durante el vuelo mostrado. En ellas se encuentran superpuesta dos series de datos: en azul se muestra el estado estimado a lo largo del vuelo y en naranja las órdenes de vuelo, es decir, en azul se muestra la trayectoria que ha llevado el dron siguiendo las órdenes marcadas en naranja.

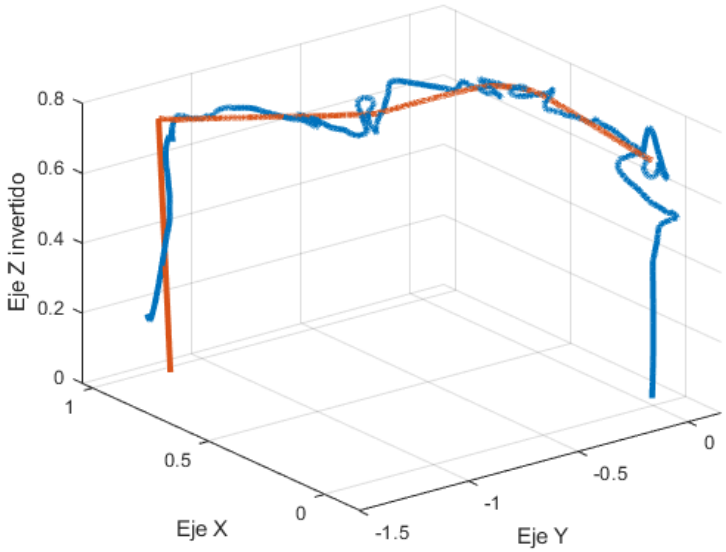


Figura 7.28: Gráfico 3D del recorrido durante el ensayo de seguimiento de trayectoria en el *hardware*

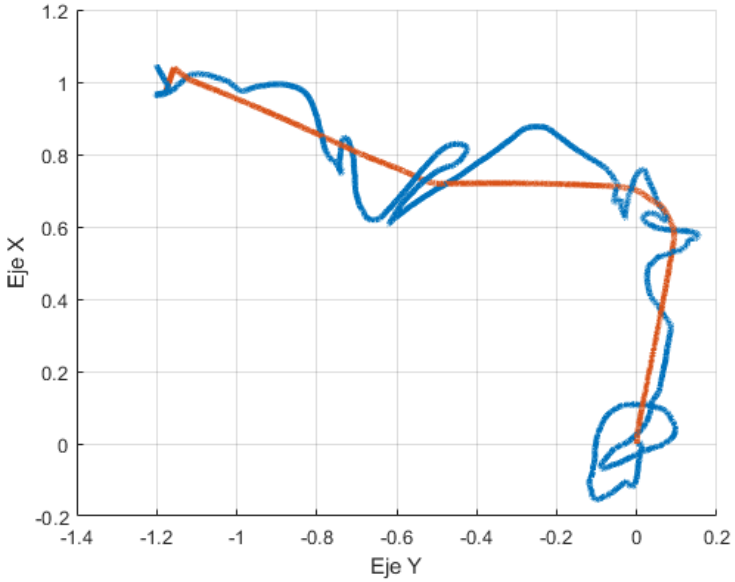


Figura 7.29: Gráfico del movimiento en el plano horizontal XY durante el ensayo de seguimiento de trayectoria en el *hardware*

Se observa claramente como el Parrot Mambo sigue la trayectoria marcada, pero de forma imprecisa ya que debe realizar correcciones de forma constante para seguir su referencia. La línea naranja muestra como el sistema de procesamiento y análisis de

imagen junto con los algoritmos de cálculo de desplazamiento determinan de forma satisfactoria el recorrido a realizar.

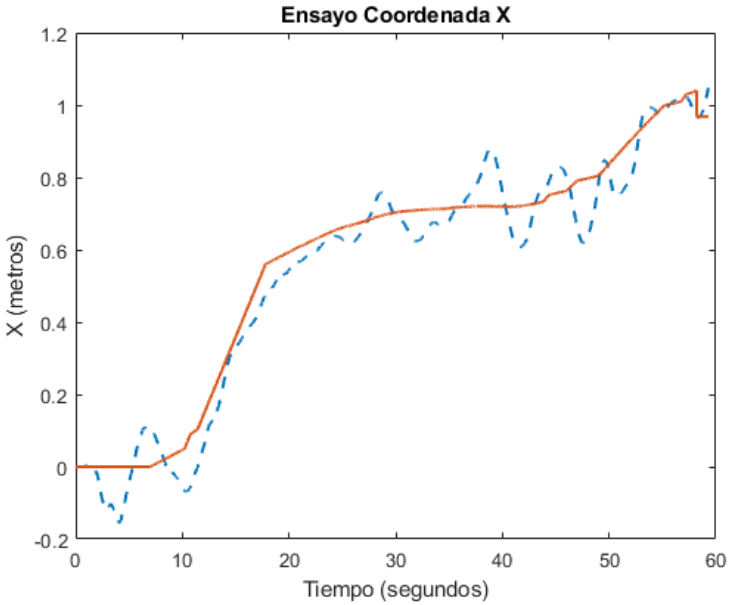


Figura 7.30: Evolución de la coordenada X durante el ensayo de seguimiento de trayectorias en el *hardware*

