FACULTAD DE INGENIERÍA DE

SISTEMAS Y ELECTRÓNICA

TESIS

Título:

SISTEMA DE CONTROL DE ACTITUD DE UN UAV TIPO QUADROTOR USANDO UNA FPGA

Bachiller:

Eduardo Pillco Porlles

Asesor:

Ing. José Ambrosio Machuca Mines

Para obtención del Título de:

INGENIERO ELECTRÓNICO

Lima, 2017

Dedicatoria

A Dios por su gracia y misericordia, por ser el guía que conduce mi camino y por darme la fe para poder concluir este trabajo. A mi familia y novia por su apoyo incondicional durante este trabajo de tesis.

Agradecimientos

Le agradezco profundamente a Dios por la vida que me ha dado, porque sé que siempre me ha acompañado y esta aventura que fue mi etapa universitaria no fue la excepción. Gracias por darme fortaleza cuando más lo necesitaba y por permitirme llegar hasta este punto de mi vida.

Agradezco a mi familia por apoyarme siempre en todos mis proyectos de vida, a mis padres Amérida y Toribio por su dedicación, esfuerzo y por confiar en mí, a mi hermana Rocío por darme su apoyo incondicional, por sus valiosos consejos y ser un ejemplo a seguir.

A mi novia Giuliana por ser parte importante de mi vida, por todo su amor, apoyo y ánimos en momentos difíciles y porque sé que siempre puedo contar con ella.

A mis profesores quienes me formaron profesionalmente y por darme muchos de los conocimientos que he logrado obtener, al profesor Eleazar Sal y Rosas quien inicio el proyecto, a la profesora Elizabeth Villota por sus conocimientos y quien junto al profesor Eleazar formaron en mí el amor por la investigación. Al profesor Miguel Risco por apoyar el proyecto con sus conocimientos de VHDL y FPGAs. Al profesor Alberto Alvarado por su apoyo en gestiones para poder llevar un curso relacionado al tema de tesis, y diversas oportunidades de presentaciones en las cuales pude exhibir el proyecto. A mi asesor el Ing. José Ambrosio Machuca por su orientación, consejos y seguimiento en la redacción de esta tesis.

A mis compañeros del proyecto Quadrotor con quienes compartí momentos importantes de mi vida, y muchas anécdotas que quedaran en el baúl de los recuerdos, a Julio Mejía, Ronny Huerta, Luis Rueda, Carlos Quispe y en especial a Miguel Chicchon por su amistad y apoyo incondicional desde el nacimiento de este proyecto de tesis hasta su término, por sus conocimientos y paciencia al guiar al grupo.

Resumen

En este trabajo de tesis se ha desarrollado el sistema de control de actitud para un vehículo aéreo no tripulado del tipo Quadrotor, es decir el computador de vuelo o también conocido como autopiloto. El trabajo desarrollado abarca el diseño e implementación hardware y software del sistema en mención en un dispositivo de lógica reconfigurable (FPGA).

Para el desarrollo de la arquitectura hardware del computador de vuelo se ha diseñado e implementado los periféricos descritos con el lenguaje VHDL que van conectados al procesador NIOS II de Altera. Los periféricos diseñados cumplen las funciones de adquisición de sensores, control de motores, adquisición de señales de referencia de un mando a control remoto y la descarga de telemetría a una estación terrena de control.

Se ha identificado y validado el modelo matemático del Quadrotor utilizando el software Matlab para realizar pruebas de control y diseñar el controlador de actitud.

El diseño de control de actitud propuesto es de dos lazos de control en cascada, un lazo interno que controla las velocidades angulares de las rotaciones y un lazo externo que controla los ángulos de orientación.

El software desarrollado contiene el controlador de actitud y la transmisión del estado de vuelo a una estación terrena de control que es una aplicación que se ejecuta en un ordenador para monitorear el estado de vuelo del Quadrotor y se ha desarrollado con el lenguaje Java.

Este proyecto de tesis aporta el diseño de un computador de vuelo para propósitos de investigación. El control de actitud es la función básica del Quadrotor y la FPGA tiene capacidad para albergar más periféricos y/o un procesador más si se requiere realizar aplicaciones de mayor nivel utilizando como base el computador de vuelo diseñado.

Palabras claves: Quadrotor, FPGA, VHDL, control, NIOS II.

Abstract

This thesis describes the development of an Attitude Control System (i.e. the Flight Controller or also known as Autopilot) for an unmanned aerial vehicle (UAV) of Quadrotor type. The scope of this work covers the design and implementation of the hardware and software for the Flight Controller using a programmable logic device (FPGA).

The hardware architecture for the Flight Controller has been designed and implemented using peripherals described in VHDL language and connect to the Altera NIOS II Processor. The peripherals provide sensor reading, engine control, reading reference signals from a remote control, and download telemetry data to a ground control station.

The mathematic model for the Quadrotor has been determined and validated using Matlab software in order to perform control testing and design the Attitude Control System.

The Attitude Control System has two cascading control loops, an internal loop controls the angular velocities of the rotations and an external loop controls the angles of orientation.

The software contains the attitude controller and transmits the flight status to a land control station. The land control station is a Java application running on a computer and used to monitor the flight status of the Quadrotor.

This project contributes with the design and implementation of a flexible Flight Controller that can be used for research purposes. The Flight Controller is the main function for the Quadrotor and the FPGA has the capacity to hold more peripherals and/or an extra processor if a higher level of application is required.

Keywords: Quadrotor, FPGA, VHDL, control, NIOS II.

Índice General

Índice de Tablas

Índice de figuras

CAPÍTULO 1

INTRODUCCIÓN

En los últimos años los UAVs (por sus siglas en inglés, *Unmanned Aerial Vehicle*) ha aumentado el interés en su estudio, ya que supone un reto controlarlos y hacer que sean autónomos a fin de cubrir diversas aplicaciones que se están dando en la actualidad con lo cual se han vuelto una herramienta necesaria. Se usa tanto en el área militar como el civil en una amplia gama de aplicaciones como filmaciones aéreas, agricultura de precisión, video vigilancia, búsqueda y rescate de personas, inspección de terrenos, etc.

En el pasado, el desarrollo de estos vehículos aéreos en escala miniatura era mucho más complicada debido a las restricciones del hardware disponible. Los avances tecnológicos en sensores y actuadores con la tecnología MEMS (de sus siglas en inglés, *Micro Electromechanical Systems*), las mejores en suministros de almacenamiento de energía y los nuevos procesadores más veloces han hecho posible la construcción de robots aéreos autónomos.

En este trabajo se eligió un UAV de tipo Quadrotor, este tiene como base de sustentación cuatro rotores dispuestos simétricamente equidistantes del centro de la aeronave que tiene la forma de una cruz como se observa en la figura 1.1.

Figura 1.1: Quadrotor DJI F450

El Quadrotor presenta muchas ventajas con respecto a otros tipos de aeronaves como un aumento en la capacidad de carga debido a la suma de los empujes que generan los cuatro rotores, una alta maniobrabilidad que le permiten volar en entornos complicados y no necesita una pista de aterrizaje porque tiene un despegue y aterrizaje vertical.

Esta aeronave presenta dinámicas rápidas debido a la interacción de los cuatro rotores, por lo cual necesita un sistema que logre estabilizarlo controlando los ángulos de orientación en tiempo real. Se decidió usar una FPGA como controlador debido a su alto rendimiento para procesar información a altas velocidades, necesario para el control del vehículo, además que tiene una alta flexibilidad dado que es posible diseñar arquitecturas de hardware específico para los requerimientos del proyecto, tiene un bajo consumo de energía, y puede implementar varios procesadores en su integrado lo que hace del

dispositivo FPGA, una tecnología *System on Chip* (SoC). Si bien no fue necesario usar más de un procesador para realizar el control de actitud del vehículo, pero se encontró que la FPGA contaba con los recursos sobrantes suficientes para la implementación de otras tareas que junto al controlador diseñado pueda usarse para aplicaciones de más alto nivel.

La tecnología FPGA es considerada cada vez por más diseñadores en varios campos de aplicación como telecomunicaciones, procesamiento de señales e imágenes, robótica, sistemas aeroespaciales y de defensa, prototipos de ASICs, etc.

El proyecto abarca desde el diseño de la arquitectura del hardware del computador de vuelo descrito en la FPGA hasta el diseño e implementación del algoritmo de control de actitud del Quadrotor para mantenerlo en *hovering*, es decir suspendido en el aire a una altura considerable con el plano que conforman las hélices paralelo al piso.

1.1 Planteamiento del problema

En la actualidad los UAVs son considerados una herramienta útil tanto en el ámbito militar y civil en los cuales están siendo usados en diversas aplicaciones. Uno de los UAVs más populares es el Quadrotor por sus características de vuelo como su alta maniobrabilidad que hacen de este vehículo uno de los preferidos. Para que estos vehículos aéreos vuelen, necesitan un autopiloto o computador de vuelo que guíe su funcionamiento de forma autónoma y casi sin intervención humana.

La dinámica de un Quadrotor suele ser muy rápida ya que tiene cuatro actuadores que ejercen fuerzas de empuje constantemente sobre su cuerpo lo que provoca que el sistema sea inestable, por lo cual se requiere de un computador de vuelo que tenga el potencial para adquirir la información de los sensores inerciales y ejecutar el algoritmo de control de orientación o actitud para establecer el vuelo del Quadrotor. El control de actitud es la función base que puede ser usado junto a otras tareas como navegación y guiado, aplicaciones de visión artificial u otras aplicaciones de mayor nivel.

Existen computadores comerciales que cumplen con los requisitos básicos de mantenerlo en vuelo pero el código es cerrado lo que impide su modificación y los de código abierto tienen capacidades limitadas, con pocos recursos sobrantes para implementar aplicaciones e integrar algunos sensores o carga útil necesarios para el cumplimiento de alguna tarea específica. Esto hace que los computadores de vuelo comerciales no sean tan útiles en proyectos de investigación con UAVs.

Por tal motivo este proyecto propone el diseño de un computador de vuelo para controlar la actitud del Quadrotor y que pueda ser reusado en otros proyectos como una plataforma de investigación para extender sus funcionalidades. Para esto se debe diseñar la arquitectura del hardware del computador de vuelo que dispondrá del software de control de actitud del vehículo, tendrá recursos sobrantes suficientes para realizar otras tareas y la capacidad para reconfigurar el hardware de tal manera que se pueda personalizar si se desea modificar o agregar otras funciones.

1.2 Justificación

La navegación no tripulada se ha vuelto hoy en día en una línea de investigación de importancia pero a la vez de cierto grado de complejidad debido a que necesita de componentes de alto rendimiento y algoritmos de control robustos para la navegación.

Existen numerosas aplicaciones donde las aeronaves no tripuladas pueden ser utilizadas, lo que ha generado el interés de muchos grupos de investigación, además de empresas en todo el mundo que se dedican a desarrollar y comercializar productos utilizando esta tecnología.

Diseñar un computador de vuelo para un Quadrotor con características y especificaciones propias, aporta una plataforma de investigación importante tanto para el presente proyecto como para futuros trabajos de investigación. Implementar el sistema de control, tanto la arquitectura del hardware como el firmware, y validar las leyes de control permite adquirir

experiencia y ayuda a comprender mejor el sistema, ya que en la práctica ocurren fenómenos que en ocasiones no son considerados previamente. De esta manera controlar un sistema no lineal de 6 DoF (de sus siglas en ingles *degrees of freedom*), crea un desafío mediante el cual se generan nuevos conocimientos en esta línea de investigación.

1.3 Objetivos

1.3.1 Objetivo general

Diseñar e implementar la arquitectura del hardware, firmware y el software de un sistema de control de actitud usando un dispositivo FPGA para mantener a un Quadrotor en *hovering* en presencia de perturbaciones.

1.3.2 Objetivos específicos

Los objetivos específicos a lograr son:

- Desarrollar el modelo matemático para describir la dinámica del Quadrotor e identificar experimentalmente los parámetros del sistema físico.
- Elaborar un simulador de vuelo usando el software Matlab en el cual se ejecuten y prueben los algoritmos de control.
- Diseñar e implementar la arquitectura hardware del sistema de control en la tarjeta de desarrollo DE0-Nano.
- Diseñar los algoritmos de control PID de actitud, simularlo en el software Matlab e implementarlo en la FPGA DE0-Nano para experimentos tanto en banco de pruebas como en vuelos en tiempo real.

1.4 Estado del arte

En esta sección se detallara los trabajos más destacados realizados sobre Quadrotores en los últimos años. Se ha dividido la sección en dos partes, en la primera se presenta los resultados de grupos de investigación más resaltantes y en la segunda, los Quadrotores comerciales más populares.

1.4.1 Plataformas de investigación

En (S. Bouabdallah P. M., 2004a) se presentó un artículo sobre el control de Quadrotores de tamaño reducido, particularmente uno denominado OS4. En este trabajo se diseñó una estrategia de control, basada en la estabilización del subsistema de rotación mediante funciones de Lyapunov, junto con un control de la altura mediante linealización por realimentación. Luego en (S. Bouabdallah A. N., 2004b), los mismos autores presentaron un modelo dinámico para el Quadrotor mediante la formulación de Lagrange-Euler, considerando además la dinámica de los actuadores. Se implementó los controladores PID y LQ y se hizo una comparación de resultados entre ellos donde resultó que el controlador PID es más eficaz que el control óptimo debido a que éste considera la dinámica de los rotores.

Un año después en (S. Bouabdallah R. S., 2005) se realizaron trabajos en la misma plataforma OS4 pero esta vez implementando dos controladores no lineales: *Backstepping* y *Sliding-Mode*. El modelo matemático se obtuvo usando el formalismo de Newton-Euler. En primer lugar se controló el sub-sistema de rotación angular empleando funciones de Lyapunov, con el objetivo de estabilizar el Quadrotor. Posteriormente se controló la altura y el desplazamiento lineal. Los resultados mostrados en simulación se validaron experimentalmente en el sistema real.

En (Guilherme V. Raffo, 2008a) se presenta una estrategia de control no lineal para resolver problemas de seguimiento de trayectorias. Las ecuaciones de movimiento del vehículo son

obtenidas del formalismo Lagrange-Euler. El controlador de actitud es realizado con técnicas de control robusto que permiten rechazar perturbaciones mantenidas y el sistema de traslación lineal es controlado con la técnica *Backstepping*. Se presentan resultados satisfactorios en simulaciones aun con presencia de perturbaciones que generan momentos aerodinámicos. Otra propuesta de los mismos autores (Guilherme V. Raffo, 2008b) es utilizar una estrategia de control predictivo basado en modelo para el sistema de traslación lineal, ésta emplea linealizaciones sucesivas para obtener el modelo del error en cada paso de muestreo.

Adicionalmente al control inercial, el Quadrotor ha sido también controlado mediante realimentación por visión artificial. En (Najib Metni, 2007), la estimación de la posición y orientación se realizó mediante visión utilizando una técnica de control servo visual basada en homografías. La trayectoria deseada se obtiene utilizando un conjunto de imágenes pre grabados, comparando la imagen actual con la imagen deseada a una imagen de referencia por las matrices de homografía en cada paso. Para determinar el vector de traslación se estima la información de la profundidad de referencia usando una ley de control adaptativa.

En los últimos años los temas de investigación con respecto a estas aeronaves han ido aumentando y muchas universidades en el mundo cuentan con laboratorios especializados para su estudio (MIT, s.f.)**,** (University of Pennsylvania, s.f.)**,** (Zurich, ETH, s.f.). Sus trabajos se basan mayormente en el control visual, utilizando un arreglo de cámaras para detectar la posición, orientación y velocidades de la aeronave. Algunos de los temas que se encuentran actualmente en investigación son: enjambres de Quadrotores, vuelo en formación, trabajo colaborativo, maniobras agresivas, aprendizaje, etc.

1.4.2 Plataformas comerciales

Actualmente existen muchas empresas que se dedican a comercializar UAVs tanto de ala fija como de ala rotatoria. Uno de los Quadrotores comerciales más populares es el Parrot AR Drone, el cual es capaz de volar en entornos cerrados y puede ser controlado remotamente con un teléfono inteligente como un iPhone o Ipad. Posee cámaras de video en la parte delante e inferior y sensores acelerómetro, giroscopios y ultrasonido. No tiene capacidad de ser expandido o mejorado mediante cargas útiles y su finalidad es puramente recreativa (Parrot SA., s.f.).

Figura 1.2: Parrot AR. Drone 2.0

Dragan Fly Innovations INC es una empresa Canadiense especializada en micro aeronaves desde hace más de 10 años. Su producto más popular es el Draganflyer X4, el cual se usa tanto para entretenimiento como para investigación en importantes universidades. Es una plataforma confiable, su construcción es de fibra de carbono lo que lo hace ligero y resistente. Cuenta con un estabilizador de cámara con el cual la toma y las fotografías aéreas son estables (Draganfly Innovations Inc., s.f.). En la figura 1.3 se muestra éste vehiculo aéreo.

Figura 1.3: Dragan Fly Innovations INC

Por otra parte, un modelo de Quadrotor muy solicitado en el mercado es el “Phantom” de la empresa DJI que actualmente se encuentra en su versión 4. Este modelo permite volar en entornos abiertos y puede cargar una cámara de alta resolución. Está equipado con un receptor de GPS y es capaz de volar en trayectorias planificadas previamente desde un computador o teléfono portátil, y en caso de pérdida de enlace de radio tiene la capacidad de volver automáticamente al punto de partida además de tener diferentes modos de vuelo y la capacidad de evitar obstáculos. Puede llegar a velocidades de vuelo de hasta 20m/s y un tiempo de autonomía máximo de 28 minutos (DJI, s.f.). El Quadrotor Phantom se muestra en la figura 1.4.

Figura 1.4: Phantom DJI

Entre los proyectos de código abierto se tiene el Arducopter que se ha desarrollado por la comunidad DIY DRONES. Es una plataforma basada en el computador de vuelo ArduPilot Pro Mega (APM), que controla aeronaves de ala fija, helicópteros multi-rotores y helicópteros tradicionales. Dentro de sus componentes se destaca una IMU de 6 DoF, magnetómetros, estabilizador de cámara, sistema de comunicación para largas distancias y GPS para seguimiento de trayectorias. Es uno de los UAVs más utilizados por su bajo costo por aficionados y desarrolladores (ArduPilot Dev Team, 2015).

Figura 1.5: Arducopter

CAPÍTULO 2

FUNDAMENTO DEL TEMA

2.1 Historia del UAV

A lo largo de la historia el hombre siempre ha mostrado su interés en poder volar lo que lo llevó a muchos intentos por lograr crear objetos voladores.

En los comienzos de la aviación el Quadrotor fue una de las primeras aeronaves en aparecer, inclusive previamente al helicóptero convencional, pero sus problemas mecánicos y su dificultad de control por parte del piloto la hicieron inviable para esa época. El primer helicóptero Quadrotor fue construido en Francia por los hermanos Louis y Jacques Bréguet en el año de 1907, fue conocido con el nombre de Gyroplane N° 1 que se observa en la figura 2.1, estaba conformada por una doble capa de propulsores, la estructura tenía un peso de 578Kg y tenía un piloto en el centro de la cruz que controlaba el vehículo con palancas mecánicas. Debido a la dificultad en su manejo, las pruebas de vuelo no fueron exitosas, alcanzando una altura máxima de tan sólo 60 centímetros (Hirschberg, 2000).

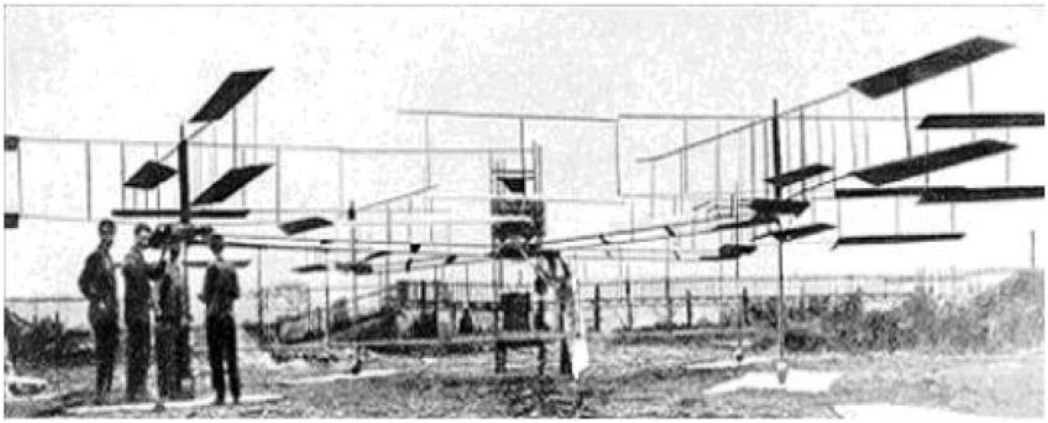


Figura 2.1: Gyroplane N° 1

Años después, en 1922 otro vehículo a gran escala con cuatro rotores de seis palas, fue construido y puesto a prueba por el norteamericano George de Bothezat bajo un contrato con el ejército Norteamericano, quién logró que se sustentara a bajas altitudes y velocidades de avance lento, siendo considerado como el primer vuelo estacionario de la historia en esta clase de aeronaves. Sin embargo debido al insuficiente rendimiento, altos costo financiero el proyecto fue cancelado (Leishman, 2000).