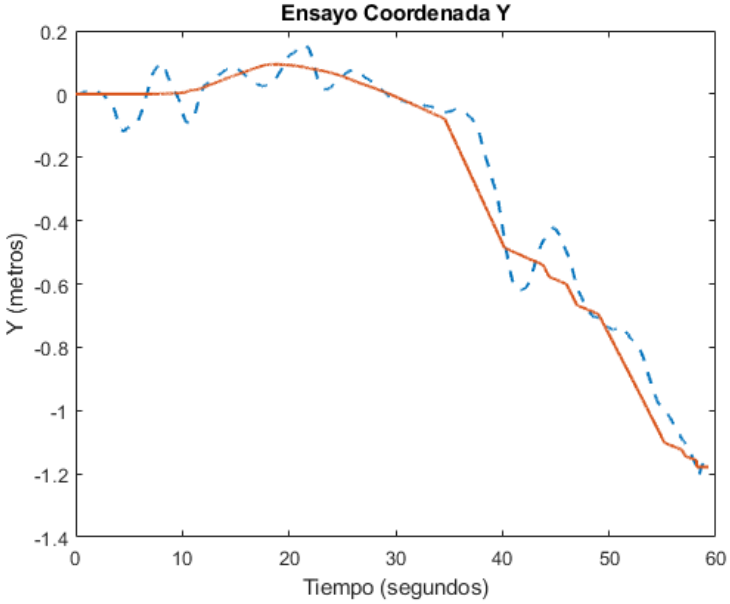


Figura 7.31: Evolución de la coordenada Y durante el ensayo de seguimiento de trayectorias en el *hardware*

El seguimiento en el eje horizontal XY es satisfactorio pero sufre de imprecisiones. Se observan oscilaciones entorno a la trayectoria comandada a lo largo de casi todo

el recorrido. Estas imprecisiones son resultado de los límites del Parrot Mambo, su capacidad de computación y su planta motora no son capaces de seguir los comandos en tiempo real de forma perfecta. Sin embargo, estas limitaciones no son suficientes para impedir la realización de la tarea propuesta.

Además, se ha de tener en cuenta que la estimación del estado del Parrot Mambo tampoco es completamente fiable. La estimación de su posición a lo largo del tiempo no es exacta ya que si se observa la distancia total recorrida en los ejes X e Y en las Figuras 7.30 y 7.31 esta resulta ser menor que las dimensiones del recorrido efectuado.

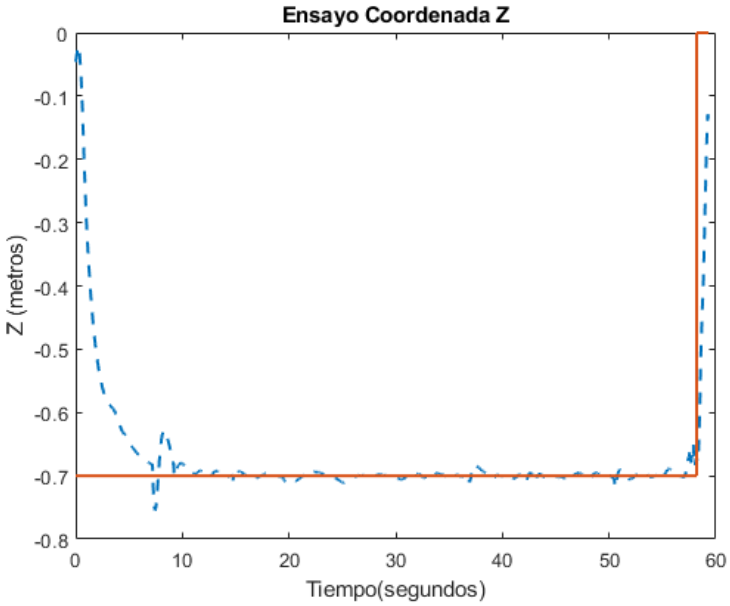


Figura 7.32: Evolución de la coordenada Y durante el ensayo de seguimiento de trayectorias en el *hardware*

El seguimiento en el eje vertical es mejor que el horizontal y también se asemeja más al simulado. El sistema tarda en alcanzar la altura de referencia pero luego la mantiene de forma constante durante todo el vuelo hasta el aterrizaje.

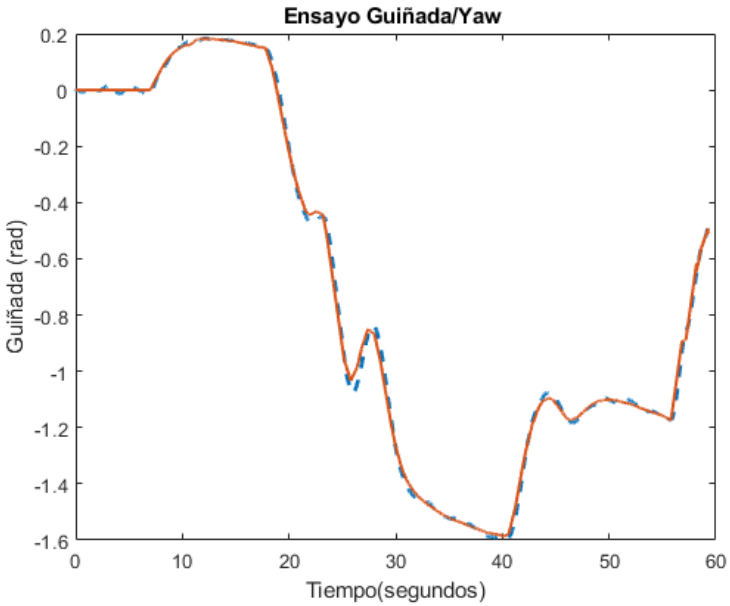


Figura 7.33: Evolución de la guiñada durante el ensayo de seguimiento de trayectorias en el *hardware*

La guiñada sigue su referencia de forma prácticamente perfecta, sin embargo, se ha comprobado a través de los ensayos que la estimación de la guiñada del Parrot Mambo es deficiente. Esto se debe a que la guiñada se determina por la integración directa de su ratio de giro procedente del giroscopio, este ratio es una señal con ruido y además el giroscopio del Parrot Mambo es uno básico, al integrarlo resulta en una estimación imprecisa.

Como resultado el sistema cree que está siguiendo de forma perfecta las órdenes de guiñada, pero la guiñada real y sus lecturas no son iguales. La estimación empeora conforme avanza el el vuelo ya que el error se va acumulando, por ello el último tramo es donde la diferencia entre la guiñada real y la estimada es mayor. El último tramo de la trayectoria tiene una orientación de 1 radián pero sus lectura marca que se encuentra más cercano a los 1.2 radianes.

Capítulo 8

Conclusiones y líneas futuras

La conclusión principal que se extrae es que se ha cumplido el objetivo propuesto en el trabajo, se ha llevado a cabo una aplicación al seguimiento de trayectorias mediante realimentación visual utilizando el mini dron Parrot Mambo. Como se ha mostrado el seguimiento no se realiza de forma perfecta, pero considerando las limitaciones y capacidades del *hardware* empleado los resultados son satisfactorios.

Además, esta propuesta ha servido como plataforma para el estudio y aprendizaje de numerosas disciplinas. Ha servido como puerta de entrada al mundo de los UAS y, más en concreto, al de los mini drones. También ha permitido conocer los principios de funcionamiento de un cuadricóptero, así como sus principios matemáticos.

Por otro lado, se ha hecho necesario conocer el funcionamiento de distintos sensores necesarios para el funcionamiento de un dron, así como el tratamiento de sus datos. El uso de la plataforma de trabajo Matlab/Simulink ha puesto en relieve los beneficios de la metodología de diseño basado en modelos, que mediante el uso de simulaciones ha facilitado el desarrollo. También el entorno de trabajo ha permitido el uso de una serie de herramientas para el ajuste de los controladores de vuelo PID y el procesamiento y análisis de imágenes en tiempo real.

En resumen, se puede decir que el presente trabajo ha servido para conocer el funcionamiento interno de los drones y el diseño de sistemas de control de vuelo. De esta forma se ha completado el objetivo principal de estudiar el control de los drones.

8.1 Consideraciones y líneas futuras

La principal consideración extraída del desarrollo de la aplicación es la limitada capacidad del Parrot Mambo, se debe tener en consideración que este dron tiene un coste de entre los 50 y 70 euros, un coste que en el mundo de los UAS es ínfimo. Esto se refleja en las numerosas limitaciones que demuestra: la conexión *bluetooth* limita mucho el ancho

de banda, la capacidad de computación limitada reduce el rango de posibles aplicaciones, la sensorización no es completamente fiable lo que se traduce en la reducción de la capacidad de control efectiva, el tiempo de vuelo reducido etc.

Todas ellas son naturales que estén presentes en un sistema de coste reducido como es el Parrot Mambo, por ello, se considera que todas las líneas de trabajo futuras pasan por el uso de drones de mayor capacidad. Drones de mayor coste que dispongan de mejor sensorización, conexiones con mayor capacidad, mejor cámara, planta motora más potente etc. Los principios expuestos en este trabajo, así como el tratamiento de imagen y la lógica implementada es aplicable a cualquier sistema que junto a una mejor base de trabajo producirían mejores resultados.