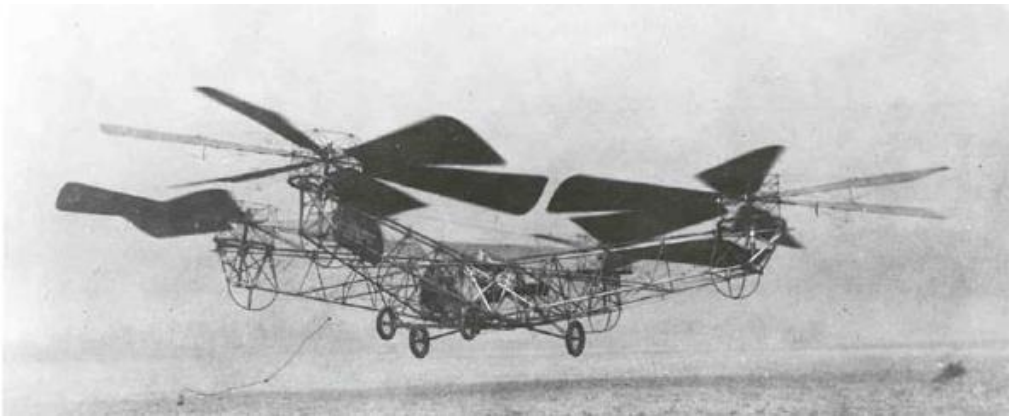


Figura 2.2: Helicóptero de Bothezat

2.2 Principio de movimiento del Quadrotor

El Quadrotor presenta efectos dinámicos complejos debido a los empujes que generan los cuatro rotores que a su vez producen torques sobre el cuerpo del vehículo. Los torques provocan que el Quadrotor tenga cierto grado de inclinación en sus tres ejes de orientación según la fuerza que el torque ejerza sobre el cuerpo y por consecuencia se genera un desplazamiento lateral. El Quadrotor presenta seis grados de libertad (6 DOF), que son los desplazamientos en los ejes que conforman un espacio tridimensional y las rotaciones en los mismos. Los actuadores del Quadrotor solo pueden actuar directamente sobre cuatro grados de libertad que son la altura y las rotaciones en sus ejes como se muestra en la figura 2.3, ya que es imposible que pueda desplazarse horizontalmente sin haber cambiado su actitud, esto hace que sea un sistema sub-actuado porque tiene más grados de libertad que actuadores (Castro, 2001).

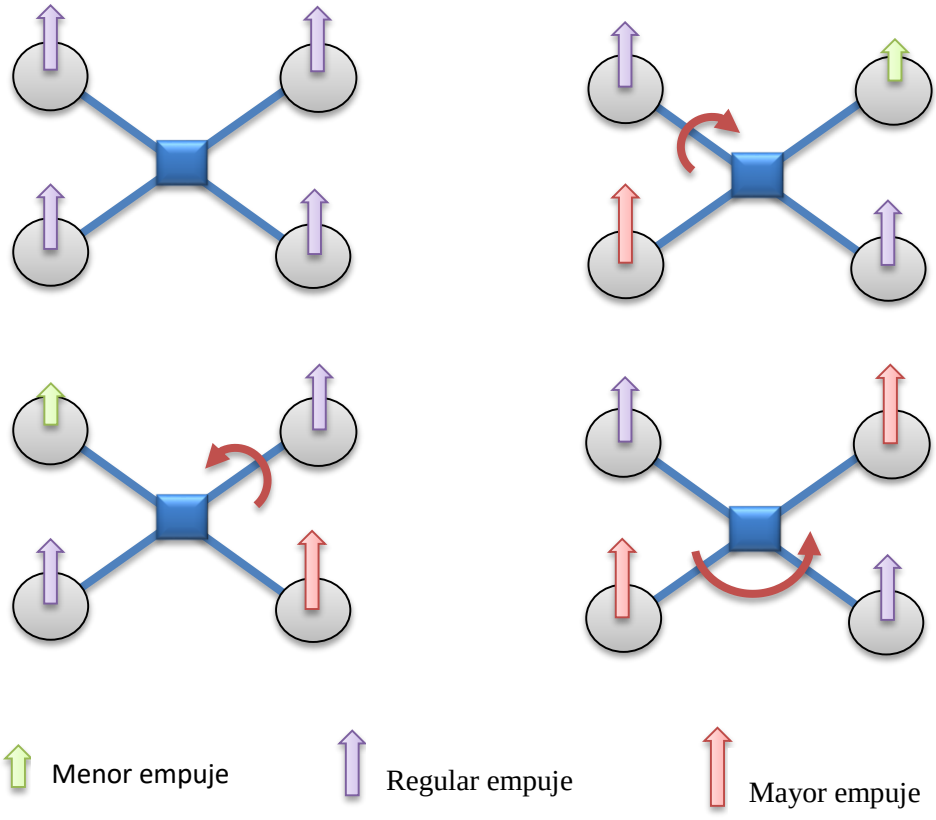


Figura 2.3: Movimientos espaciales del Quadrotor

Las rotaciones generadas, alrededor de sus ejes tridimensionales, por las diferencias de empujes de los actuadores producen los ángulos de orientación que son conocidos como alabeo, cabeceo y guiñada, donde cada ángulo está relacionado con un eje de referencia.

La suma de los cuatro empujes de los actuadores, es el empuje total denominado *throttle* el cual genera un movimiento vertical. Cuando el plano que conforma los brazos del vehículo se encuentra paralelo al piso, es decir no presenta ninguna rotación, el empuje total genera un movimiento puramente vertical en la aeronave, pero basta que se presente un pequeño cambio en la orientación, éste genera un desplazamiento horizontal. De lo dicho anteriormente se entiende que es necesario mantener la aeronave nivelada si se desea que ésta se mantenga en *hovering*, cualquier pequeño cambio en la orientación provocará desplazamientos indeseados.

Una perspectiva horizontal del Quadrotor se puede observar en la figura 2.4, el diagrama de fuerzas muestra que al presentarse un ángulo de inclinación existe una fuerza lateral que produce un desplazamiento en la dirección de la fuerza.

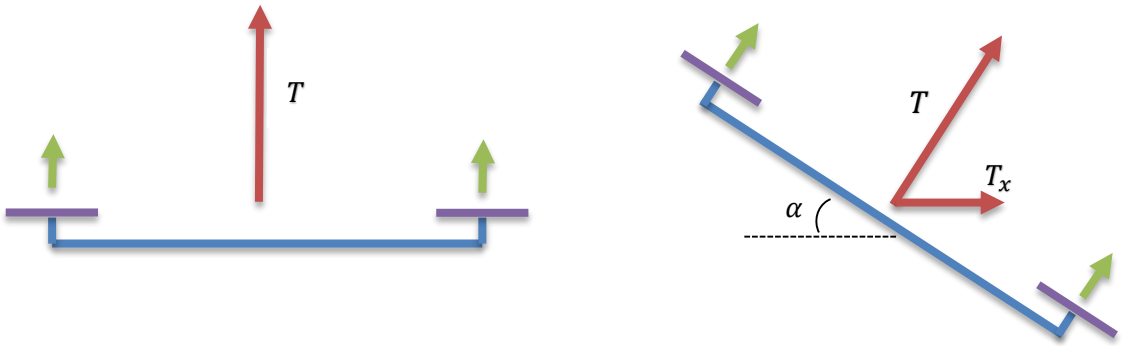


Figura 2.4: Perspectiva horizontal y diagrama de fuerzas

2.3 Componentes básicos de un Quadrotor

2.3.1 Motores DC sin escobillas

Son motores trifásicos utilizados para convertir la fuerza eléctrica provista por la batería a fuerza mecánica que junto con las hélices generan los empujes necesarios para que la aeronave pueda volar.

Los motores BLDCM (*BrushLess DC Motor*) presentan muchas ventajas con respecto a los motores con escobillas:

- ✓ Mayor tiempo de vida útil
- ✓ Mayor relación torque peso
- ✓ Alta eficiencia debido a que no hay rozamiento entre el rotor y estator
- ✓ Menor emisión electromagnética

Estas características además de su bajo peso y confiabilidad hacen que sea muy usado en aplicaciones para UAVs, además también se emplean en sectores industriales tales como: automovilístico, aeroespacial, equipos de automatización e instrumentación.

Estos motores son ideales para el Quadrotor pues son más eficientes y pueden ser acoplados directamente a la hélice sin necesidad de engranes de reducción. Para controlarlos, es necesario conmutar las fases en función de la posición del rotor, por lo que se requiere medir esta posición. En motores más grandes y costosos se realiza con sensores de efecto Hall, pero en el caso de los pequeños, se mide por medio del efecto de la fuerza contra electromotriz (*Back EMF* en inglés) (Jorge M. Cotter, 2010)**.** Para variar la velocidad del motor se utiliza una señal PWM que modula esta entrada de voltaje trifásico, variando la tensión media entregada a cada fase. En la figura 2.5 se observa el motor usado en este proyecto.

Figura 2.5: Motores Brushless DJI 2212

2.3.2 ESC (Controlador electrónico de velocidad)

El controlador electrónico de velocidad de un motor (ESC, del inglés *Electronic Speed Controller*) o también llamado variador permite regular la potencia suministrada al motor transformando la corriente continua de las baterías en una tensión alterna trifásica que energizan a los bobinados del motor en cierta secuencia dependiendo de la posición del rotor y recibe una señal de control PWM para ajustar la velocidad angular del motor.

El ESC permite lograr una velocidad estable en el rotor, sin importar la carga o las perturbaciones que se presenten. En general una hélice se considera una carga con un conjunto de perturbaciones dinámicas lo cual hace de éste un problema de control complejo.

Dado que los motores Brushless suelen funcionar con corrientes que varían entre 15A y 30A, dependiendo de su tamaño, el diseño del variador correspondiente no es sencillo de realizar además que los componentes que contiene no son fáciles de adquirir en el mercado local. Por tal razón se prefirió utilizar controladores de velocidad comerciales que ya se encuentran probados para este tipo de aplicaciones. En este caso se decidió usar el ESC de DJI 30 A Opto que incorpora un microcontrolador digital que acepta señales de control entre 30Hz hasta 400Hz de actualización en forma de señal PWM, además se energiza de Baterías Lipo de 3 celdas o 4 celdas (DJI Wiki, s.f.).

Figura 2.6: ESC DJI 30A OPTO

2.3.3 Hélices

La hélice es un dispositivo constituido por un número variable de aspas o palas que al girar alrededor de un eje producen una fuerza propulsora. Cada pala está formada por un conjunto de perfiles aerodinámicos que cambian progresivamente su ángulo de incidencia desde el eje hasta el extremo (mayor en el eje, menor en el extremo). La hélice está acoplada directamente al eje de salida del motor que le proporciona el movimiento de rotación (Muñoz, s.f.)**.** Tiene dos parámetros importantes que son el diámetro y el paso. El diámetro es el tamaño de punta a punta que tiene la hélice, mientras más diámetro tenga la hélice genera más consumo de energía en el motor. El paso de la hélice es el ángulo que forman sus palas con el eje longitudinal de la aeronave, suponiendo que el eje de giro de la hélice es perpendicular al eje longitudinal del vehículo. El paso de la hélice determina cuanto avanza al dar una vuelta entera. Este concepto es similar a otros elementos mecánicos como el tornillo que también avanza una distancia determinada por su paso sobre un medio sólido cuando gira sobre su eje. En la figura 2.7 se ilustra estas dos propiedades de una hélice en giro.

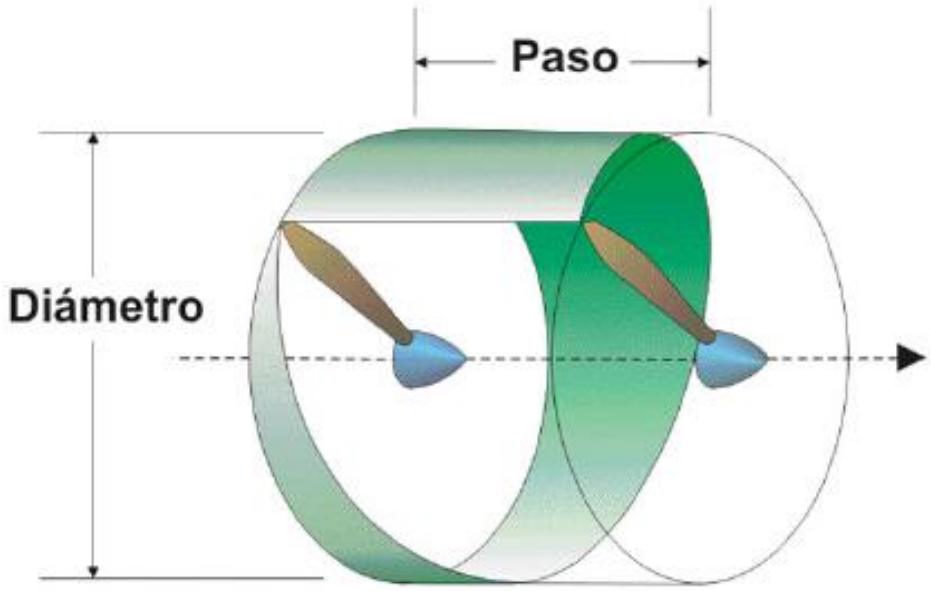


Figura 2.7: Paso y diámetro de una hélice

Normalmente para aplicaciones con Quadrotores estas hélices son de 2 palas con paso fijo pero existen aplicaciones en los cuales el paso de la hélice puede ser variado por el usuario para lograr mejor eficiencia en el sistema.

Las hélices usadas en este proyecto son las DJI 8x4.5 mostradas en la figura 2.8, que quiere decir 8 pulgadas de diámetro y 4.5 de paso fijo. En un Quadrotor se requieren un par de hélices de tipo *push* y otro par de tipo *puller*, la diferencia entre ambas radica en el sentido de giro que siguen para generar el empuje que comandará el vuelo del vehículo aéreo.

Figura 2.8: Imagen de las hélices DJI 8X4.5

2.3.4 Batería Lipo

Las baterías utilizadas para este tipo de aeronaves son de polímero de litio (en inglés, *Lithium Polymer*) por eso la abreviación Lipo. Estas baterías tienen propiedades muy útiles para aplicaciones con Quadrotores como una conveniente relación peso-potencia ya que son livianas y capaces de entregar altas cantidades de corriente necesarias para suministrar potencia a los motores.

La batería que se adquirió para este proyecto es la Turnigy 4500mAh 3S 30C Lipo, que proporciona 11.1v de alimentación, tiene capacidad de entregar 4.5 amperios de corriente por hora y tiene un peso de 386g (Hobbyking, s.f.). Si es que la carga conectada requiere de más corriente entonces el tiempo de autonomía se reduce inversamente proporcional al factor de aumento de corriente suministrada, es decir si se requiere una intensidad de 9A entonces el tiempo de autonomía se reduce a la mitad siendo este de media hora. De esta manera la batería Lipo puede entregar altas cantidades de corriente con la desventaja que el tiempo de autonomía se reducirá. Aun así estas baterías cuentan con un límite de

máxima cantidad de corriente suministrada y esto se determina por la constante de descarga que en este caso es de 30C, con lo cual la batería entregará un máximo de 135A.

Figura 2.9: Batería Turnigy 4500mAh 3S 30C Lipo

2.3.5 Unidad de medición inercial (IMU)

El IMU (de sus siglas en inglés *Inertial Measurement Unit*) o unidad de medición inercial es un dispositivo electrónico que incorpora principalmente acelerómetros, giróscopos y magnetómetros que permiten determinar la orientación exacta de un sistema móvil respecto al eje de referencia de la tierra, esto mediante algoritmos matemáticos que resuelven el problema de la orientación.

Aunque existen IMUs completas que solucionan la determinación de los ángulos, éstas son soluciones de mayor costo. En este tipo de aplicaciones se pueden utilizar sensores integrados de bajo costo, por lo que se utilizaron sensores con la tecnología MEMS (*Micro electro mecanichal sensors*).

Los acelerómetros, giróscopos y magnetómetros integrados en la IMU miden las aceleraciones, velocidades angulares y la fuerza y dirección del campo magnético local respectivamente. Estos suelen estar integrados en el mismo chip, y es común utilizarlos en grupos de tres, uno por cada eje de referencia.

Los datos provistos por la IMU no proveen una estimación de actitud por eso es necesario utilizar filtros de fusión de datos para obtener estas variables.

La IMU utilizada en este trabajo es el IMU 10DOF de DFRobot (10 grados de libertad), es un sensor compacto de bajo coste que integra el acelerómetro ADXL345, el magnetómetro HMC5883L, el giroscopio ITG-3205 y el barómetro BMP085 todo en una placa que contiene además un regulador que admite voltajes de alimentación de 3 a 8v (DFRobot, s.f.).

Figura 2.10: IMU 10DOF de DFRobot

2.3.6 Módulo de comunicaciones

Para establecer la telemetría entre el Quadrotor y una estación terrena, la aeronave debe contener algún módulo de comunicación inalámbrica con la capacidad de comunicarse bidireccionalmente con una estación terrena de control. Esto sirve para poder descargar hacia tierra la información del estado de vuelo, sensores, batería y para poder recibir algunos comandos de vuelo de configuración o señales de referencia.

Para el proyecto se eligió un módulo Xbee de la marca *Digi* presentado en la figura 2.11, que con un tamaño reducido ofrece grandes capacidades para manejar información de forma inalámbrica. El módulo Xbee es de fácil configuración cuando se desea realizar comunicaciones simples de punto a punto, puede ser usado con un mínimo número de conexiones: alimentación (3,3V), *tierra*, *data in* y *data out* que son las líneas de recepción y transmisión de datos. De esta manera se conecta fácilmente con cualquier controlador que incorpore periféricos para comunicación UART.

Algunas de las características del módulo Xbee son (Digi International Inc.):

- ✓ Enlace de datos bidireccional.
- ✓ Frecuencia de operación de 2.4 GHz.
- ✓ Alcance de 120 metros.
- ✓ Velocidad de transmisión hasta 250Kbps.
- ✓ Interfaz digital.
- ✓ Antena integrada

Figura 2.11: Xbee 2mW Wire Antenna - Series 2 (ZigBee Mesh)

2.3.7 Control Remoto

Los UAVs no llevan un piloto a bordo por lo que es necesario manejarlos desde tierra enviando comandos de control o señales de referencia para establecer un rumbo deseado. Para esto se suele utilizar un dispositivo emisor, también conocido como mando radio control, operado por un piloto desde tierra y un receptor a bordo de la aeronave para decodificar las señales de referencia enviadas.

En algunas aeronaves es posible utilizar este sistema de control remoto (emisor y receptor) para controlar directamente los actuadores de manera que el emisor envía los comandos de control necesarios para actuar directamente sobre las dinámicas del UAV. En el caso del Quadrotor resulta complicado controlar directamente los actuadores enviando señales de referencia con el control remoto ya que la dinámica varía rápidamente.

En este caso el emisor sirve de interfaz entre el piloto y el controlador de vuelo programado en el Quadrotor. Su funcionamiento consiste en interpretar los movimientos que ejerce el piloto sobre sus pequeñas palancas como valores de referencia de ángulos para el sistema de control de la aeronave, es decir los grados de inclinación deseados y convertirlos a señales de radio que son enviadas hacia el receptor de la aeronave.

El receptor es el dispositivo que permite decodificar las señales provenientes del emisor y las convierte a señales eléctricas que son interpretadas por el computador de vuelo como señales de referencia establecidas por el piloto en tierra para seguir un rumbo deseado. En la figura 2.12 se muestra el mando y receptor utilizado para el control remoto del Quadrotor en este proyecto.