La principal línea de trabajo futura donde se siga utilizando el Parrot Mambo pasa por el uso de un sistema de captura de movimiento como Optitrack (29). Este tipo de sistemas de posicionamiento determinan los movimientos de cualquier objeto mediante el uso de cámaras y, en este caso, se usarían para sustituir el sistema de estimación de posición del Parrot Mambo. De esta forma se tendrían los datos de posicionamiento de una forma mucho más fiable y precisa lo cual permitiría expandir las posibilidades de trabajo.

Si se habla en un término más amplio, las líneas de trabajo futuras para aplicaciones de seguimiento de trayectorias mediante realimentación visual deben centrarse en el ámbito de la vigilancia y reconocimiento. Se podría trabajar en sistemas que supervisasen de forma autónoma recorridos continuos en sus características como podría ser carreteras, tendidos eléctricos, perímetros vallados etc. Las trayectorias a seguir se definirían según el caso y el guiado mediante la imagen de la cámara ofrecerían mayor precisión que el uso del GPS en el caso autónomo, o permitiría sustituir el uso de pilotos. Además, la toma de imágenes se podría aprovechar para la detección de irregularidades como baches en carreteras o brechas en vallados.

Capítulo 9

Presupuesto

En este capítulo se van a especificar los costes relativos al desarrollo del presente trabajo. Se dividirán en gastos de mano de obra y material.

9.1 Mano de obra

Dentro de la sección de mano de obra se va a subdividir entre trabajo de ingeniero y de reparación. Estos costes atienden a la investigación realizada, planteamientos, experimentación y desarrollo de la memoria. El resumen se muestra mediante la siguiente tabla con los costes desglosados y total.

Mano de obra	**Tiempo (horas)**	**Coste por hora (€/h)**	**Coste total €**
Ingeniería	337.5	13.5	4556.2
Reparación	1	10	10
Total			**4566.2**

Tabla 9.1: Desglose de los gastos relacionados con los costes de mano de obra

9.2 Material

La sección de material comprende todas las herramientas utilizadas para llevar a cabo el proyecto. Se incluye también el *software* utilizado, donde se muestra el precio de todos productos asociados a Matlab/Simulink empleados.

Herramienta	Cantidad	Coste por unidad	Coste total €
Ordenador	1	1000	1000
Dron Parrot Mambo	1	60	60
Cartulina Azul	10	0.8	8
Cinta adhesiva	3	2	6
Licencia Anual Matlab/Simulink	1	800	800
Licencia Anual Aerospace Toolbox	1	500	500
Licencia Anual Aerospace Blockset	1	500	500
Licencia Anual Simulink 3D Animation	1	460	460
Licencia Anual Signal Processing Toolbox	1	400	400
Licencia Anual Control System Toolbox	1	460	460
Licencia Anual Image Processing Toolbox	1	400	400
Licencia Anual Simulink Control Design	1	460	460
Total			**5054**

Tabla 9.2: Desglose de los gastos relacionados con los costes de material

9.3 Coste total

Coste	Coste total €
Mano de obra	4566.2
Material	5054
Total	**9620.2**

Tabla 9.3: Desglose de los gastos totales

Apéndice A

Código

```
function BW = createMaskYUV2(R,G,B)
%Máscara para el procesamiento de imagen. Segmenta el color azul del
    fondo
% Auto-generated by colorThresholder app on 15-Jul-2021
%------------------------------------------------------


% Convert RGB image to chosen color space
I = cat(3,R,G,B);
I = rgb2ycbcr(I);

% Define thresholds for channel 1 based on histogram settings
channel1Min = 0.000;
channel1Max = 220.000;

% Define thresholds for channel 2 based on histogram settings
channel2Min = 127.000;
channel2Max = 252.000;

% Define thresholds for channel 3 based on histogram settings
channel3Min = 0.000;
channel3Max = 255.000;

% Create mask based on chosen histogram thresholds
sliderBW = (I(:,:,1) >= channel1Min ) & (I(:,:,1) <= channel1Max) & ...
    (I(:,:,2) >= channel2Min ) & (I(:,:,2) <= channel2Max) & ...
    (I(:,:,3) >= channel3Min ) & (I(:,:,3) <= channel3Max);
BW = sliderBW;
BWEdge=edge(BW);
% Initialize output masked image based on input image.
maskedRGBImage = cat(3,R,G,B);

% Set background pixels where BW is false to zero.
maskedRGBImage(repmat(~BW,[1 1 3])) = 0;

end
```

Listing A.1: Código extraído de la aplicación Color Thresholder, segmenta la imagen de la cámara del dron.

```
%load('PruebaVuelo.mat')
%Grafica el vuelo en 3D (XYZ) mediante los datos obtenidos del dron
t=rt_estimatedStates.time;
X=rt_estimatedStates.signals.values(:,1);
Y=rt_estimatedStates.signals.values(:,2);
Z=rt_estimatedStates.signals.values(:,3);
%xlim([-0.5 2.5])
%ylim([-0.5 2.5])
plot3(Y,X,-Z,'LineWidth',3)
grid on
xlabel('Eje Y')
ylabel('Eje X')
zlabel('Eje Z invertido')
```