Figura 2.12: Mando y receptor para control remoto

2.4 FPGA (Field Programmable Gate Array)

Las FPGA son circuitos integrados digitales que incorporan bloques de lógica programable e interconexiones también programables eléctricamente, que hacen que las FPGA sean flexibles para configurarlos de tal manera que pueden convertirse en casi cualquier tipo de circuito o sistema digital (I. Kuon, 2008). Los bloques de lógica programable son los elementos lógicos básicos de las FPGA y están compuestos por LUTs (*LookUp Tables*), que pueden implementar funciones lógicas de varias variables de entrada, multiplexores y biestables tipo D.

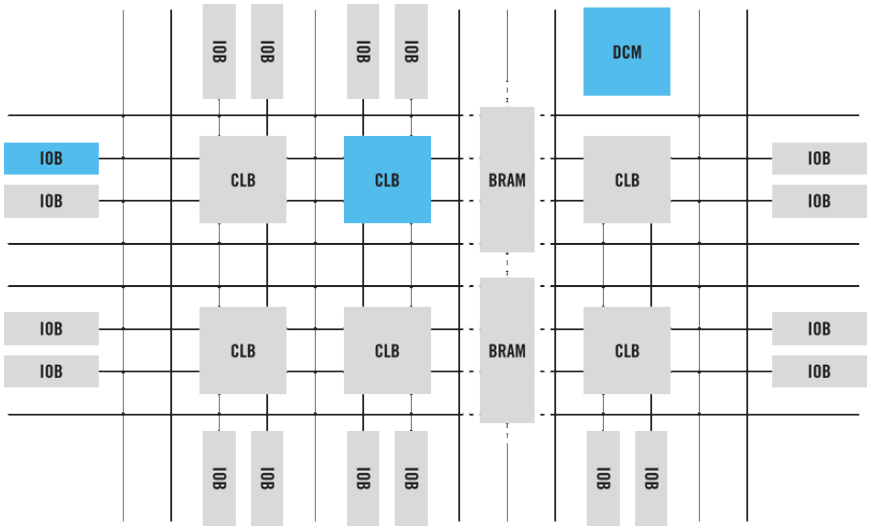


Figura 2.13: FPGA, bloques lógicos programables (CLB de Xilinx)

Otros elementos lógicos pueden estar incluidos en una FPGA, pero estos varían de acuerdo al dispositivo y pueden ser: memorias RAM embebidas, bloques multiplicadores, interfaces de entrada/salida de alta velocidad, DSP (de sus siglas en ingles *Digital Signal Processor*), etc.

Además de los componentes integrados que conforman la FPGA, existe una variedad de bloques IP, también llamados IP core (del inglés *Intellectual Property*), que han provocado que se popularice este tipo de dispositivo. Los IP *cores* son bloques de lógica descritos en algún lenguaje de descripción de hardware o en algún formato propietario y que pueden integrarse de forma sencilla en diseños de SoC (*System on Chip*) para FPGA o ASIC, permitiendo acortar el tiempo de diseño. Estos bloques IP son proporcionados por los fabricantes de FPGAs, por otras empresas dedicadas a la electrónica, e incluso por algunas comunidades de hardware libre, y varían desde interfaces de comunicaciones como UARTs, buses I2C, interfaces *Ethernet*, hasta controladores de memoria, buses de sistema y procesadores (Huerta P. , 2009).

Actualmente las FPGAs están cubriendo una amplia gama de aplicaciones debido a su flexibilidad y alto desempeño. Uno de los campos donde se utilizadan frecuentemente es en el prototipo de diseños en ASIC o como plataforma hardware para verificar la implementación física de nuevos algoritmos, aunque su bajo coste de desarrollo en relación a los ASIC y su corto tiempo de desarrollo hasta la puesta en el mercado hacen que cada vez más se utilicen en productos finales.

Otras aplicaciones donde las FPGAs se utilizan son: aeronáutica y defensa, sistemas de visión artificial, sistemas de visión médica, comunicaciones, electrónica de consumo, etc.

Si bien en sus inicios fue Xilinx en 1985 quien lanzó al mercado la primera FPGA, la XC2064 que tenía una capacidad de 1200 compuertas lógicas equivalentes (UC Berkeley) hoy en día se cuenta con FPGAs con millones de puertas lógicas equivalentes tanto en FPGAs de Xilinx como de Altera que son las marcas que actualmente dominan el mercado.

2.4.1 Tarjeta de desarrollo FPGA DE0-Nano

Se eligió la tarjeta de desarrollo DE0-Nano de Altera como controlador para el Quadrotor debido al tamaño compacto, apto para aplicaciones en robótica y proyectos portátiles. Esta plataforma contiene una FPGA Altera Cyclone IV (con 22,320 elementos lógicos), 32 MB de memoria SDRAM, 2Kb de EEPROM y dispositivo de configuración serial (Terasic, s.f.).

Además esta plataforma presenta otras características importantes (Terasic, 2012):

- ✓ Dos headers de 40 pines cada uno (GPIOs), 72 pines de entrada/salida, dos pines de 5V, dos de 3.3V y cuatro pines conectados a tierra.
- ✓ 8 LEDs verdes.
- ✓ 2 Pushbuttons.
- ✓ 4 DIP switch.
- ✓ Un acelerómetro integrado ADXL345.
- ✓ Convertidor ADC de 8 canales con 12 bits.
- ✓ 50 MHz de reloj.

En la figura 2.14 se observa la tarjeta de desarrollo DE0-Nano de la marca Altera utilizado en este proyecto.

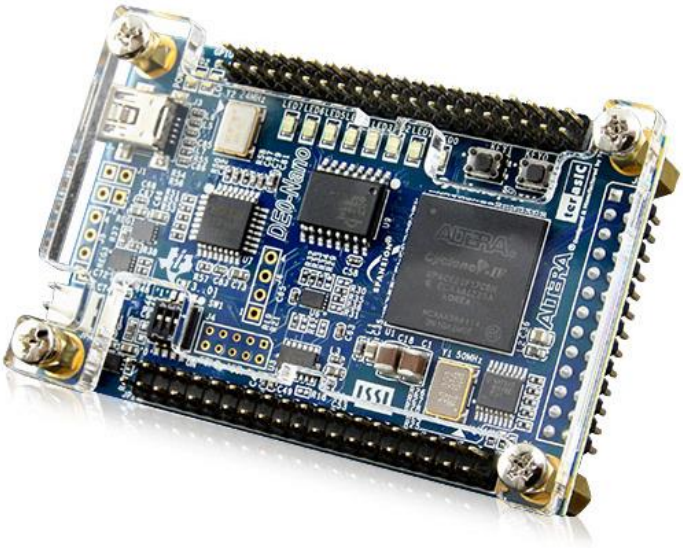


Figura 2.14: DE0-Nano board

Altera provee plataformas de desarrollo con las herramientas necesarias para realizar el diseño, análisis y síntesis del hardware que se desee describir en una FPGA. La plataforma de desarrollo permite realizar el diseño basado en un procesador llamado NIOS II, en el cual se pueden agregar los bloques IP deseados para generar la arquitectura hardware como desarrollar el software que se ejecute en dicho procesador gracias a las herramientas de desarrollo hardware y software que incluye.

NIOS II es un procesador RISC de 32 bits y presenta una arquitectura Harvard. Sus principales características se listan a continuación (Altera, 2015):

- ✓ 32 registros de propósito general de 32 bits.
- ✓ Operaciones de punto flotante de precisión simple.
- ✓ Bus de direcciones de 32 bits.
- ✓ Posibilidad de añadir lógica a la ALU para crear nuevas instrucciones de propósito específico.
- ✓ Memoria Caché de instrucciones y caché de datos opcionales.

CAPÍTULO 3

MODELADO MATEMÁTICO DEL SISTEMA

En este capítulo se describe el modelo matemático del sistema físico mediante ecuaciones que describen su comportamiento dinámico.

Para realizar un diseño de control automático es necesario analizar el modelo matemático del sistema que se debe controlar. Este modelo describe el comportamiento real del sistema Quadrotor.

Se establecen algunas consideraciones para el estudio del modelo del sistema según los estudios realizados en Quadrotores (S. Bouabdallah A. N., 2004b), (Bresciani, 2008).

- ✓ El vehículo aéreo es un cuerpo rígido
- ✓ La estructura del móvil es perfectamente simétrica
- ✓ Las hélices son completamente rígidas
- ✓ El empuje y torque generado por las hélices es siempre en la dirección z.

En la figura 3.1 se presenta el diagrama de fuerzas y torques ejercidos por los rotores que interactúan sobre el cuerpo del móvil, además de los ángulos de orientación que describen el movimiento rotacional. Estos ángulos de rotación son conocidos como alabeo, cabeceo y guiñada que en adelante se nombraran con su notación en ingles llamados *roll* (ϕ), *pitch*

(θ) y *yaw* (ψ) respectivamente, dado que esta notación es la más usada en la literatura. Se han establecido dos marcos de referencia, el marco del cuerpo $\boldsymbol{B}$ que tiene como origen el centro de masa de la aeronave y el marco inercial $\boldsymbol{I}$ que tiene como origen un punto fijo en la tierra. Estos dos marcos de referencia facilitan el estudio del movimiento de la aeronave para obtener las ecuaciones que describen su comportamiento. En el marco del cuerpo se observa la configuración de los actuadores enumerados para una mejor identificación, los cuales generan fuerzas de empuje T_i perpendiculares al plano que forman los brazos del Quadrotor, y torques M_i alrededor del eje z. Las fuerzas y torques son los que comandan el movimiento traslacional y rotacional del vehículo. La suma de fuerzas de los rotores es conocida como la fuerza colectiva o empuje total T_z, esta fuerza actúa sobre el eje z en el mismo sentido que sus componentes y sustenta todo el peso del Quadrotor consiguiendo que la aeronave se mantenga sobre la tierra. Además de esto, las fuerzas de los rotores también influyen en el movimiento rotacional de la aeronave ejerciendo torques alrededor de cada eje de coordenadas del marco del cuerpo que estarán representados por el vector $\tau = [\tau_\phi \quad \tau_\theta \quad \tau_\psi]^T$.

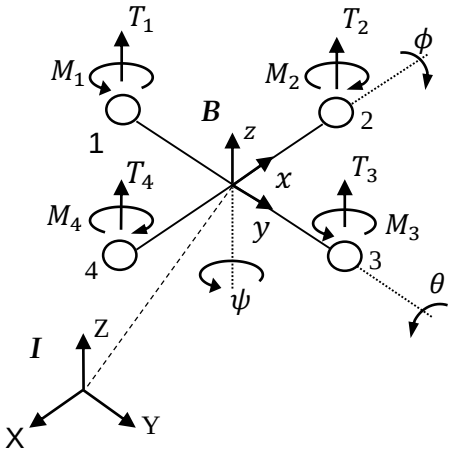


Figura 3.1: Diagrama de fuerzas y momentos

Para formular el modelo del sistema Quadrotor se ha dividido en bloques que se estudian con detalle, identificando las ecuaciones que representan cada etapa. El sistema Quadrotor se observa en la figura 3.2 donde se muestran los bloques con sus respectivas señales y/o fuerzas de entrada y salida.

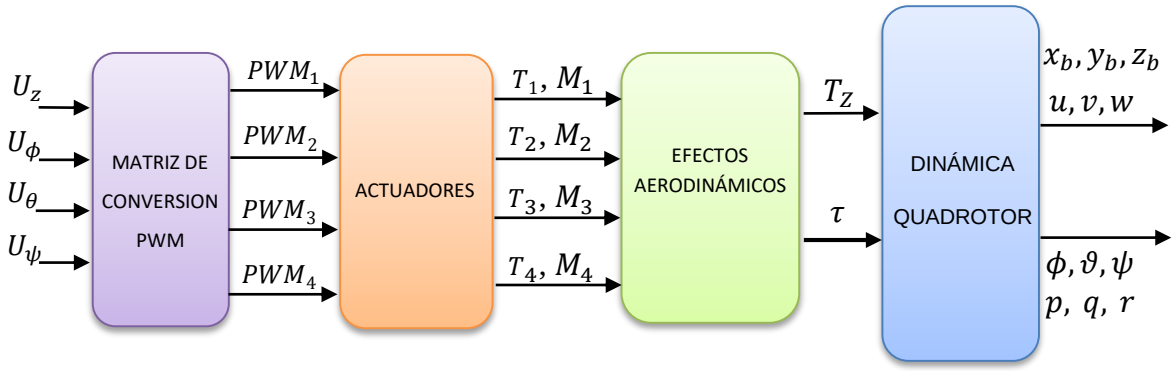


Figura 3.2: Diagrama de bloques del sistema Quadrotor

El sistema Quadrotor tiene como variables de entradas las señales de control U_z, U_ϕ, U_θ y U_ψ que influyen directamente en la actitud y altura del vehículo. Estas señales son calculadas por el controlador según los parámetros deseados, es decir según la dinámica que se requiere lograr en el Quadrotor como ángulos de inclinación, posición y velocidades. Las señales de control se combinan en la matriz de conversión PWM para calcular el suministro de energía que energice a cada actuador. Luego el bloque actuadores recibe esta energía modulada mediante señales PWM para generar fuerzas y torques sobre el cuerpo de la aeronave con el fin de lograr las dinámicas deseadas. El bloque de efectos aerodinámicos tiene como variables de entrada las fuerzas y momentos de los actuadores y genera como variable de salida la fuerza total de los actuadores T_Z y el vector de torques $\tau = [\tau_\phi \quad \tau_\theta \quad \tau_\psi]^T$ que representa a los momentos ejercidos sobre cada eje de coordenadas del marco de referencia del cuerpo en el Quadrotor. Por último el bloque de dinámica muestra como esta fuerza y torques influyen sobre el móvil para cambiar su posición y actitud en el tiempo.

En las secciones 3.1, 3.2, 3.3, 3.4 y 3.5 se detallan el desarrollo de la formulación matemática de cada bloque y los ensayos realizados para identificar los parámetros propios del sistema Quadrotor.

3.1 Dinámica del Quadrotor

En esta sección se presenta el desarrollo del modelo dinámico del Quadrotor basado en leyes físicas que describen el comportamiento de la aeronave. El desarrollo se realizó mediante la formulación Newton-Euler. En la figura 3.1 se muestran los marcos de referencia establecidos para representar del movimiento del Quadrotor, estos marcos de referencia están relacionados mediante una matriz de orientación descrita por rotaciones sucesivas usando los ángulos de Euler. Esta matriz permite representar la posición y orientación del móvil en el marco inercial.

3.1.1 Orientación del Quadrotor

La matriz de rotación del marco del cuerpo respecto del marco inercial está dada por las rotaciones respecto a los ejes del plano de coordenadas *XYZ,* los ángulos roll, pitch y yaw. Esta matriz es comúnmente conocida como Matriz Coseno Directa y se representa de la siguiente manera:

$$R_b^i = \begin{bmatrix} cos\psi cos\theta & cos\psi sen\theta sen\phi - sen\psi cos\phi & cos\psi sen\theta cos\phi + sen\psi sen\phi \\ sen\psi cos\theta & sen\psi sen\theta sen\phi + cos\psi cos\phi & sen\psi sen\theta cos\phi - cos\psi sen\phi \\ -sen\theta & cos\theta sen\phi & cos\theta cos\phi \end{bmatrix} \quad (3.1)$$

3.1.2 Cinemática del Quadrotor

La relación de la velocidad angular del Quadrotor con respecto a los ejes del plano coordenado expresado en el marco del cuerpo $\omega = [p \quad q \quad r]^T$ y la variación en el tiempo de los ángulos de Euler $\dot{\eta} = [\dot{\phi} \quad \dot{\theta} \quad \dot{\psi}]^T$ se obtiene utilizando la matriz Jacobiana.

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & sen\phi tan\theta & cos\phi tan\theta \\ 0 & cos\phi & -sen\phi \\ 0 & sen\phi sec\theta & cos\phi sec\theta \end{bmatrix} . \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{3.2}$$

Y usando la matriz Jacobiana inversa se obtiene las velocidades angulares del Quadrotor.

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -sen\theta \\ 0 & cos\phi & cos\theta sen\phi \\ 0 & -sen\phi & cos\theta cos\phi \end{bmatrix} . \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{3.3}$$

3.1.3 Formulación Dinámica de Newton-Euler

El modelo de la dinámica traslacional y rotacional se expresa mediante las ecuaciones Newton-Euler que se muestran a continuación:

$$\begin{bmatrix} mI_{3x3} & 0 \\ 0 & J \end{bmatrix} \begin{bmatrix} \dot{v} \\ \dot{\omega} \end{bmatrix} + \begin{bmatrix} \omega \times mv \\ \omega \times J\omega \end{bmatrix} = \begin{bmatrix} F \\ \tau \end{bmatrix} \tag{3.4}$$

Dónde:

F: Vector de fuerzas externas expresadas en el cuerpo.

τ: Vector de torques aerodinámicos

I_{3x3} : Matriz identidad de 3x3.

v : Vector de velocidades traslacionales expresadas en el cuerpo

J : Matriz de inercias.

m: Masa total del Quadrotor.

3.1.3.1 Movimiento traslacional

De la formulación dinámica Newton-Euler se obtienen las ecuaciones de movimiento traslacional. Reemplazando las primeras tres ecuaciones por sus componentes vectoriales y expresando el vector de velocidades lineales como $v = [u \quad v \quad w]^T$ se obtiene:

$$F = m\left(\begin{bmatrix}\dot{u}\\ \dot{v}\\ \dot{w}\end{bmatrix} + \left\{\begin{bmatrix}p\\ q\\ r\end{bmatrix} x \begin{bmatrix}u\\ v\\ w\end{bmatrix}\right\}\right) \tag{3.5}$$

Resolviendo las ecuaciones vectoriales (3.5) se obtienen las tres primeras ecuaciones del cuerpo rígido:

$$\begin{aligned}\dot{u} &= vw - qr + gsen\theta\\ \dot{v} &= pw - ur + gcos\theta sen\phi\\ \dot{w} &= qu - pv - gcos\theta cos\phi + \frac{T_z}{m}\end{aligned} \tag{3.6}$$

3.1.3.2 Movimiento rotacional

Teniendo en cuenta que se considera al Quadrotor como perfectamente simétrico, la matriz de inercia se expresa de la siguiente manera:

$$J = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{3.7}$$

Reemplazando la matriz de inercia (3.7) y los componentes vectoriales en la ecuación de movimiento rotacional del cuerpo rígido se tiene la expresión siguiente:

$$\begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} + \left\{ \begin{bmatrix} p \\ q \\ r \end{bmatrix} x \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \right\} \tag{3.8}$$

Resolviendo el sistema de ecuaciones (3.8) se obtienen las tres ecuaciones de movimiento rotacional del cuerpo rígido:

$$\begin{aligned} \tau_\phi &= I_{xx}\dot{p} + qr(I_{zz} - I_{yy}) \\ \tau_\theta &= I_{yy}\dot{q} + pr(I_{xx} - I_{zz}) \\ \tau_\psi &= I_{zz}\dot{r} + pq(I_{yy} - I_{xx}) \end{aligned} \tag{3.9}$$

De esta manera se obtienen las ecuaciones que rigen el comportamiento rotacional del móvil que se analiza para el desarrollo de los controladores de actitud.

3.1.4 Desacoplo de dinámicas del Quadrotor

Para desarrollar un diseño de múltiples controladores PID se ha elegido el punto de operación al estado de *hovering* para realizar simplificaciones que describan de manera aproximada la dinámica del Quadrotor y facilite el diseño de los controladores. Cuando el Quadrotor se encuentra en *hovering* se asume lo siguiente:

- ✓ La suma de las fuerzas generadas por los actuadores debe ser equivalente al peso del móvil de 11.7N (dado que tiene aproximadamente 1.2 Kg de masa) para que este pueda mantenerse en *hovering* a una altura constante. Cada actuador debe generar un empuje de 2.9N aproximadamente para que se cumpla esta condición.
- ✓ Cuando el Quadrotor está en *hovering* se considera que realiza pequeños movimientos traslacionales por lo que los ángulos de Euler y las velocidades angulares de cada eje son cercanos a cero, esto hace que las multiplicaciones de las velocidades angulares (los efectos giroscópicos) sean despreciables y no se consideren en el modelo obtenido en las ecuaciones (3.9).

Teniendo en cuenta estas consideraciones se simplifica el modelo (3.9) obteniendo las ecuaciones siguientes:

$$\begin{aligned} \tau_{\phi} &= I_{xx}\dot{p} \\ \tau_{\theta} &= I_{yy}\dot{q} \\ \tau_{\psi} &= I_{zz}\dot{r} \end{aligned} \tag{3.10}$$

De esta manera se obtiene un modelo más simple que no presenta acoplamiento entre ejes y hace posible la realización de un diseño de control para cada eje de manera independiente como se ve en la sección 3.5. Para expresar el comportamiento del sistema como funciones de transferencia se convierten las ecuaciones diferenciales al dominio de Laplace y se expresa de la siguiente manera:

$$\frac{p_{(s)}}{\tau_{\phi(s)}} = \frac{1}{s.I_{xx}} \tag{3.11}$$

$$\frac{q_{(s)}}{\tau_{\theta(s)}} = \frac{1}{s.I_{yy}} \tag{3.12}$$

$$\frac{r_{(s)}}{\tau_{\psi(s)}} = \frac{1}{s.I_{zz}} \tag{3.13}$$

Los momentos de inercia se han determinado realizando un diseño del móvil en un programa de dibujo Solidwork, el cual permite obtener los momentos de inercia del modelo dibujado. Los valores obtenidos se muestran en la tabla 3.1.

Tabla 3.1: Valores de momentos de inercia

Momento de inercia	$\boldsymbol{\frac{Kg}{m^2}}$
$\boldsymbol{I_{xx}}$	0.009609
$\boldsymbol{I_{yy}}$	0.009992
$\boldsymbol{I_{zz}}$	0.018863

3.2 Efectos Aerodinámicos

Como se ha visto en el diagrama de bloques del sistema Quadrotor (figura 3.1), la interacción de las fuerzas y torques ejercidos por los actuadores producen los torques aerodinámicos sobre el cuerpo del móvil. Estos torques son descritos a continuación:

- La diferencia de fuerzas de empuje T_2 y T_4 genera el torque τ_ϕ alrededor del eje x del marco del cuerpo.
- La diferencia de fuerzas de empuje T_1 y T_3 es el torque τ_θ alrededor del eje y del marco del cuerpo.

- La diferencia del par de torques M_1, M_3 y M_2, M_4 es el torque τ_ψ alrededor del eje z del marco del cuerpo.

El vector de torques $\tau = [\tau_\phi \quad \tau_\theta \quad \tau_\psi]^T$ se expresa como un sistema de ecuaciones de la siguiente manera:

$$\begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} l\,(T_2 - T_4) \\ l\,(T_3 - T_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix} \tag{3.14}$$

Donde l equivale a la distancia desde el centro de masa del Quadrotor hasta el eje de la hélice.

El empuje total o fuerza colectiva T_z , se compone por la suma de las fuerzas de los cuatro propulsores y actúa sobre el eje z del Quadrotor.

$$T_Z = \sum_{i=1}^{4} T_i \tag{3.15}$$

3.3 Actuadores

En el sistema Quadrotor el motor *BrushLess*, el ESC y la hélice en conjunto generan las fuerzas que gobiernan el movimiento del vehículo aéreo y conforman lo que se conoce como el actuador. El diagrama de funcionamiento de los actuadores y las señales que intervienen en la operación de este sistema se puede observar en la figura 3.3. Se muestra que la señal de entrada es de tipo PWM y las señales de salida generadas es una fuerza de empuje y un torque.

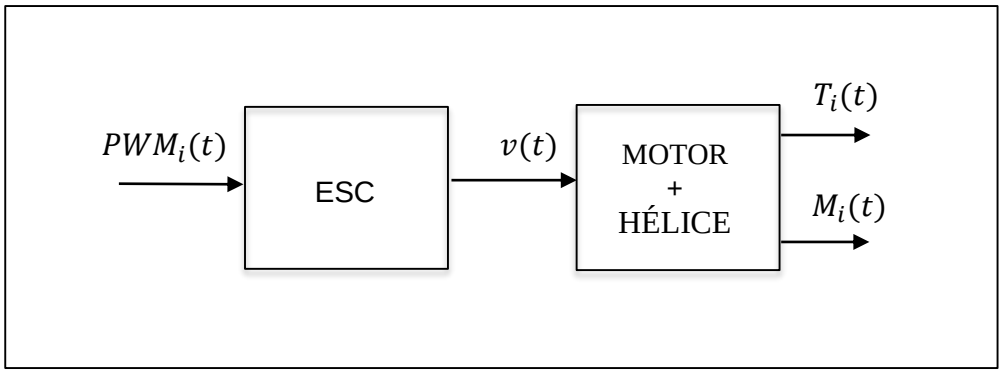


Figura 3.3: Diagrama de bloques de actuadores

El sistema Quadrotor consta de cuatro actuadores que son comandados por el controlador mediante señales PWM que suministran la energía a los ESC con un voltaje variable modulado por el ciclo de trabajo. Los ESC a su vez varían el voltaje trifásico $v(t)$ que entregan al motor para que gire a una velocidad y torque definidos que transmiten a las hélices generando así las fuerzas y torques de los actuadores.

Con el fin de modelar el sistema Quadrotor fue necesario identificar la dinámica de los actuadores para el diseño del control. Es conocido que los actuadores pueden ser modelados con la ecuación siguiente (Bouabdallah, 2011).

$$T_i = b\Omega_i{}^2 \tag{3.16}$$

$$M_i = d\Omega_i{}^2 \tag{3.17}$$

Donde:

b : Constante de empuje

d : Constante de arrastre

Ω : Velocidad angular de los propulsores $\left[\frac{deg^2}{s^2}\right]$

De las ecuaciones (3.16) y (3.17) se deduce que las fuerzas y torques son proporcionales a la potencia al cuadrado de las velocidades angulares de los rotores, sin embargo en (Kharsansky, 2013) se realizó ensayos con los actuadores de un Quadrotor donde se logró identificar un modelo lineal de los actuadores que tiene como entrada la señal de PWM y como salida las fuerzas y torques aerodinámicos. Estos modelos son de mayor utilidad para el diseño de control porque se pueden expresar como funciones de transferencia con una variable de entrada que es manipulada directamente como el PWM, a diferencia de la velocidad angular que no puede ser manipulada directamente.

La frecuencia de la señal PWM es de 400Hz o es lo mismo decir que tiene un periodo de 2.5ms que está dentro del rango de operación de los ESCs de DJI (funcionan de 50 a 450Hz), se modula esta señal variando el ancho de pulso en voltaje alto lo que es conocido como el ciclo de trabajo y tiene una resolución de 14 bits, es decir el ciclo de trabajo puede estar entre 0 y 16384 valores discretos.

El motor BLDC presenta una zona muerta y una zona de saturación, en consecuencia la señal PWM logra poner en funcionamiento al rotor en un corto rango de ciclo de trabajo, por eso manipular todo el rango de la señal PWM sería inútil. Por tal razón se decidió utilizar la señal PWM variando el ciclo de trabajo desde 5600 a 12400, en sus valores discretos, lo que se denomina el rango de operación del actuador. Además se expresa el ciclo de trabajo en porcentaje seleccionando a 5600 como el 0% y a 12400 como el 100%, esto se escogió por conveniencia para expresar la señal PWM como una variable porcentual y hacer coincidir la zona muerta del rotor cuando la entrada de la señal PWM presenta 13% de ciclo de operacion y que sature cuando se encuentra al 100%. Por lo que se hará referencia a la señal PWM como una señal porcentual.

Se mencionó anteriormente que en el punto de operación los actuadores generan la misma fuerza que el peso del vehículo, es decir 11.7N, y cada actuador debe aportar la cuarta parte de esta fuerza o sea 2.9N. Se comprobó que los actuadores generan esa cantidad

de empuje cuando están funcionando alrededor del 65% de señal PWM. Teniendo esto en consideración se realizaron experiencias en un banco de pruebas para identificar la respuesta de los actuadores al introducirle señales PWM. La respuesta en el tiempo consta de dos componentes: respuesta transitoria y respuesta en estado estacionario.

Con el fin de analizar la respuesta estacionaria generada por los propulsores se realizaron experimentos en dos bancos de pruebas, uno para medir el empuje y otro para medir la fuerza de arrastre. En la figura 3.4 se muestran los bancos de pruebas para los actuadores, en el lado de la izquierda se observa el banco utilizado para medir la fuerza de empuje generado por los actuadores y en el lado de la derecha el banco utilizado para medir la fuerza de arrastre. La medición se realiza con una balanza digital tomando datos y anotándolos en una tabla de valores.

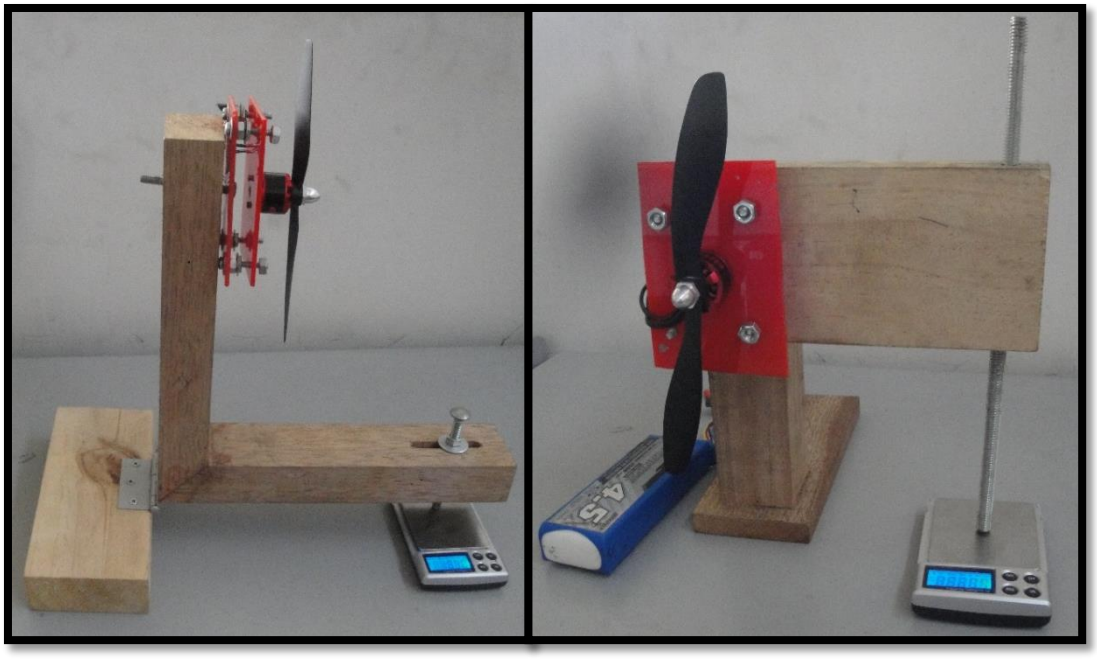


Figura 3.4: Banco de pruebas de actuadores

3.3.1 Respuesta estacionaria

Para obtener la respuesta estacionaria del empuje se fijó el actuador en el extremo superior de la maqueta de manera que la fuerza de empuje generada se transmita al extremo que está sobre la balanza como se observa en el lado de la izquierda de la figura 3.4.

Se registraron datos del empuje incrementando la señal PWM desde 0% hasta 100% para poder construir una gráfica que relacione estas variables en todo el rango de operación. Este ensayo se realiza para identificar la constante de empuje que relaciona la señal PWM con el empuje como se ve en la ecuación siguiente:

$$T_i = K_t . PWM_i \tag{3.18}$$

En la figura 3.5 se observa la gráfica que relaciona las variables de empuje y la señal PWM, además la regresión lineal que aproxima a la respuesta obtenida mediante una función lineal, donde la constante de empuje es la pendiente de la recta obtenida con la regresión. Se muestra que conforme el valor de la señal PWM aumenta y se aleja del punto de operación de 65%, la respuesta obtenida con la regresión lineal se distancia de la respuesta real revelando que la identificación del actuador es aproximado cuando está cerca del punto de operación y mientras más se aleje pierde precisión. De igual manera se realizó los mismos ensayos con los demás actuadores dando como resultado los valores que se muestran en la tabla 3.2.

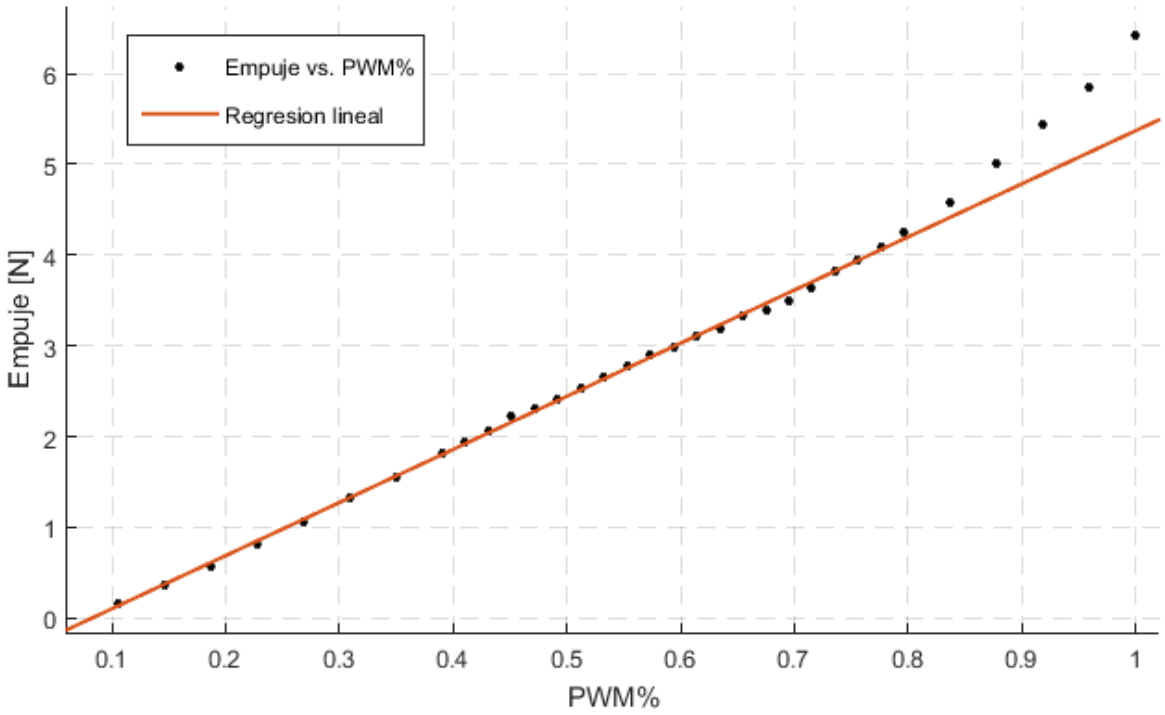


Figura 3.5: Regresión lineal de PWM% vs Empuje

Tabla 3.2: Constantes de empuje de actuadores

ACTUADOR	$K_T \left[\frac{N}{\%}\right]$
ACTUADOR 1	6.446
ACTUADOR 2	6.747
ACTUADOR 3	6.636
ACTUADOR 4	6.652

De la misma manera se realizaron los experimentos para construir la gráfica que relaciona la señal PWM vs torque con el banco de pruebas presentado en el lado derecho de la figura 3.4. La ecuación que relaciona estas variables se expresa de la forma siguiente:

$$M_i = K_M . PWM_i \tag{3.19}$$

En la figura 3.6 se muestra la respuesta obtenida al tomar datos en todo el rango de operación de los PWM:

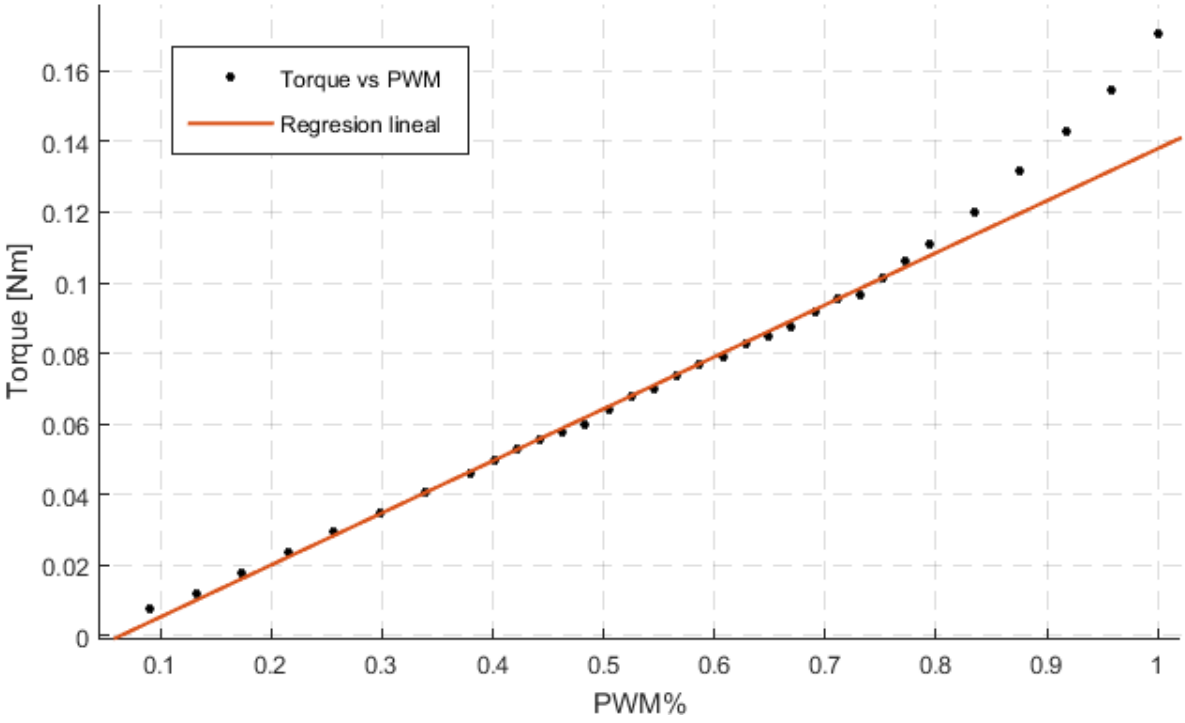


Figura 3.6: Regresión lineal de PWM% vs Momento

De manera similar a la regresión lineal realizada con el empuje, la constante de arrastre se identifica aproximando la respuesta no lineal a una función lineal donde la constante toma el valor de la pendiente de la recta. Los valores obtenidos de los cuatro actuadores se muestran en la tabla 3.3.