Listing A.2: Código para la generación de gráficos 3D.

```
%load('RutaVuelo.mat')
%Grafica las coordenadas frente al tiempo mediante los datos obtenidos
    por el dron
%Muestra tanto la estimación como sus comandos
test=rt_estimatedStates.time;
Xest=rt_estimatedStates.signals.values(:,4);
t=rt_posYaw.time;
X=rt_posYaw.signals.values(:,4);
plot(test,Xest,'--','LineWidth',2)
xlabel('Tiempo(segundos)')
ylabel('Guiñada (rad)')
title('Ensayo Guiñada/Yaw ')
hold on
plot(t,X,'LineWidth',1.5)
```

Listing A.3: Código para la generación de gráficas frente al tiempo. En esta configuración muestra la evolución de la guiñada.

```

%% Plotter Coordenada X

%%
%Script para el dibujo de las gráficas de respuesta ante escalon para...
%la puesta a punto de los PID. (X Coord)

%Los datos se obtienen de la simulación del proyecto
    parrotMinidroneHover
fig=figure;

plot(XYZsim.time,XYZsim.data(:,1),'LineWidth',1.5) %Función orden, escaló
    n tras 10s de simulación
hold on
t=estimatedStates.time;
```

```
X=estimatedStates.signals.values(:,1); %Coord X simulada
plot(t,X,'--','LineWidth',1.5)

%Ajuste grafica
xlabel('Tiempo (segundos)')
ylabel('Coord X (metros)')
title('Respuesta ante escalón. Coord X')
legend('Escalon','Respuesta X')
xlim([0 30])
ylim([-0.5 1.5])
%%Guardamos fichero con los datos

saveas(fig,'xRespuestaP12.fig')
```

Listing A.4: Código para la generación de gráficas con los datos procedentes del simulador. En esta configuración muestra la coordenada X. Códigos análogos se han utilizado para graficar las coordenadas Y y Z.

```
%% Plotter Velocidad DX

%%
%Script para el dibujo de las gráficas de respuesta ante escalon para...
%la puesta a punto de los PID. (DX Vel)

%Los datos se obtienen de la simulación del proyecto
    parrotMinidroneHover

fig=figure;

grid on
t=estimatedStates.time;
DX=estimatedStates.signals.values(:,7); %Coord DX simulada
plot(t,DX,'--','LineWidth',1.5)

%Ajuste gráfica
xlabel('Tiempo (segundos)')
ylabel('Velocidad Dx (metros/segundo)')
title('Velocidad lineal X')
legend('Escalon orden','Respuesta X')
xlim([0 30])
ylim([-1.5 0.5])
%Guardamos figura con la gráfica
saveas(fig,'zRespuestaBase.fig')
```

Listing A.5: Código para la generación de gráficas con los datos procedentes del simulador. En esta configuración muestra la velocidad lineal en X. Códigos análogos se han utilizado para graficar las velocidades lineales de Y y Z.

```
%% Plotter Guiñada Yaw

%%
%Script para el dibujo de las gráficas de respuesta ante escalon para...
%la puesta a punto de  los PID. (Yaw)
```

```
6
% Los datos se obtienen de la simulación del proyecto
    parrotMinidroneHover
8
fig=figure;
10
plot(YawRollPitchsim.time,YawRollPitchsim.data(:,1),'LineWidth',1.5) %
    Función orden, escalón tras 10s de simulación
12 hold on
grid on
14 t=estimatedStates.time;
Yaw=estimatedStates.signals.values(:,4); %Yaw simulado
16 plot(t,Yaw,'--','LineWidth',1.5)

18 % Ajuste gráfica
xlabel('Tiempo (segundos)')
20 ylabel('Yaw (rad)')
title('Guiñada')
22 legend('Escalon orden','Respuesta Yaw')
xlim([0 30])
24 ylim([-1.5 3])
% Guardamos figura con la gráfica
26 saveas(fig,'yawRespuestaBuena.fig')
```

Listing A.6: Código para la generación de gráficas con los datos procedentes del simulador. En esta configuración muestra la guiñada. Códigos análogos se han utilizado para graficar los ángulos de orientación Alabeo y Cabeceo.

```
%% Plotter Velocidad Angular r (yaw)
2
%%
4 % Script para el dibujo de las gráficas de respuesta ante escalon para...
% la puesta a punto de los PID mediante. (r Vel)
6
% Los datos se obtienen de la simulación del proyecto
    parrotMinidroneHover
8
fig=figure;
10
grid on
12 t=estimatedStates.time;
r=estimatedStates.signals.values(:,12); %Velocidad angular p simulada
14 plot(t,r,'--','LineWidth',1.5)

16 % Ajuste gráfica
xlabel('Tiempo (segundos)')
18 ylabel('Velocidad angular r (rad/segundo)')
title('Velocidad angular r (Guiñada/Yaw)')
20 legend('Escalon orden','Respuesta r')
xlim([0 30])
22 ylim([-1.5 0.5])
% Guardamos figura con la gráfica
24 saveas(fig,'zRespuestaBase.fig')
```

Listing A.7: Código para la generación de gráficas con los datos procedentes del simulador. En esta configuración muestra la velocidad angular de guiñada. Códigos análogos se han utilizado para graficar las velocidades angulares Alabeo y Cabeceo.