Tabla 3.3: Constantes de arrastre de los actuadores

ACTUADOR	$K_M \left[\frac{Nm}{\%}\right]$
ACTUADOR 1	0.150
ACTUADOR 2	0.154
ACTUADOR 3	0.147
ACTUADOR 4	0.152

Dado que es conveniente formular un modelo único de los actuadores se promediaron los valores de las constantes para obtener un valor promedio de empuje y torque para los cuatro actuadores. La tabla 3.4 muestra los valores resultantes:

Tabla 3.4: Valores promedio de constantes de actuadores

Constante de actuadores	Valor
K_T	6.62
K_M	0.150

3.3.2 Respuesta transitoria

También se realizaron experimentaciones para identificar la respuesta transitoria de los rotores. En esta experiencia interesa más la forma cómo evoluciona la respuesta en el tiempo hasta que logra una estabilidad, por ello se decidió utilizar la velocidad angular del rotor como variable de salida por la facilidad con la que se puede medir esta variable para obtener una respuesta temporal a diferencia del empuje. Se utilizaron variables de entrada de tipo escalón con amplitudes alrededor del punto de operación (65% de PWM) para

excitar al actuador y se midió la respuesta en el tiempo de la velocidad angular implementando un tacómetro casero con el sensor óptico cny70 como mostrado en la figura 3.7, configurado para que discrimine colores oscuros de claros se cubrió de blanco la armadura del rotor con una delgada franja negra para que el sensor óptico pueda diferenciar los colores enviando un pulso cuadrado cada vez que el rotor gire una vuelta completa y el sensor pase por encima de la franja negra. La FPGA captura los flancos de subidas de cada pulso enviado y midiendo el tiempo entre cada flanco se obtiene el periodo de una vuelta, usando la ecuación (3.20) se obtiene la velocidad angular.

$$\Omega = 2\pi f \tag{3.20}$$

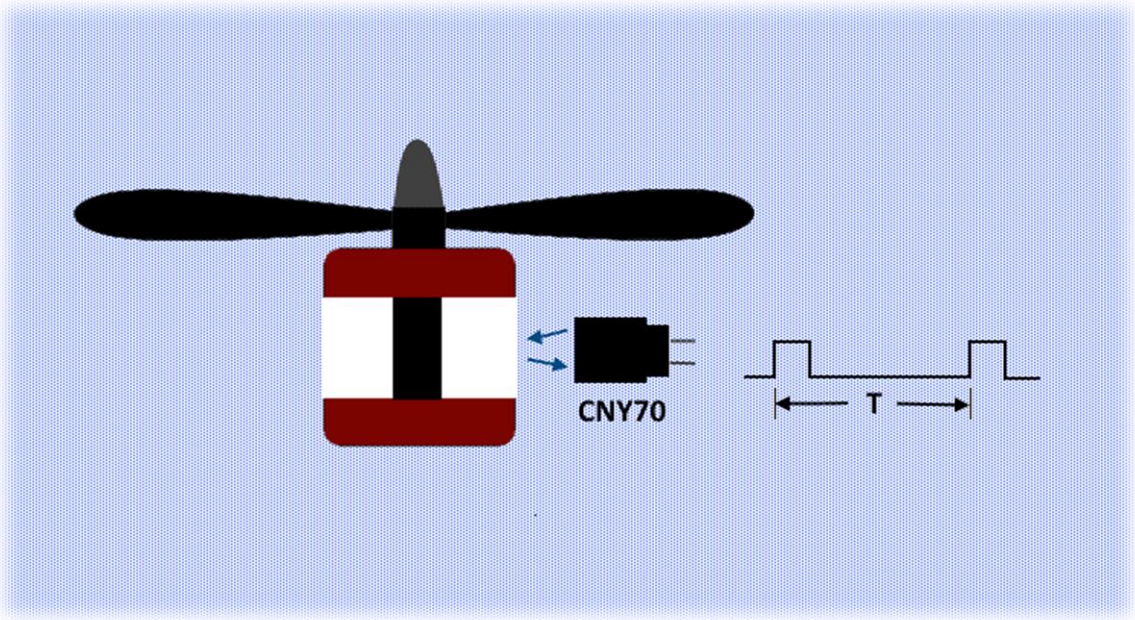


Figura 3.7: Tacómetro basado en el sensor CNY70

Dónde:

Ω : Velocidad angular del rotor $\left[\frac{deg}{s}\right]$

f: Frecuencia de giro $[Hz]$

El comportamiento transitorio del rotor se observa en la figura 3.8 donde se muestra la respuesta ante variables de entrada del tipo escalón con amplitudes alrededor del punto de operación.

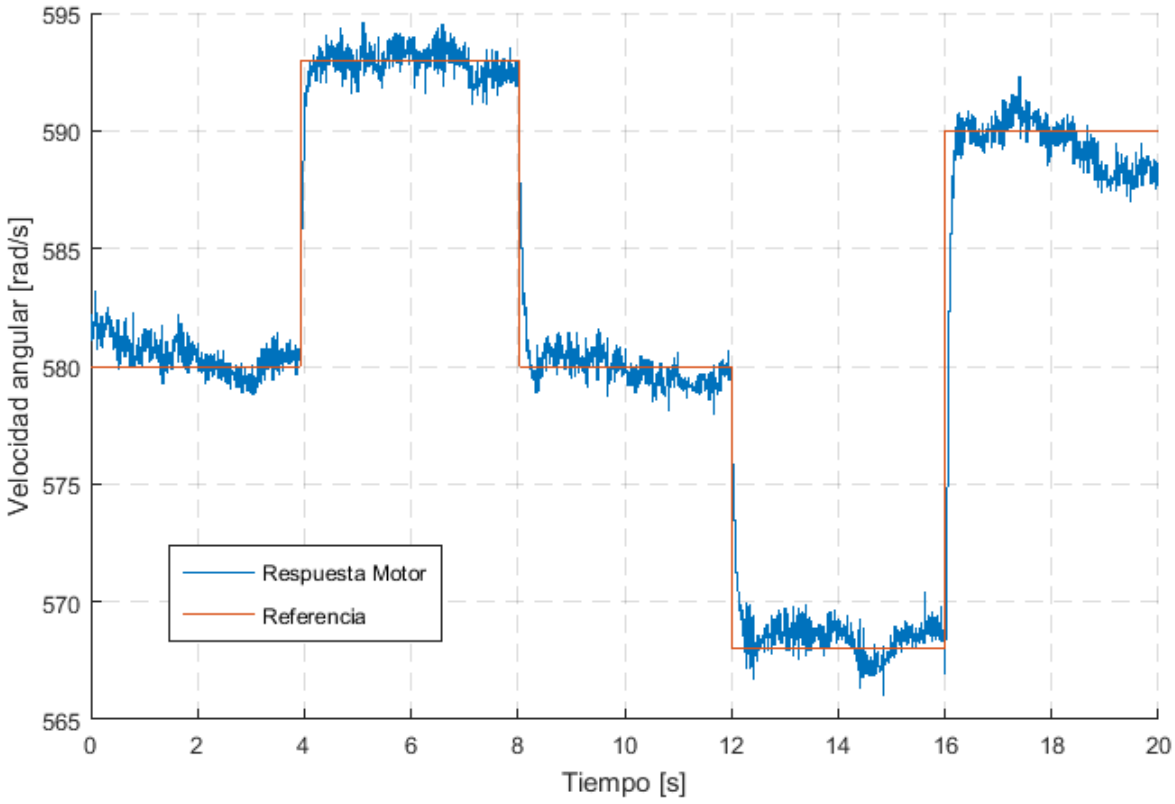


Figura 3.8: Respuesta transitoria de un rotor

Mediante el *toolbox* del software Matlab para identificar sistemas (MathWorks Inc., s.f.), se logra obtener los parámetros del sistema obteniéndose el modelo matemático correspondiente a una ecuación diferencial de primer orden donde se determina el tiempo de crecimiento $\mathcal{T} = 0.075\ seg$.

De esta manera mediante ambos experimentos (la respuesta estacionaria y transitoria) se obtiene el modelo matemático completo de los actuadores para el empuje y el momento de arrastre.

$$\frac{T_{i(s)}}{PWM_{i(s)}} = \frac{K_t}{\mathcal{T}.s + 1} \tag{3.21}$$

$$\frac{M_{i(s)}}{PWM_{i(s)}} = \frac{K_M}{\mathcal{T}.s + 1} \tag{3.22}$$

3.4 Matriz de conversión PWM

En el diagrama de bloques presentado en la figura 3.2 se observa que hay cuatro variables de entradas de control que no son directamente las señales de control de los actuadores, sino que estas señales provienen de los controladores de cada uno de los cuatro grados de libertad (altura, *roll*, *pitch* y *yaw*). El bloque matriz de conversión PWM combina estas variables de entradas para enviar órdenes a cada actuador, es decir cada actuador recibe una acción de control de varios controladores. Se presenta las ecuaciones en forma de matriz.

$$\begin{bmatrix} PWM_1 \\ PWM_2 \\ PWM_3 \\ PWM_4 \end{bmatrix} = M_C . \begin{bmatrix} U_Z \\ U_\phi \\ U_\theta \\ U_\psi \end{bmatrix} \tag{3.23}$$

En la matriz de torques aerodinámicos (3.14), se observa la acción que cada actuador ejerce sobre los ángulos de orientación (*roll*, *pitch* y *yaw*), además los cuatro actuadores contribuyen a la fuerza colectiva que controla la altura del móvil, por tanto la matriz de control debe presentar una forma que concuerde con este enfoque. Siguiendo este planteamiento se presenta la matriz de conversión de PWM:

$$\begin{bmatrix} PWM_1 \\ PWM_2 \\ PWM_3 \\ PWM_4 \end{bmatrix} = \begin{bmatrix} K_z & 0 & -K_\theta & K_\psi \\ K_z & K_\phi & 0 & -K_\psi \\ K_z & 0 & K_\theta & K_\psi \\ K_z & -K_\phi & 0 & -K_\psi \end{bmatrix} \cdot \begin{bmatrix} U_Z \\ U_\phi \\ U_\theta \\ U_\psi \end{bmatrix} \quad (3.24)$$

Donde los valores K_z, K_ϕ, K_θ y K_ψ son magnitudes de las ganancias que se ajustaron para realizar la combinación de las señales de control. Estas ganancias forman parte del diseño de control y se entiende como el peso que determina la sensibilidad o agresividad de cada uno de los comandos de control (Kharsansky, 2013)**.** Por conveniencia se eligió que la señal de control U_Z varíe entre 0.0 y 1.0 para el control de la altura y las señales de control angular varíen entre -1.0 y 1.0.

Las ganancias además de mezclar de las señales de entrada, determinan el porcentaje de acción de control que se transmite a los actuadores. En la tabla 3.5 se muestran los valores de las ganancias que se eligieron.

Tabla 3.5: Ganancias de matriz de conversión PWM

Ganancia	**Valor**
K_z	80%
K_ϕ	40%
K_θ	40%
K_ψ	30%

El criterio para el cálculo de las ganancias fue asignarle la mayor prioridad a la ganancia de la altura K_z dado que el Quadrotor requiere mayor potencia de los actuadores para lograr mantenerse en vuelo que para realizar algún cambio en su orientación además en las

ganancias de acción angular se estableció mayor porcentaje a las entradas de los ángulos *roll* y *pitch* dado que el control de estos ángulos debe ser rápido para poder estabilizar el vuelo del móvil y menor porcentaje a la entrada de control de *yaw* ya que no es de vital importancia que sea tan agresiva.

3.5 Desacoplamiento de subsistemas

Para realizar el diseño de control lineal con controladores PID resulta conveniente dividir el sistema Quadrotor en tres partes que representan a cada eje de rotación *roll, pitch* y *yaw* que en adelante se llamaran subsistemas, de esta manera se diseñan los controladores para cada subsistema de manera independiente.

El sistema Quadrotor representado por diagramas de bloques permite el desacoplamiento en subsistemas representados por funciones de transferencia que se presentan a continuación como los modelos teóricos de cada subsistema. Además se presenta otra alternativa a la separación de subsistemas llamada modelo experimental.

3.5.1 Modelos teóricos de subsistemas del Quadrotor

Esto modelos se basan en el desarrollo del sistema Quadrotor por diagrama de bloques que se ha presentado previamente. Para mostrar el procedimiento se desarrolla el desacoplamiento del subsistema *yaw*, los demás subsistemas siguen un procedimiento similar.

De las ecuaciones (3.14) y (3.22) se tiene que:

$$\tau_{\psi}(s) = \frac{K_M.(PWM_1(s) - PWM_2(s) + PWM_3(s) - PWM_4(s))}{(\mathcal{T}.s + 1)} \qquad (3.25)$$

Resolviendo el sistema de ecuaciones (3.24) se obtiene:

$$\begin{aligned} PWM_1 &= U_z.K_z - K_\theta.U_\theta + K_\psi.U_\psi \\ PWM_2 &= U_z.K_z + K_\phi.U_\phi - K_\psi.U_\psi \\ PWM_3 &= U_z.K_z + K_\theta.U_\theta + K_\psi.U_\psi \\ PWM_4 &= U_z.K_z - K_\phi.U_\phi - K_\psi.U_\psi \end{aligned} \tag{3.26}$$

Transformando estas ecuaciones al dominio de Laplace y reemplazando en la ecuación (3.25), para $U_\psi \neq 0$ se obtiene:

$$\frac{\tau_\psi(s)}{U_\psi(s)} = \frac{4.K_M.K_\psi}{\mathcal{T}.s+1} \tag{3.27}$$

Uniendo esta expresión con el modelo desacoplado (3.13), se consigue separar el subsistema yaw expresándolo con la función de transferencia siguiente:

$$\frac{r(s)}{U_\psi(s)} = \frac{4.K_M.K_\psi}{I_{zz}.s(\mathcal{T}.s+1)} \tag{3.28}$$

Siguiendo un procedimiento similar se obtienen las funciones de transferencia para los subsistemas *roll* y *pitch*:

$$\frac{p(s)}{U_\phi(s)} = \frac{2.l.K_t.K_\phi}{I_{xx}.s(\mathcal{T}.s+1)} \tag{3.29}$$

$$\frac{q(s)}{U_\theta(s)} = \frac{2.l.K_t.K_\theta}{I_{yy}.s(\mathcal{T}.s+1)} \tag{3.30}$$

3.5.2 Modelo experimental del sistema

El desarrollo de los modelos teóricos permitió conocer la forma de las funciones de transferencia de cada subsistema, es decir el número de polos y ceros, así como las variables de salida y de entrada. Teniendo esta información previa se presenta otra alternativa para el desacoplamiento en subsistemas de forma experimental. Esto se logra con el método de identificación de sistemas, teniendo previo conocimiento del modelo, además la posibilidad de medir la variable de salida del subsistema y manipulando directamente la variable de entrada es posible identificar los subsistemas de forma experimental haciendo uso del *toolbox* del software Matlab para identificar sistemas (Nonami, Kendoul, Suzuki, Wang, & Nakazawa, 2010), (Stanculeanu & Borangiu, 2011), (Jaramillo & Gómez, 2013). Para este ensayo se utiliza el banco de pruebas mostrado en la figura 3.9, atando cuerdas a los extremos del Quadrotor dejándolo suspendido en el aire con el fin de restringir el movimiento del vehículo a un solo eje de rotación. Al lado izquierdo se observa que el Quadrotor es sujetado con cuerdas hacia el extremo superior e inferior del banco de pruebas de manera que solo pueda rotar en el eje z, esta configuración es válida para la identificación del subsistema *yaw*. Al lado derecho de la figura 3.9 se muestra la configuración para la identificación del subsistema *pitch* atando las cuerdas a dos brazos del Quadrotor con los soportes laterales del banco de pruebas. La configuración para el subsistema *roll* es similar al subsistema *pitch*, atando cuerdas a los dos brazos restantes con los soportes laterales. Este banco de pruebas también se utilizara para probar los algoritmos de control como se ve en el capítulo 5.

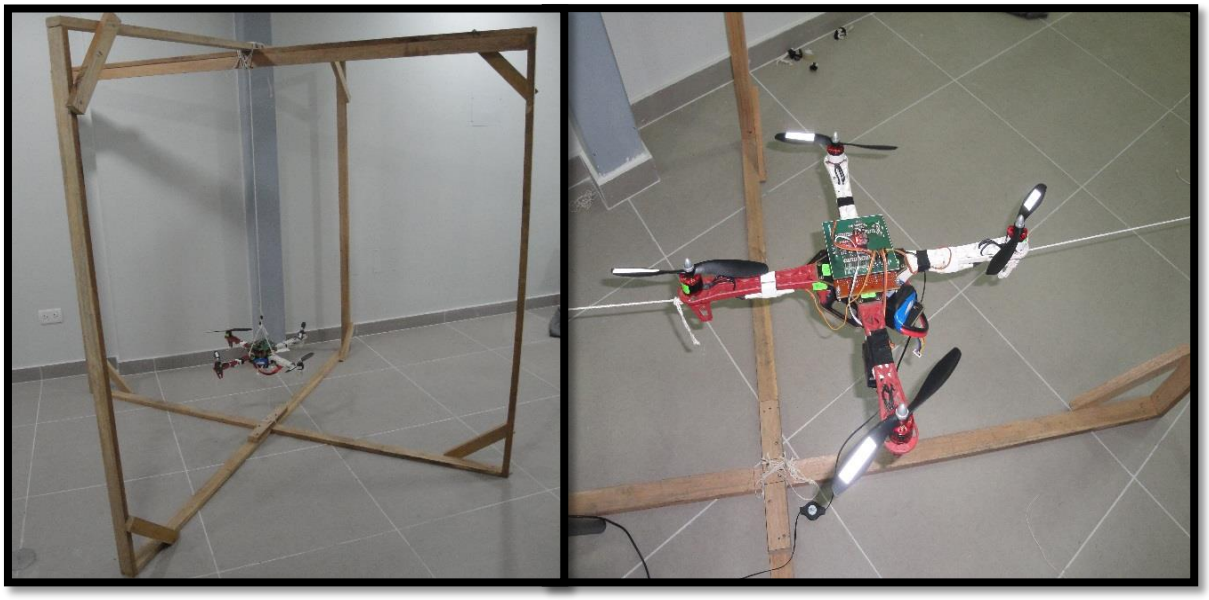


Figura 3.9: Banco de pruebas

Se ha realizado ensayos para cada eje de rotación seleccionando cada subsistema como una “caja negra”. El modelo de “caja negra” tiene como variable de entrada la velocidad angular deseada y como variable de salida la velocidad angular del móvil, ambas señales se graban en la memoria para luego analizarlas en el software Matlab. Se registraron datos cuando el Quadrotor está operando con los actuadores a una potencia alrededor del punto de operación, es decir cada actuador está funcionando alrededor del 65% de ciclo de operación de PWM. Debido a que el sistema Quadrotor es inherentemente inestable es necesario implementar un controlador proporcional que permita realizar una adquisición más coherente entre las señales de entrada y las señales de salida. El esquema de la “caja negra” se observa en la figura 3.10.

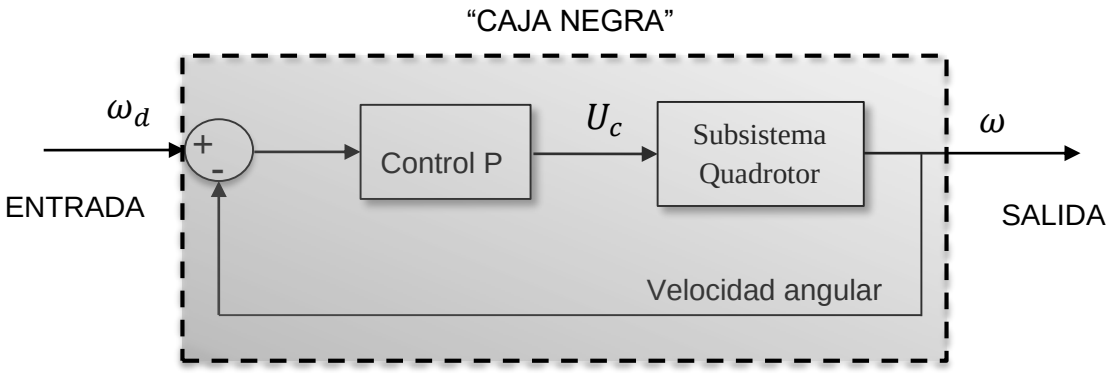


Figura 3.10: Modelo de caja negra para un subsistema

Las señales de entrada se generaron con el mando de radio control simulando señales escalones y senoidales a diferentes frecuencias mientras que la señal de respuesta de velocidad angular se adquirió a un tiempo de muestreo de 10 ms. En la sección anterior se obtuvo los modelos dinámicos teóricos basados en los parámetros del sistema Quadrotor (3.28), (3.29) y (3.30). Dado que solo se necesita conocer la forma de la función de transferencia, es decir el número de polos y ceros, y los 3 modelos dinámicos tienen la misma forma estos se pueden expresar de la siguiente manera:

$$\frac{\omega(s)}{U_c(s)} = \frac{K}{s(\mathcal{T}.s+1)} \tag{3.31}$$

Agregando el controlador proporcional con un parámetro K_p a la expresión (3.31) y realimentando la respuesta de salida se obtiene la expresión de la "caja negra":

$$\frac{\omega(s)}{\omega_d(s)} = \frac{K.K_p}{\mathcal{T}.s^2+s+K.K_p} \tag{3.32}$$

En la figura 3.11 se observa el resultado de la identificación experimentando en el subsistema *pitch*, los demás ángulos siguen el mismo procedimiento.