Apéndice B

Figuras

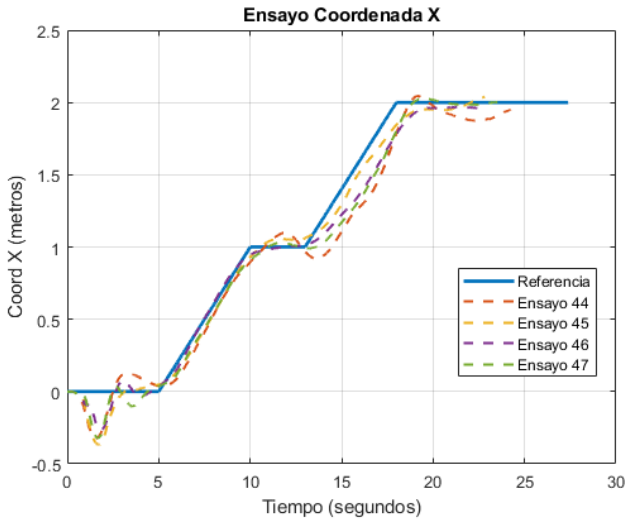


Figura B.1: Gráfica de ensayos de circuito en X. Ensayos 44-47

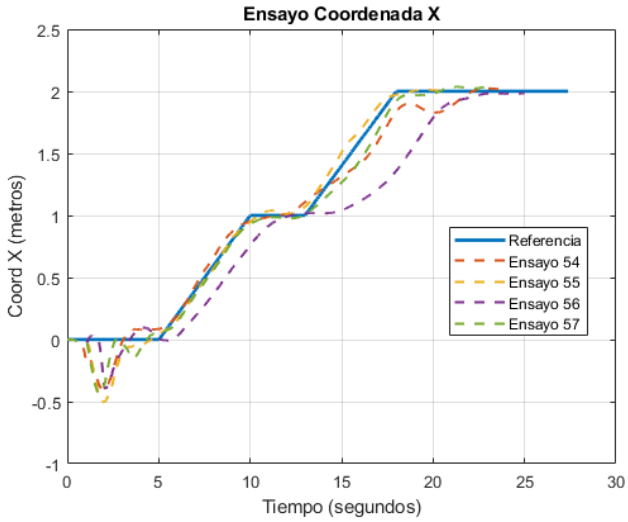


Figura B.2: Gráfica de ensayos de circuito en X. Ensayos 54-57

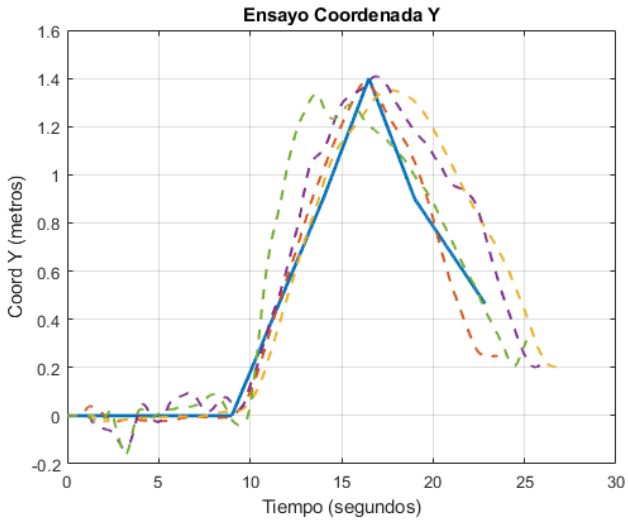


Figura B.3: Gráfica de ensayos de circuito en Y. Ensayos 47-50

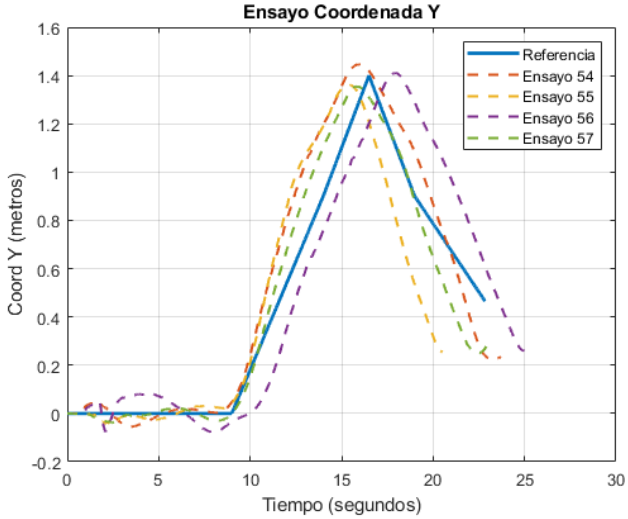


Figura B.4: Gráfica de ensayos de circuito en Y. Ensayos 54-57

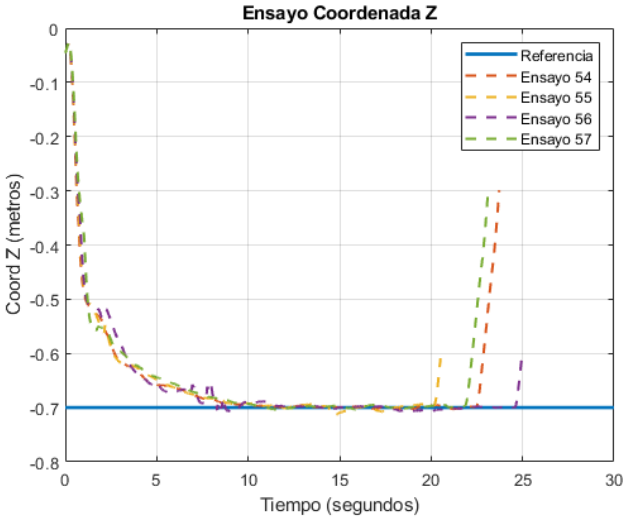


Figura B.5: Gráfica de ensayos de circuito en Z. Ensayos 54-57

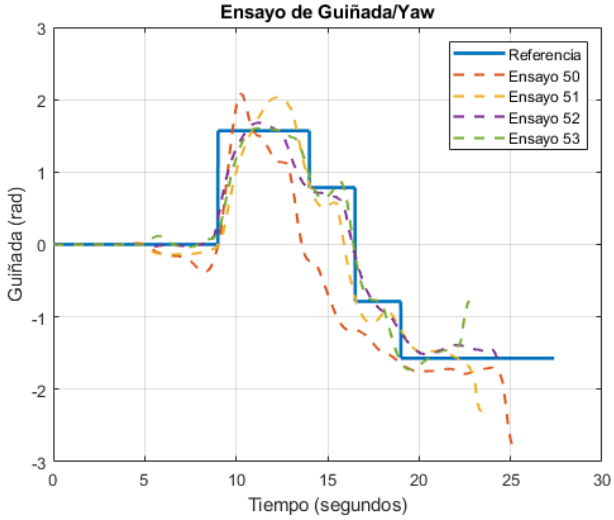


Figura B.6: Gráfica de ensayos de circuito en guiñada. Ensayos 50-53

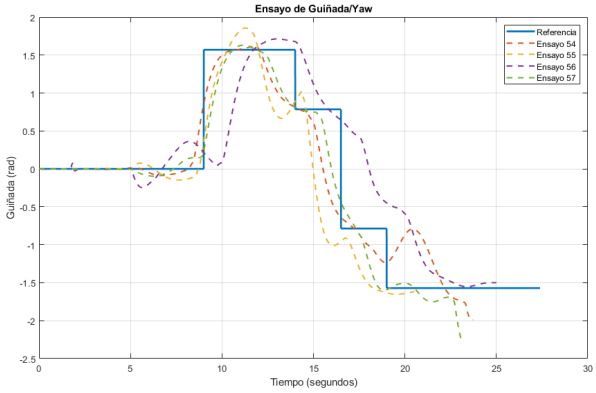


Figura B.7: Gráfica de ensayos de circuito en guiñada. Ensayos 54-57

Bibliografía

[1] Minidrone-js. library for parrot minidrones. URL: https://github.com/Mechazawa/minidrone-js.

[2] Amazon. Parrot mambo. URL: https://www.amazon.es/Parrot-Mambo-Fly-cuadric\%C3\%B3ptero-programable/dp/B074TGFML6.

[3] BP. Drones provide bp with eyes in the skies. URL: https://www.bp.com/en/global/corporate/news-and-insights/reimagining-energy/drones-provide-bp-eyes-in-the-skies.html.

[4] Britannica. Rigid bodies. URL: https://www.britannica.com/science/mechanics/Rigid-bodies.

[5] Levitate Capital. The future of the drone economy. URL: https://levitatecap.com/levitate/wp-content/uploads/2020/12/White-Paper-v4.pdf.

[6] Andrea Civita, Simone Fiori, and Giuseppe Romani. A mobile acquisition system and a method for hips sway fluency assessment. *Information*, 9:321, 12 2018. https://doi.org/10.3390/info9120321 doi:10.3390/info9120321.

[7] Gobierno de España. Ministerio de Fomento. Plan estratégico para el desarrollo del sector civil de los drones en españa 2018-2021. URL: https://www.mitma.gob.es/el-ministerio/planes-estrategicos/drones-espania-2018-2021.