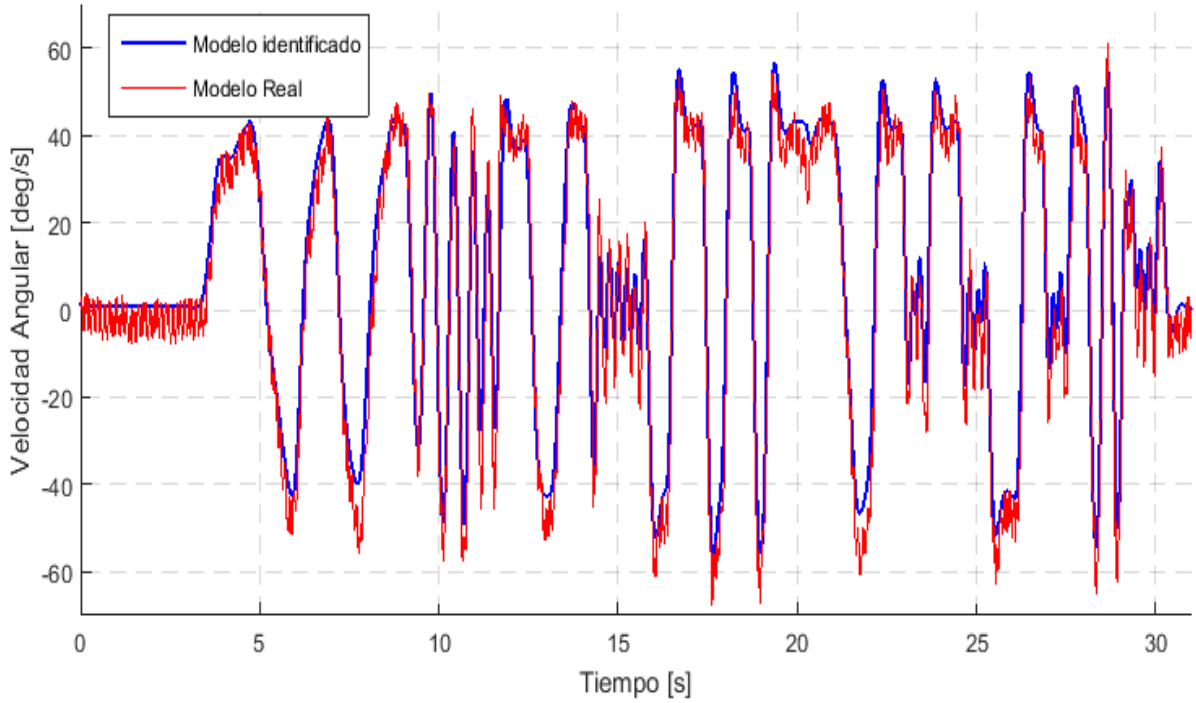


Figura 3.11: Identificación de caja negra de pitch

Para obtener el modelo dinámico del subsistema, que tiene la forma de la expresión (3.31) se realizó una regresión al modelo físico identificado (3.32) a fin de omitir el controlador P. El resultado obtenido se sometió a simulaciones con el software Matlab con entradas reales del mando de radio control y se comparó las salidas simuladas con las salidas reales del Quadrotor para validar los modelos dinámicos obtenidos. En la figura 3.12 se muestra la validación del modelo experimental del subsistema *pitch*.

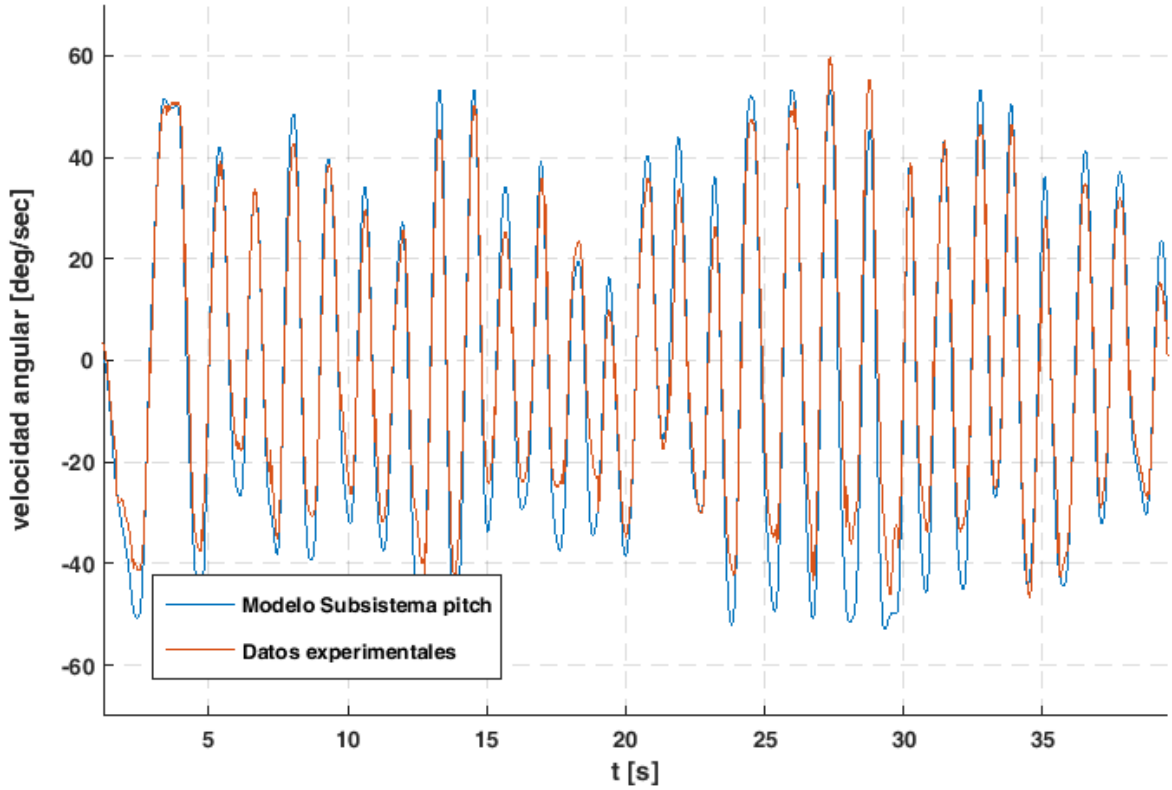


Figura 3.12: Validación del subsistema pitch

El resultado de la identificación de los tres subsistemas se observa en la tabla 3.6.

Tabla 3.6: Modelos experimentales de los tres subsistemas

Subsistema	Función de transferencia
Roll	$\frac{\phi(s)}{U_{\phi}(s)} = \frac{110.931}{s^2(0.0601.s+1)}$
Pitch	$\frac{\theta(s)}{U_{\theta}(s)} = \frac{114.329}{s^2(0.0603.s+1)}$
Yaw	$\frac{\psi(s)}{U_{\psi}(s)} = \frac{8.604}{s^2(0.00024.s+1)}$

Los modelos dinámicos experimentales identificados difieren ligeramente de los modelos teóricos en los subsistemas *roll* y *pitch*. Sin embargo haciendo la comparación de los modelos para el subsistema *yaw* se encuentra que hay una diferencia considerable en uno de los polos de su función de transferencia, donde el modelo dinámico identificado experimentalmente revela que este polo es prácticamente despreciable ya que no influye mucho en la dinámica del sistema.

De las experiencias presentadas y validaciones sucesivas de los modelos dinámicos obtenidos se comprobó que los modelos experimentales del sistema caracterizan mejor la dinámica del sistema Quadrotor y fueron usadas para construir el simulador en el que se prueban los algoritmos de control.

De esta manera se obtienen las funciones de transferencia de cada eje por separado para realizar un diseño de control para cada subsistema de manera independiente. El problema de un sistema de control de varias entradas y varias salidas (MIMO), se redujo a tres sistemas de control lineales e independientes de una entrada y una salida (SISO). Esto reduce considerablemente la complejidad del diseño de control y hace posible el uso de técnicas de control convencional.

3.6 Diagrama de bloques representados en Simulink

Los modelos dinámicos desarrollados se utilizaron para diseñar el simulador de vuelo del Quadrotor que fue utilizado para realizar las pruebas de control. Este simulador fue útil debido a que facilitó el estudio y análisis del sistema para comprender la dinámica del Quadrotor. Además permitió diseñar y probar diferentes tipos de controladores evaluando el desempeño de éstos para seleccionar el controlador más adecuado.

En la figura 3.13 se presenta el diagrama de bloques del sistema Quadrotor mediante Simulink el cual consta de cuatro variables de control, los bloques ya estudiados en el modelamiento dinámico del sistema que contienen los modelos con los parámetros

identificados de cada bloque y las variables de salidas del sistema que describen la dinámica de vuelo del Quadrotor.

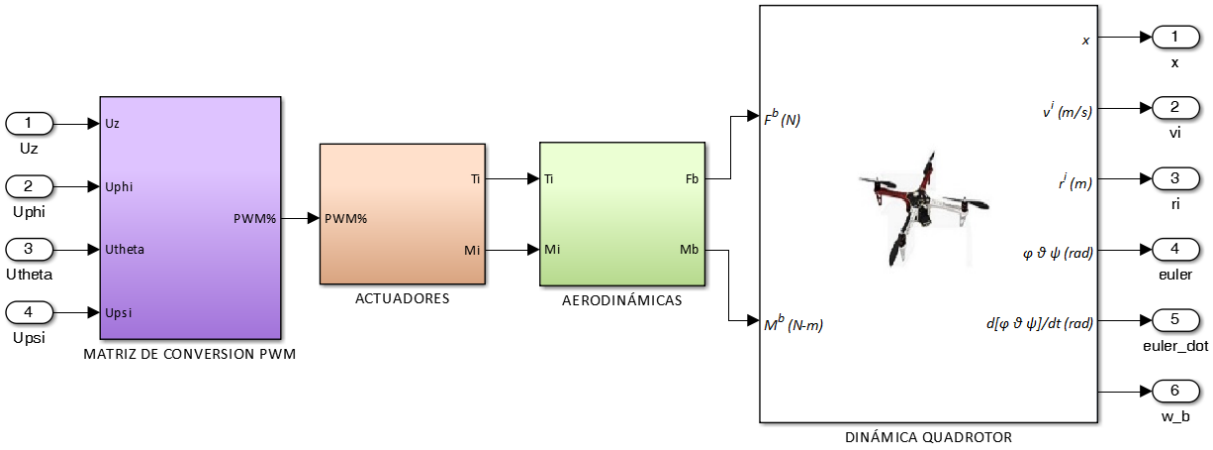


Figura 3.13: Diagrama de bloque del simulador

En las figuras 3.14, 3.15 y 3.16 se observa el funcionamiento interno de cada bloque.

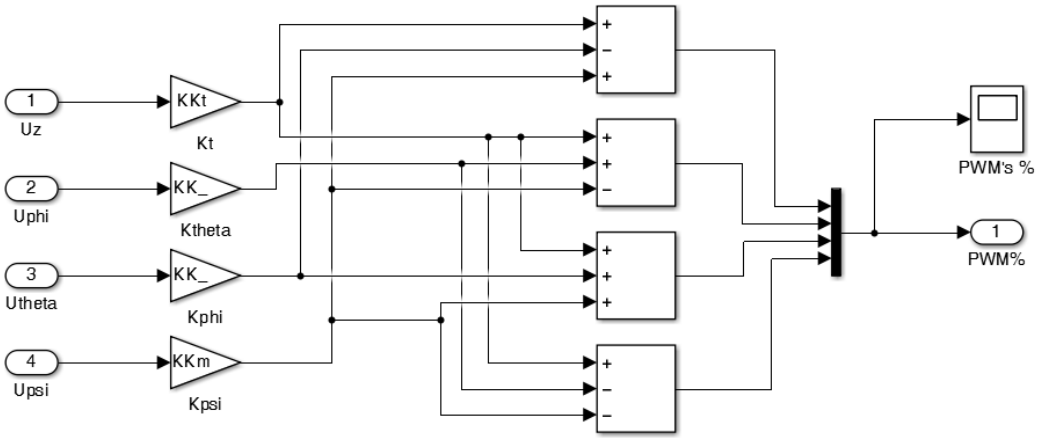


Figura 3.14: Diagrama de bloques de la Matriz de conversión PWM

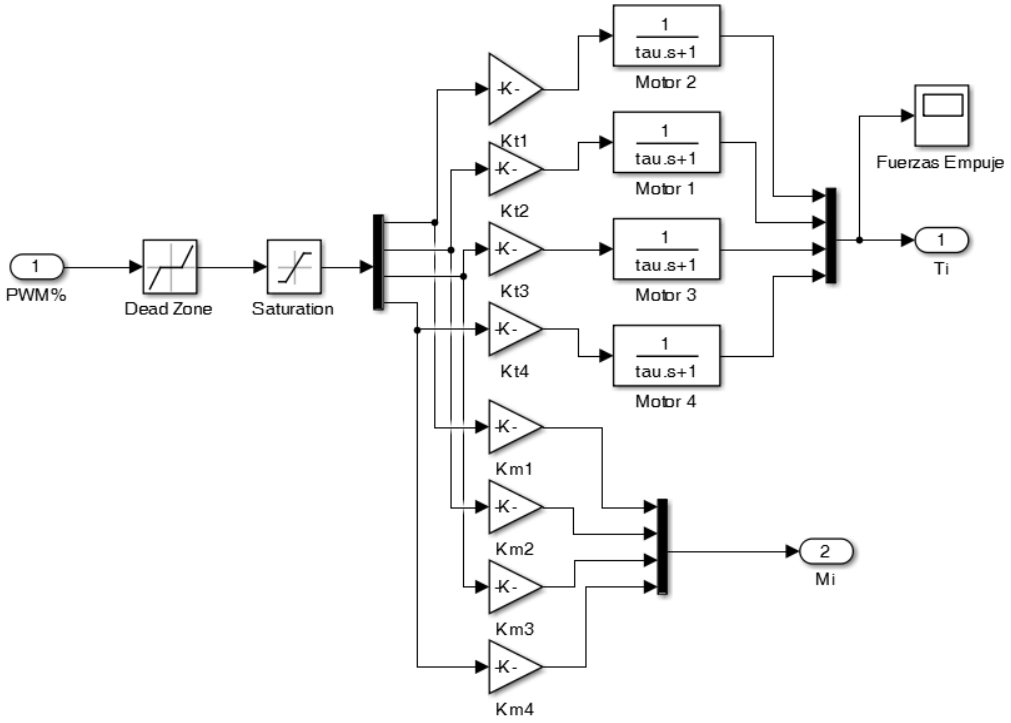


Figura 3.15: Diagrama de bloques de los actuadores

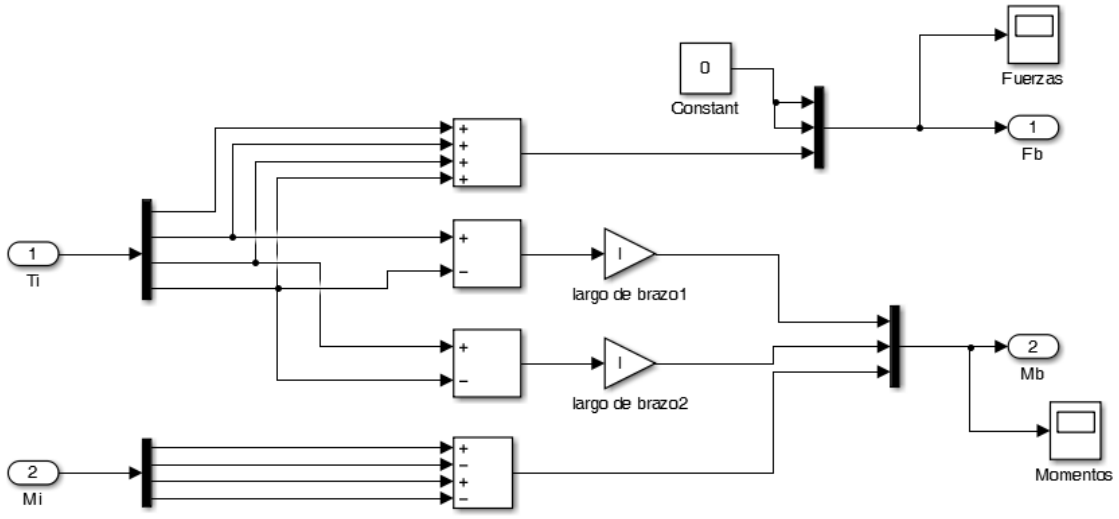


Figura 3.16: Diagrama de bloques de los efectos aerodinámicos

CAPÍTULO 4

ESTRUCTURA HARDWARE DEL SISTEMA

En este capítulo se detalla el diseño de la arquitectura del hardware en la FPGA DE0-Nano. Se presentan los periféricos y módulos diseñados e implementados en la FPGA utilizando el lenguaje VHDL. Se implementa el procesador NIOS II que gobierna los periféricos conectados para cumplir los requerimientos del proyecto. Además se incorpora el sistema de comunicaciones con una estación terrena de control.

Para el presente proyecto se adquirió el *frame* DJI F450 (DJI, s.f.) que se observa en la figura 4.1, debido al bajo costo y porque es un *frame* muy utilizado por desarrolladores en este campo.

Figura 4.1: Quadrotor DJI F450

Para desarrollar el computador de vuelo se adquirió la tarjeta de desarrollo DE0-Nano de la marca Altera basado en la FPGA Cyclone IV, en el cual se implementa todo el diseño descrito en lenguaje VHDL. Se diseñó una tarjeta electrónica de interfaz para realizar la conexión de la tarjeta de desarrollo DE0-Nano con el resto de componentes del sistema de control. En la figura 4.2 se muestra la vista del lado superior del computador de vuelo en el cual se aprecia la tarjeta interfaz, que va sobre la tarjeta de desarrollo DE0-Nano, conectado a una tarjeta IMU y las conexiones con salidas acondicionados para conectar los periféricos PWM, Xbee, receptor de mando a control remoto, de forma opcional un dispositivo GPS y pines libres para cualquiera sensor que se desee conectar al computador de vuelo.

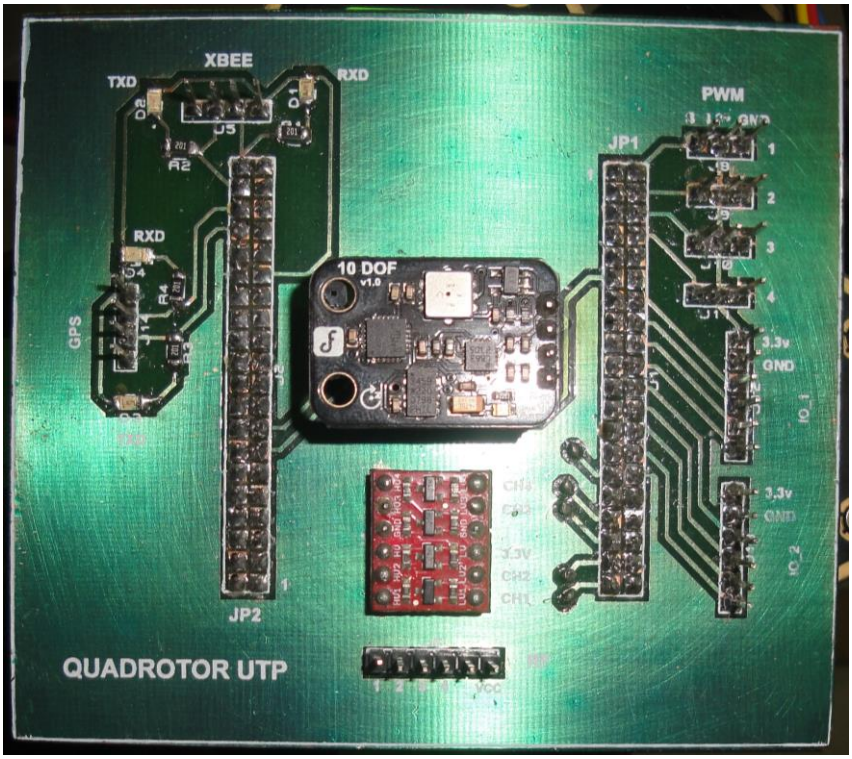


Figura 4.2: Tarjeta interfaz del computador de vuelo

En la tabla 4.1 se muestran los componentes del proyecto, una breve descripción y el peso de cada uno.

Tabla 4.1: Componentes del Quadrotor con sus respectivos pesos

Cant.	Componente	Descripción	Peso (gr)	Subtotal
1	Frame	DJI F450	290	290
4	Motor	2212/920 Kv	56	224
4	ESC	30A Opto	24	96
1	Batería	Turnigy 4500mAh 3S Lipo	389	389
4	Helices	DJI 4x8.5	6	24
1	Computador de vuelo	DE0-Nano kit + Tarjeta interfaz	90	90
1	Regulador 5v	Regulador DC-DC	24	24
1	Xbee	Xbee 2mW serie 2 + socket	7	7
1	Receptor	Futaba R2006GS	23	23

La tabla 4.1 muestra los componentes sin considerar algunos cables, tuercas, cintillos que se usaron, por lo que el total del peso es aproximadamente1200 gramos.

4.1 Arquitectura del hardware

La ventaja de usar una FPGA como controlador es que permite realizar un diseño del hardware de forma personalizada, es decir se diseña solo la lógica necesaria para el funcionamiento del sistema para ahorrar recursos y puede ser modificada y mejorada continuamente si se desea realizar actualizaciones del sistema. Además debido al paralelismo que presenta la FPGA permite realizar procesamiento a altas velocidades que no se podría alcanzar con otros controladores, esto es de gran utilidad para sistemas de

control que presentan dinámicas rápidas y que requieren un procesamiento veloz. En este proyecto de tesis se diseñó el sistema de control de actitud que es el nivel de funcionamiento básico para el Quadrotor, pero es posible implementar otras funciones de mayor nivel como planeamiento de vuelo y aplicaciones de visión artificial. Para estas funciones la FPGA resulta útil debido que tiene la capacidad de realizar la adquisición de más sensores de forma paralela y/o puede utilizar un hardware dedicado para alguna tarea específica reduciendo los cálculos que tenga que realizar el procesador.

El diseño de la arquitectura del hardware en la FPGA fue extenso debido a que el lenguaje VHDL es de bajo nivel. El proceso de diseño involucra tanto la simulación e implementación de cada periférico de manera independiente. La simulación permite corregir errores de diseño y actualizarlo hasta lograr un buen funcionamiento, para esto se diseña en VHDL un banco de pruebas que genera señales que estimulan al periférico en prueba y se estudia las variables de salida observando si se comporta de la manera esperada o no.

La arquitectura implementada en la FPGA presenta varios periféricos o bloques de hardware descritos en lenguaje VHDL que se probaron independientemente para usarlos y evaluarlos en conjunto. Los bloques interactúan con el procesador embebido de nombre NIOS II provisto por Altera utilizando el software de diseño Quartus II. Este procesador consta de 32 bits de datos y procesa a 50 MHz, se programa en lenguaje C, en el cual se desarrollan los cálculos necesarios para ejecutar los algoritmos de control. De esta manera el diseño con FPGA resulta más óptimo ya que abarca tanto el diseño del hardware y el software.

El hardware descrito fue desarrollado de acuerdo a los requisitos del sistema de control, se realizó varias modificaciones en el transcurso del desarrollo del proyecto conforme se desarrollaba el software del procesador con los algoritmos de control de manera que el hardware se adapte a los requerimientos del software.

El sistema completo está conformado por un módulo de comunicación I2C para realizar la lectura de los sensores inerciales, cuatro módulos PWM para control de motores, decodificadores para las señales recibidas del mando a control remoto, periférico de comunicación UART con memoria FIFO para telemetría y el procesador embebido NIOS II en el cual se ejecutan los algoritmos de control. En la figura 4.3 se muestra el diagrama de bloques del hardware del computador de vuelo.

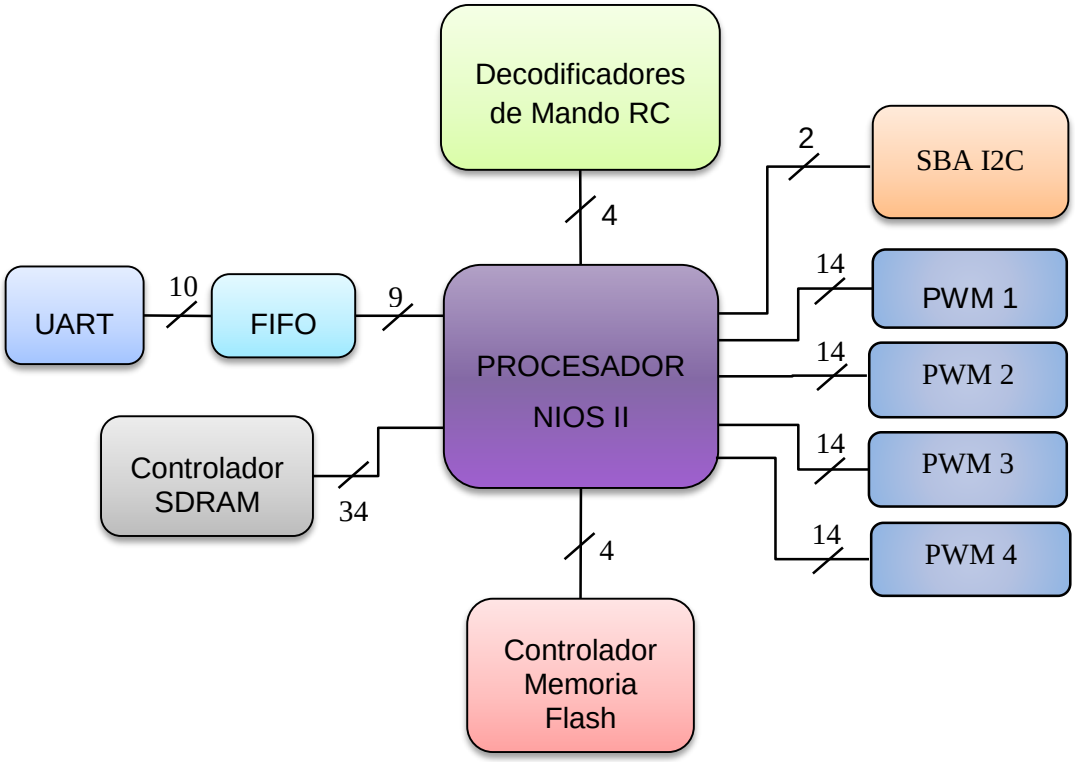


Figura 4.3: Arquitectura del hardware del computador de vuelo

El periférico SBA I2C permite comunicar datos con los sensores inerciales para realizar los cálculos de la actitud. El funcionamiento del periférico está basado en la interfaz de comunicaciones SBA (*Simple Bus Arquitecture*) (Risco, 2014). Mayor información de la adquisición de los sensores inerciales, estimación y determinación de la actitud se puede ver en el siguiente trabajo desarrollado (Huerta R. , 2015).

En las secciones 4.1.1, 4.1.2 y 4.1.3 se detalla el funcionamiento de los bloques periféricos presentados en la figura 4.3 y que se han desarrollado en lenguaje VHDL.

4.1.1 Módulo PWM

Este periférico permite controlar la velocidad angular de un motor Brushless modulando el ancho de pulso a una frecuencia fija de 400Hz (los ESC de los motores Brushless aceptan frecuencias de 50 a 450Hz), el mismo diseño se reutilizó por cuadruplicado para controlar cuatro motores de forma independiente. Cada módulo está conectado al procesador NIOS II mediante un bus de datos de 14 bits que establece el valor de ciclo de operación de la señal PWM y la señal del clock del sistema. La señal de salida de 1 bit de ancho se conecta al ESC de los motores para modular el voltaje mediante la señal PWM.

El diagrama de bloques de la lógica digital del diseño del hardware del módulo PWM se observa en la figura 4.4. Este esquema se conoce como vista RTL (del inglés *Register-Transfer Level*) y es generado por el software Quartus II al realizar el diseño

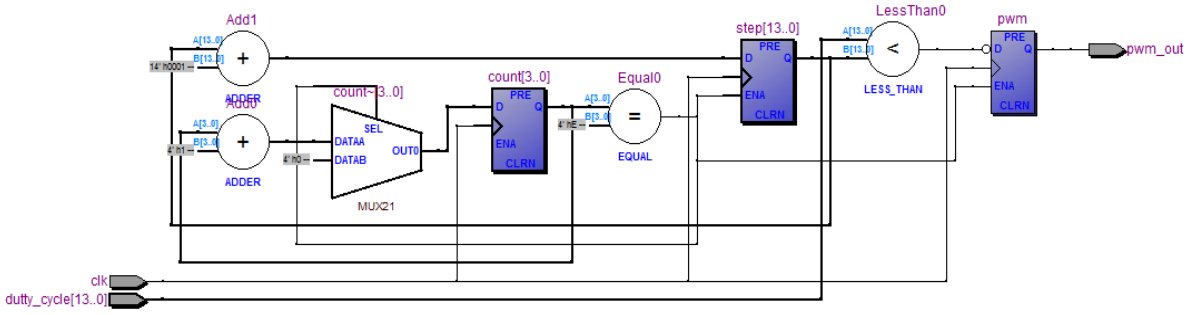


Figura 4.4: Vista RTL del módulo PWM

4.1.2 Decodificadores de mando a control remoto

Las señales del mando a control remoto son captadas por un receptor de cuatro canales que genera ondas cuadradas con una frecuencia fija de 47 Hz o que es lo mismo decir a 21ms de periodo para cada canal. Las ondas cuadradas varían su ancho de pulso en un

rango de 1ms a 2 ms de ciclo de operación según la posición de las palancas del mando a control remoto y se utiliza para establecer las señales de referencia de ángulos y/o velocidades angulares que se desea lograr en el Quadrotor.

La configuración de los canales en las palancas del mando a control remoto se observa en la figura 4.5, en la cual se muestra hacia que posiciones de cada palanca se incrementa o reduce el ancho de pulso de las ondas cuadradas.

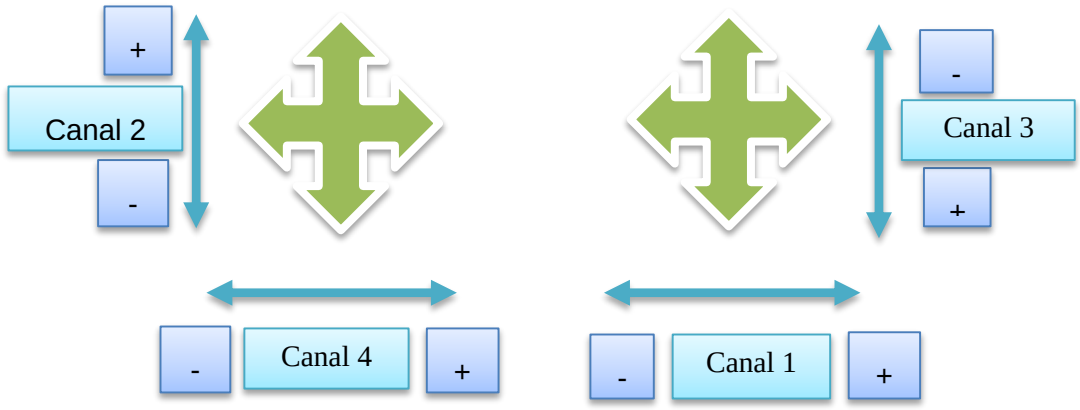


Figura 4.5: Canales del mando a control remoto

Para obtener valores numéricos que indiquen las referencias de ángulos y/o velocidades angulares deseadas, se debe decodificar las ondas cuadradas recibidas del mando captando el tiempo en que la onda cuadrada se mantiene en estado lógico alto, es decir el ciclo de operación. Los cuatro canales del receptor indican referencias para tres ángulos de orientación y uno para elevación, y a cada canal se conecta un periférico decodificador. Los periféricos decodificadores transmiten la magnitud del tiempo del ciclo de operación al procesador NIOS II mediante un bus de datos de 12 bits donde se realiza la conversión a valores numéricos que son usados como los valores de la señal de referencia para los

controladores de ángulo y elevación. La vista RTL del decodificador de un canal del mando a control remoto se observa en la figura 4.6.

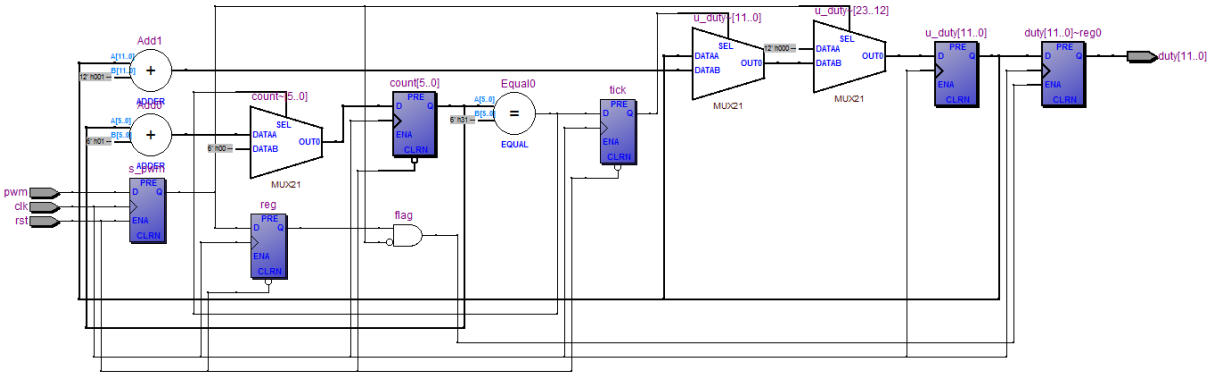


Figura 4.6: Vista RLT de un decodificador de mando a control remoto

4.1.3 Módulo de Telemetría

El hardware de telemetría se utilizó para realizar la descarga de datos del estado de vuelo del Quadrotor hacia la estación terrena de forma inalámbrica. Está compuesto por un periférico UART conectado a un módulo Xbee y una memoria FIFO configurable de 64 bytes de capacidad, aunque se puede extender su capacidad solo con cambiar los parámetros de configuración.

Inicialmente solo se diseñó la comunicación UART para la descarga de datos pero esto retrasaba al procesador NIOS II ya que se transmiten 52 bytes a 115200 baudios lo cual ocupa 1.05 ms y si es que se requiere enviar mayor cantidad de información requeriría aún más tiempo. Por esta razón se diseñó la memoria FIFO para que el procesador solo guarde la trama de telemetría y luego pase a realizar otras tareas mientras que en paralelo la memoria FIFO carga uno a uno los bytes al UART hasta transmitir toda la trama. El tiempo que demora el procesador en guardar la trama en la memoria FIFO es de 48.8us, ahorrando tiempo de procesamiento considerable al NIOS II.

La trama de telemetría se descarga cada 20 ms, tiempo menor al tiempo de muestreo de 10 ms, dado que no necesita operar en tiempo real pues solo sirve para visualizar el estado de vuelo del Quadrotor en la estación terrena además que el Xbee presentó problemas al tratar de forzar la transmisión de toda la trama cada 10 ms ya que se perdían tramas al no poder enviar tanta información en tiempo real.

La vista RTL del hardware de telemetría se observa en la figura 4.7.

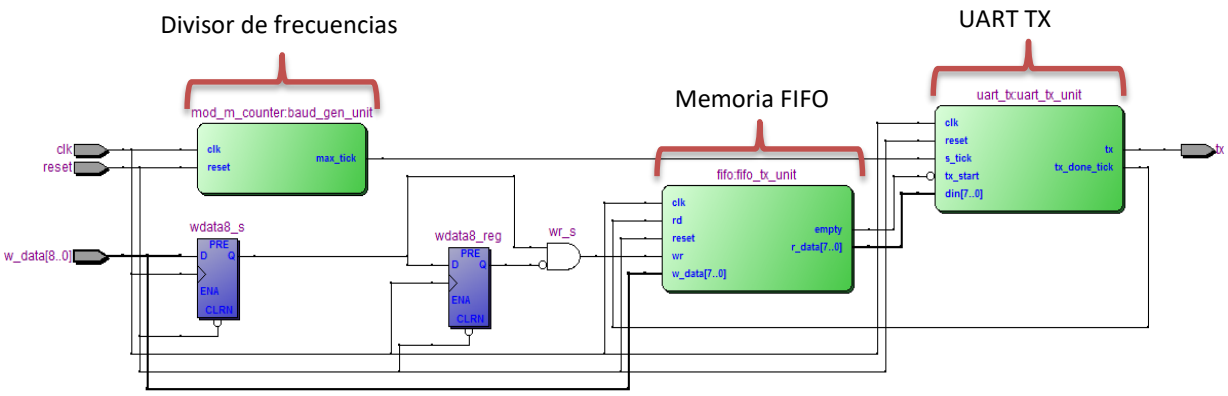


Figura 4.7: Vista RTL del hardware de telemetría

4.1.4 Configuración del procesador NIOS II

Los controladores de memoria SDRAM, memoria Flash y la configuración interna del procesador NIOS II se implementó utilizando la herramienta de diseño SOPC Builder del software Quartus II que permite elegir las características del procesador según el tipo de configuración. Existen tres tipos de configuración que son: NIOS II/f (*fast*), NIOS II/s (*standard*), y NIOS II/e (*economy*) de las cuales se eligió el tipo NIOS II/f debido a que presenta mejores características de las cuales se nombran las más resaltantes a continuación (Altera, s.f.):

- ✓ Presenta arquitectura RISC de 32 bits
- ✓ Utiliza un clock de 50 Mhz

- ✓ Ocupa entre 1200 a 1400 elementos lógicos
- ✓ Presenta unidad de gestión de memoria (MMU del inglés Memory Management Unit).
- ✓ Unidad de protección de memoria (MPU del inglés Memory Protection Unit).
- ✓ Utiliza multiplicadores y divisores embebidos
- ✓ Hardware multiplicador en un solo ciclo.
- ✓ Incorpora hardware dedicado a operaciones de punto flotante
- ✓ Módulo de depuración JTAG

Con la herramienta SOPC Builder se configura el procesador con los siguientes periféricos:

- SDRAM Controller: Controlador que realiza la comunicación con la memoria SDRAM de 32 Mbytes.
- GPIO: para configurar puertos digitales de entrada/salida, se establecieron puertos digitales que se conectan con los periféricos diseñados, además de otros puertos que se dejaron libres para realizar pruebas o para otros posibles sensores que se puedan agregar al sistema.
- TIMER: Se agregó un timer para generar un tiempo de muestreo de 10ms, tiempo suficiente para realizar el procesamiento del software del Quadrotor que incluye la lectura de sensores, determinación de la actitud, manejo de los periféricos y realizar los cálculos necesarios para los algoritmos de control.
- EPCS Serial Flash Controller: Este controlador de memoria Flash se utiliza para almacenar de manera permanente los archivos que contienen el diseño desarrollado tanto el software como el hardware. El archivo **.sopcinfo* contiene toda la información de la arquitectura Hardware, y el archivo **.elf* contiene la información del software, ambos se guardan en la memoria Flash.
- JTAG UART: Interfaz de comunicación entre el ordenador y el procesador NIOS II se utilizó para depurar el software.

- UART: Se habilito la comunicación serial para un sensor GPS. También se utilizó la recepción de un periférico UART para recibir información desde la estación terrena, dado que el hardware de telemetría diseñado en VHDL solo se utiliza para transmisión hacia la estación terrena.

En la figura 4.8 se observa parte de la configuración del procesador NIOS II con los buses y periféricos en la herramienta SOPC Builder.

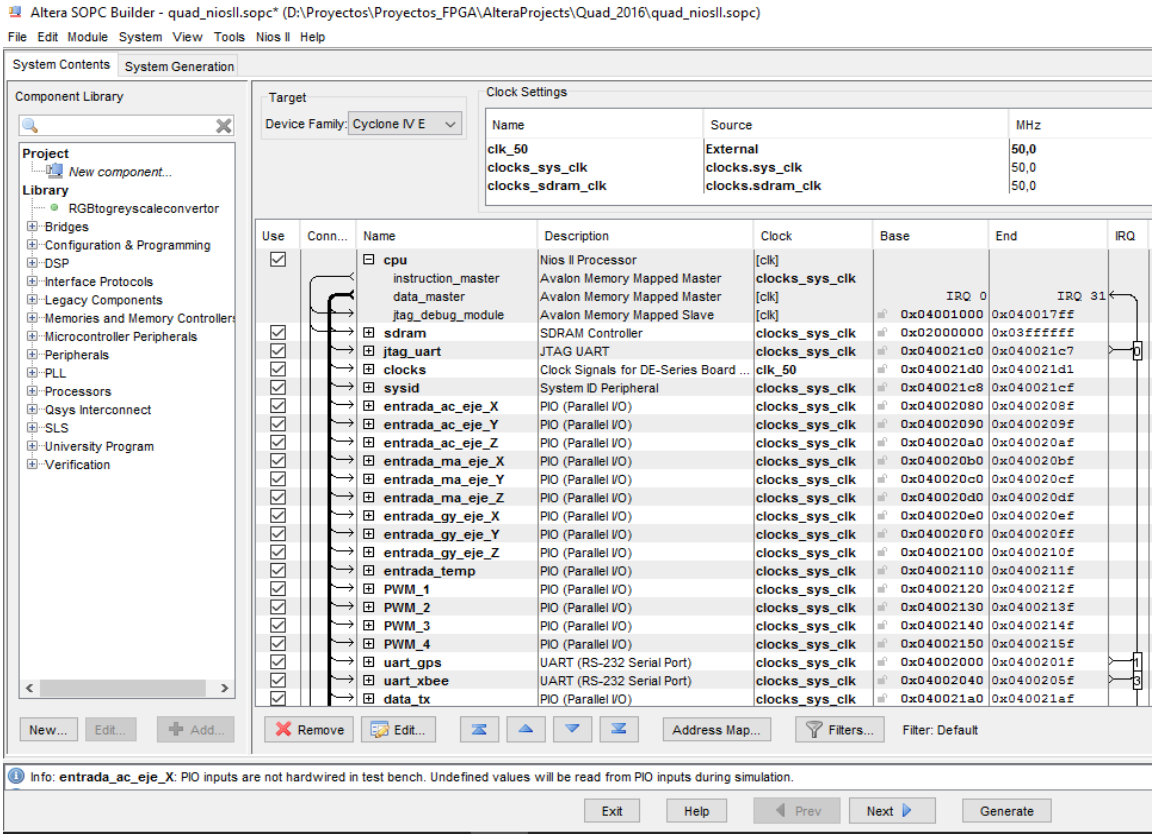


Figura 4.8: Vista de la interfaz gráfica de la herramienta SOPC Builder de Altera

En la figura 4.9 se observa la vista RTL generada por el software Quartus II en el cual se muestra una representación en alto nivel de los bloques de lógica digital involucrados al describir la arquitectura del computador de vuelo en VHDL, su conexión con el procesador NIOS II y los pines de entrada/salida hacia el exterior de la FPGA.