[8] Brian Douglas. Videos and webinar series. drone simulation and control. URL: https://www.mathworks.com/videos/series/drone-simulation-and-control.html.

[9] Control Dron. Dron parrot mambo. URL: https://www.controldron.com/dron-parrot-mambo/.

[10] Embention. Cómo elegir la imu adecuada para un uav. URL: https://www.embention.com/es/news/la-imu-adecuada-para-un-uav/.

[11] Marcos Roa Escobar. Diseño del control de navegación de un dron parrot mambo. URL: https://repositorio.comillas.edu/xmlui/bitstream/handle/11531/32123/TFG-Roa\%20Escobar\%2C\%20Marcos.pdf?sequence=1&isAllowed=y.

[12] Jorge Daniel Gallo Sanabria, Paula Andrea Mozuca Tamayo, and Rafael Iván Rincón Fonseca. Autonomous trajectory following for an uav based on computer vision. URL: https://dialnet.unirioja.es/servlet/articulo?codigo=7373062.

[13] Masoud Gheisari, Javier Irizarry, and Bruce Walker. Uas4safety: The potential of unmanned aerial systems for construction safety applications. pages 1801–1810, 05 2014. https://doi.org/10.1061/9780784413517.184 doi:10.1061/9780784413517.184.

[14] Daniel García González. Aplicación de matlab/simulink al posicionamiento y control de drones en interiores. URL: https://ebuah.uah.es/dspace/handle/10017/44709.

[15] Christine M. Branche John Howard, Vladimir Murashov. Unmanned aerial vehicles in construction and worker safety. URL: https://pubmed.ncbi.nlm.nih.gov/29027244/.

[16] Khristopher Kabbabe Poleo, William J. Crowther, and Mike Barnes. Estimating the impact of drone-based inspection on the levelised cost of electricity for offshore wind farms. *Results in Engineering*, 9:100201, 2021. URL: https://www.sciencedirect.com/science/article/pii/S2590123021000025, https://doi.org/https://doi.org/10.1016/j.rineng.2021.100201 doi:https://doi.org/10.1016/j.rineng.2021.100201.

[17] Mehmet Karahan and Cosku Kasnakoglu. Modeling and simulation of quadrotor uav using pid controller. In *2019 11th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, pages 1–4, 2019. https://doi.org/10.1109/ECAI46879.2019.9042043 doi:10.1109/ECAI46879.2019.9042043.

[18] MathWorks. Aerospace blockset. URL: https://www.mathworks.com/products/aerospace-blockset.html.

[19] MathWorks. Aerospace toolbox. URL: https://www.mathworks.com/products/aerospace-toolbox.html.

[20] MathWorks. Color thresholder. URL: https://www.mathworks.com/help/images/ref/colorthresholder-app.html.

[21] MathWorks. Control system toolbox. URL: https://www.mathworks.com/products/control.html.

[22] MathWorks. Image processing toolbox. URL: https://www.mathworks.com/products/image.html.

[23] MathWorks. Parrot minidrones support from simulink. URL: https://www.mathworks.com/hardware-support/parrot-minidrones.html.

[24] MathWorks. Signal processing toolbox. URL: https://www.mathworks.com/products/signal.html.

[25] MathWorks. Simulink 3d animation. URL: https://www.mathworks.com/products/3d-animation.html.

[26] MathWorks. Simulink control design. URL: https://www.mathworks.com/products/simcontrol.html.

[27] MathWorks. White paper. model-based design for embedded control systems. URL: https://www.mathworks.com/content/dam/mathworks/white-paper/gated/model-based-design-with-simulation-white-paper.pdf.

[28] Dr. Amy McGovern. Pyparrot. python interface for parrot drones. URL: https://github.com/amymcgovern/pyparrot.

[29] OptiTrack. Optitrack for robotics. URL: https://optitrack.com/applications/robotics/.

[30] Goldman Sachs Global Investment Research. Drones: Reporting for work. URL: https://www.goldmansachs.com/insights/technology-driving-innovation/drones.

[31] Ryze Robotics. Sdk 2.0 user guide. URL: https://dl-cdn.ryzerobotics.com/downloads/Tello/Tello\%20SDK\%202.0\%20User\%20Guide.pdf.

[32] Ryze Robotics. Tello. URL: https://www.ryzerobotics.com/es/tello.

[33] S STEVANOVIC, J KASAC, and J STEPANIC. Robust tracking control of a quadrotor helicopter without velocity measurement. URL: http://repozitorij.fsb.hr/6225/.

[34] DJI Store. Dji tello edu. URL: https://m.dji.com/es/product/tello-edu.

[35] Revista Time. Drones are helping catch poachers operating under cover of darkness. URL: https://time.com/5279322/drones-poaching-air-shepherd/.

[36] Ángel Bello Guisado. Diseño de controladores de vuelo para un dron modelo parrot mambo minidrone. URL: https://riunet.upv.es/handle/10251/117441.