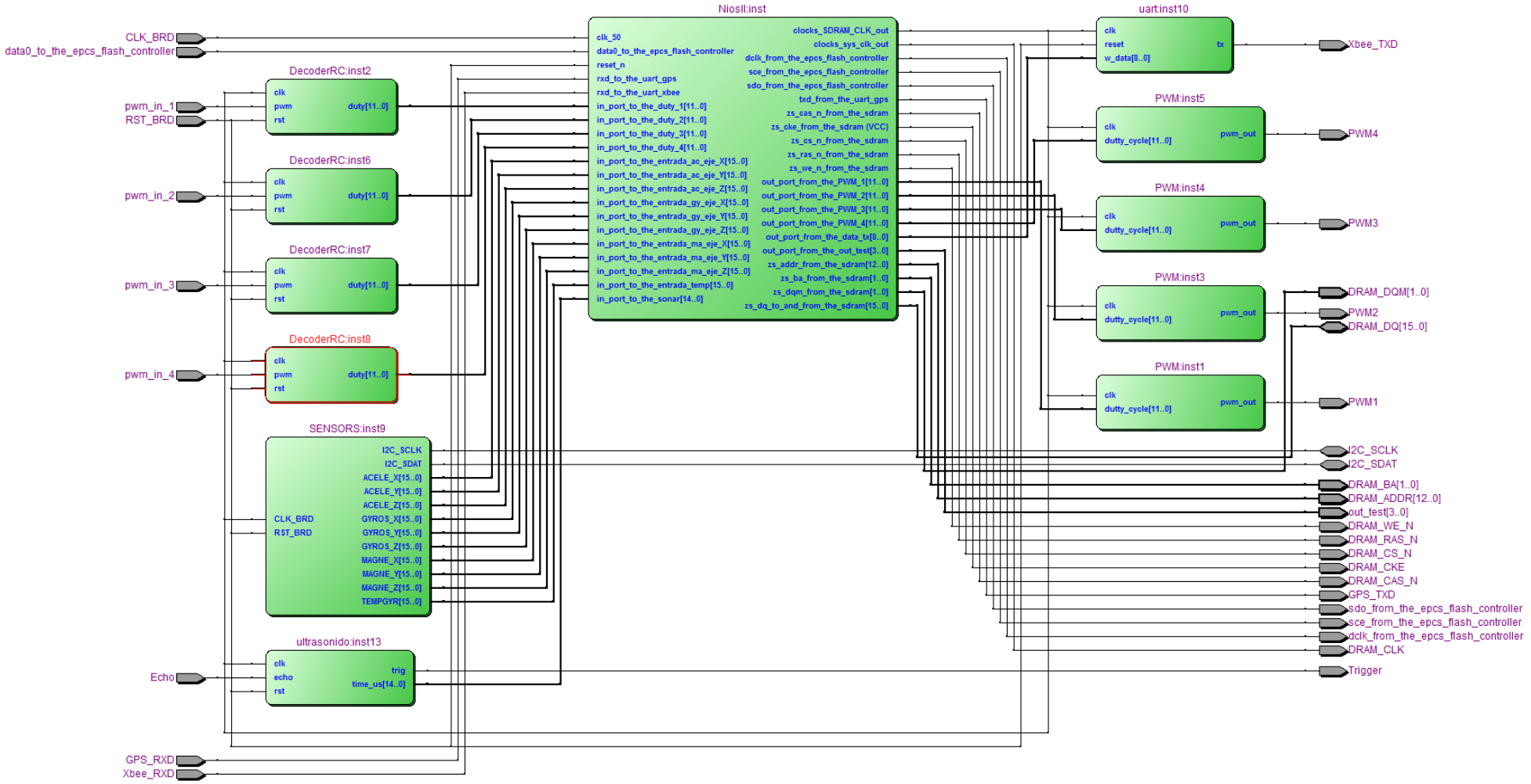


Figura 4.9: Vista RTL de la arquitectura hardware del computador de vuelo

4.2 Estación terrena de control

La estación terrena de control (o GCS del inglés *Ground Control Station*) está compuesta por un computador portátil presentando una interfaz gráfica de usuario desarrollada en Java y un módulo de comunicaciones Xbee para realizar el enlace con el Quadrotor. Su función es monitorear el estado de vuelo del vehículo aéreo mostrando la información recibida por telemetría mediante Xbee de manera que pueda ser aprovechada por el usuario. Los datos de vuelo se muestran mediante los *widgets* de navegación o mediante una gráfica en el tiempo de manera que se puede comparar las variables de referencias de los ángulos con la respuesta real o cualquier información que se desee. Además se puede utilizar para planificar tareas específicas enviando ciertos comandos útiles para el control y planeamiento de vuelo del vehículo.

En la fase intermedia del proyecto se habilitó una trama de envío de parámetros de los controladores PID para actualizar los controladores en línea y de esta manera ajustarlos, sin tener que modificar el código y descargando el programa repetidas veces. Al obtenerse los parámetros finales se deshabilitó esta trama y se modificó el código del programa para inicializar con los nuevos parámetros como constantes y dejar de modificarlos.

La GCS permite grabar la información de telemetría de una prueba de vuelo en un archivo de texto de manera que se pueda reusarlo más adelante para el análisis de vuelo y obtener información útil.

La interfaz gráfica recibe mediante Xbee una trama de 52 bytes de datos a una frecuencia de 50 Hz, es decir cada 20ms. La trama está compuesta por un byte de inicio, los tres ángulos de orientación, las velocidades angulares de cada eje de rotación, las variables de referencias de ángulos, una variable de tiempo y un byte de fin de trama.

En la figura 4.10 se observa la interfaz gráfica desarrollada para el proyecto.

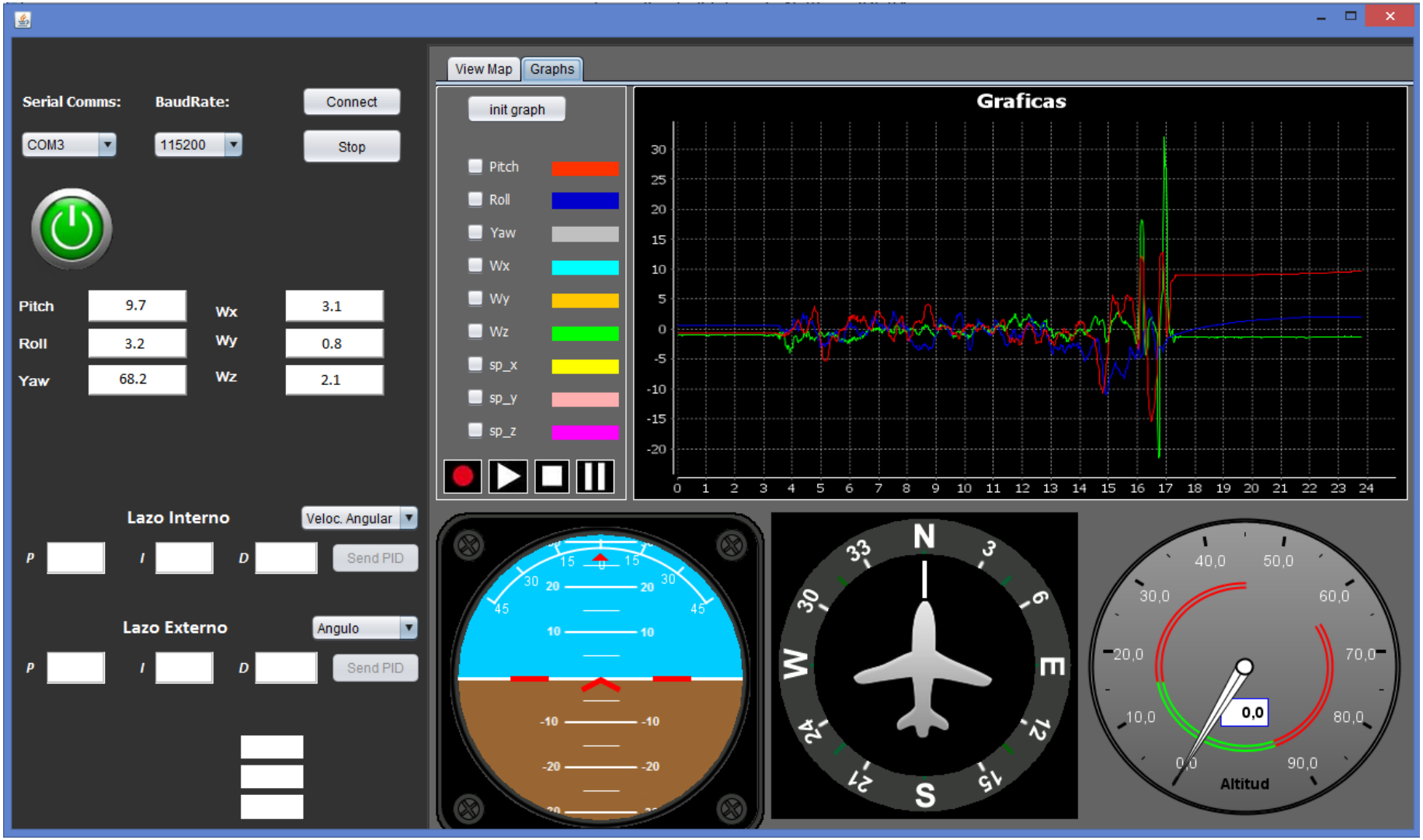


Figura 4.10: Interfaz gráfica de la estación terrena de control

CAPÍTULO 5

DISEÑO E IMPLEMENTACIÓN DEL CONTROLADOR

En este capítulo se presenta el diseño del controlador de actitud del Quadrotor que fue simulado mediante el software Matlab e implementado en el computador de vuelo usando el modelo matemático presentado en el capítulo 3. Se presenta un enfoque de control en cascada basado en la técnica de control clásico PID y algunas variantes del algoritmo para mejorar su rendimiento.

El diseño de control se realizó de forma independiente para cada subsistema con estructura en cascada como se observa en la figura 5.1 que consta de dos lazos de control. El lazo interno contiene un controlador de velocidad angular del Quadrotor y el lazo externo un controlador de actitud. Este enfoque de control resulta favorable ya que el controlador del lazo interno mejora la estabilidad del subsistema facilitando la acción del controlador de actitud del Quadrotor, el cual es una técnica muy utilizada en la aeronáutica (Szafranski & Czyba, 2011), (Kharsansky, 2013), (Jaramillo & Gómez, 2013).

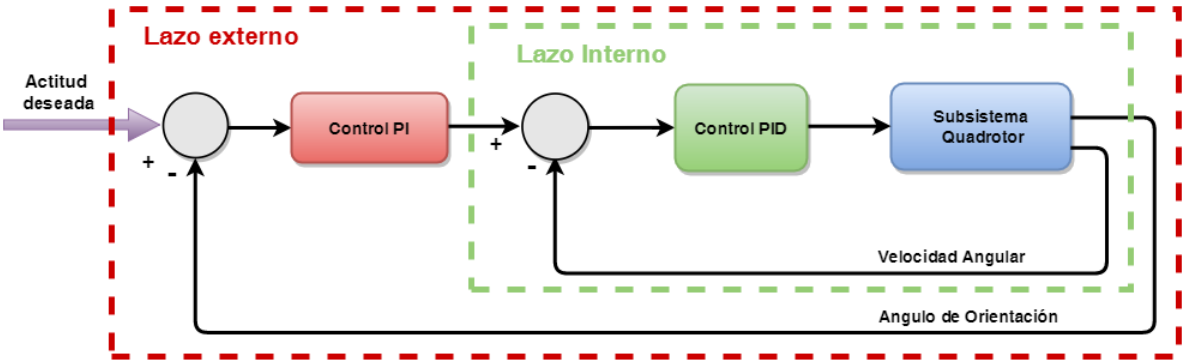


Figura 5.1: Estructura de control para un subsistema del Quadrotor

A continuación se presenta el proceso de diseño de los controladores para los subsistemas del Quadrotor. En la sección 5.1 se presenta el proceso de diseño de control para el lazo interno del subsistema *yaw,* los demás subsistemas siguen un procedimiento similar. Se utilizó el banco de pruebas presentado en la figura 3.9, sujetando al Quadrotor con cuerdas hacia los extremos superior e inferior para restringir el movimiento y que solo rote en el eje z . Los controladores para el subsistema *roll* y *pitch* se diseñaron utilizando el mismo banco de pruebas realizando las experiencias de manera separada para cada eje de rotación.

5.1 Lazo interno

En este lazo de control, también llamado lazo secundario, se implementó el controlador PID del tipo no interactivo (Morilla Garcia) que tiene la forma observada en la ecuación (5.1).

$$G(s) = K\left(1 + \frac{1}{sT_i} + sT_d\right) \tag{5.1}$$

El controlador PID asegura la estabilidad del subsistema controlando la velocidad angular. Se ajustaron los parámetros del controlador para que la respuesta siga a las señales de referencia generadas por el lazo de control externo y además para rechazar las perturbaciones.

Para realizar el diseño del controlador se utilizó un simulador del subsistema *yaw* basado en el modelo dinámico identificado y validado experimentalmente en la sección 3.5.2. Dado que el modelo matemático utilizado en el simulador caracteriza de forma muy aproximada el comportamiento real del Quadrotor, se agregó al simulador el calibrador automático PID que provee Simulink (MathWorks Inc., s.f.) para ajustar las constantes del controlador PID iterando las simulaciones hasta obtener las constantes que cumplan los requisitos mencionados. En la figura 5.2 se observa el simulador diseñado para el subsistema *yaw* con el calibrador automático del PID.

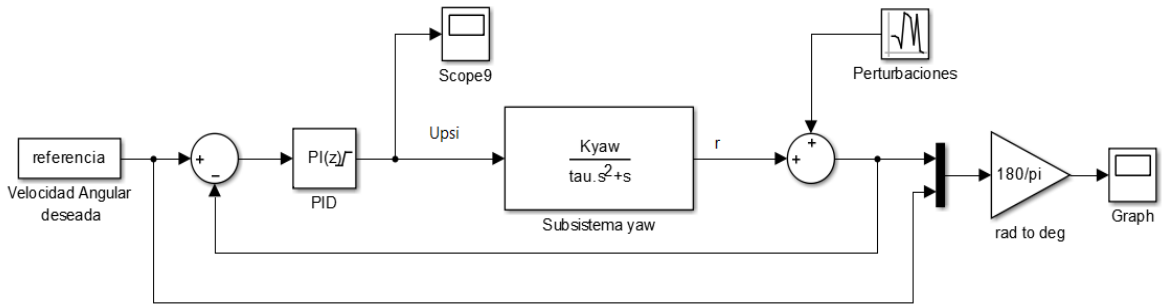


Figura 5.2: Diseño del controlador en Simulink

En el simulador diseñado también se incluyó un bloque generador de perturbaciones para comprobar que el controlador PID rechaza las perturbaciones Se realizaron diversas simulaciones con distintos valores de varianza en la perturbación para observar el desempeño del controlador en cada caso. En la figura 5.3 se observa las simulaciones con perturbaciones donde se puede ver que el controlador presenta un buen desempeño ante distintos valores de varianza pero cuando esta alcanza el valor de 30 la magnitud del error se incrementa considerablemente.

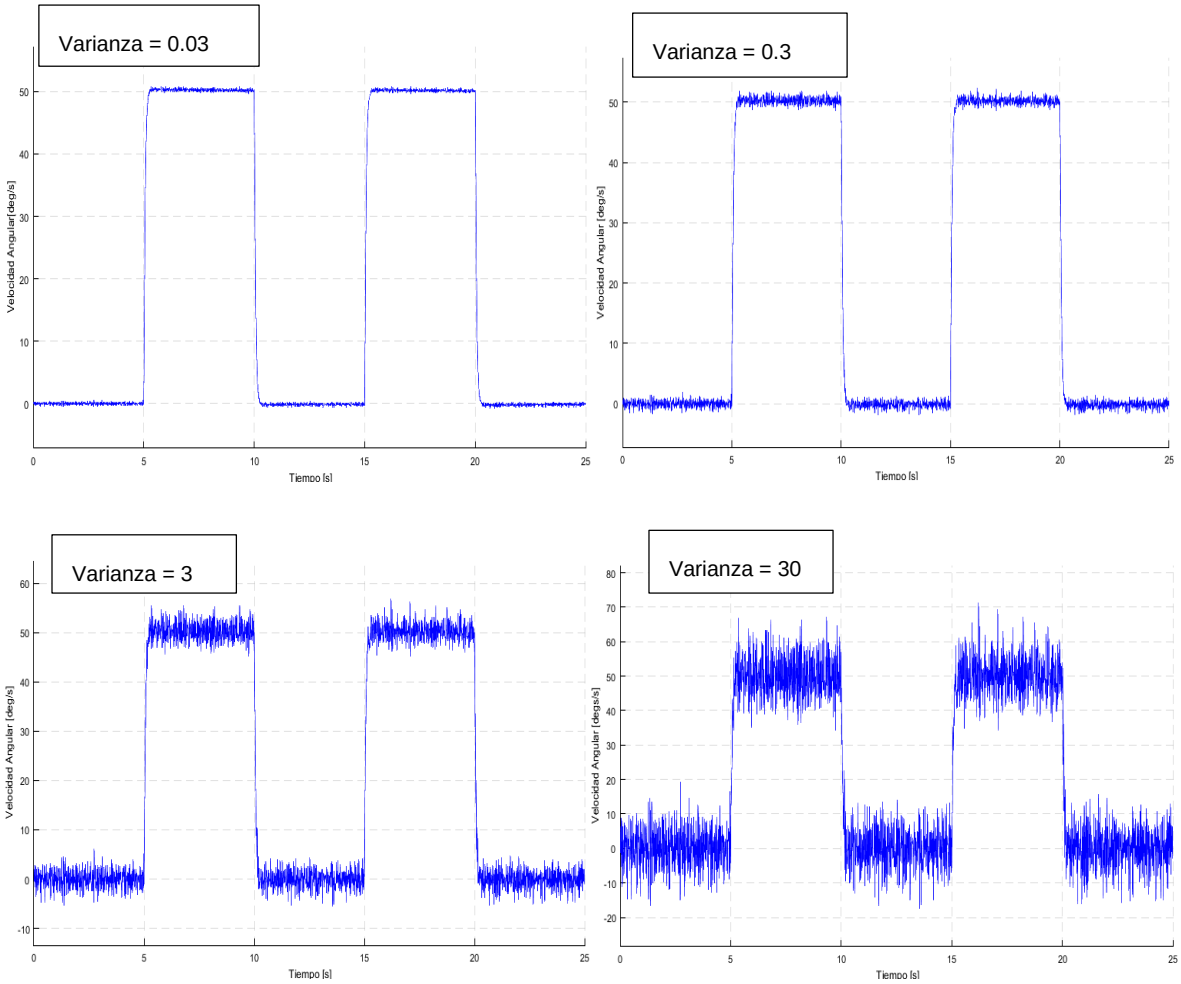


Figura 5.3: Controlador PID ante diversas perturbaciones

Los parámetros PID ajustados en la simulación se implementaron en el computador de vuelo y se realizaron ensayos en el banco de pruebas presentando un correcto funcionamiento en el sistema real. En la figura 5.4 se observa el desempeño de este controlador ante entradas de tipo escalón en la simulación y en la respuesta real, además también se observa que ambas respuestas son similares.

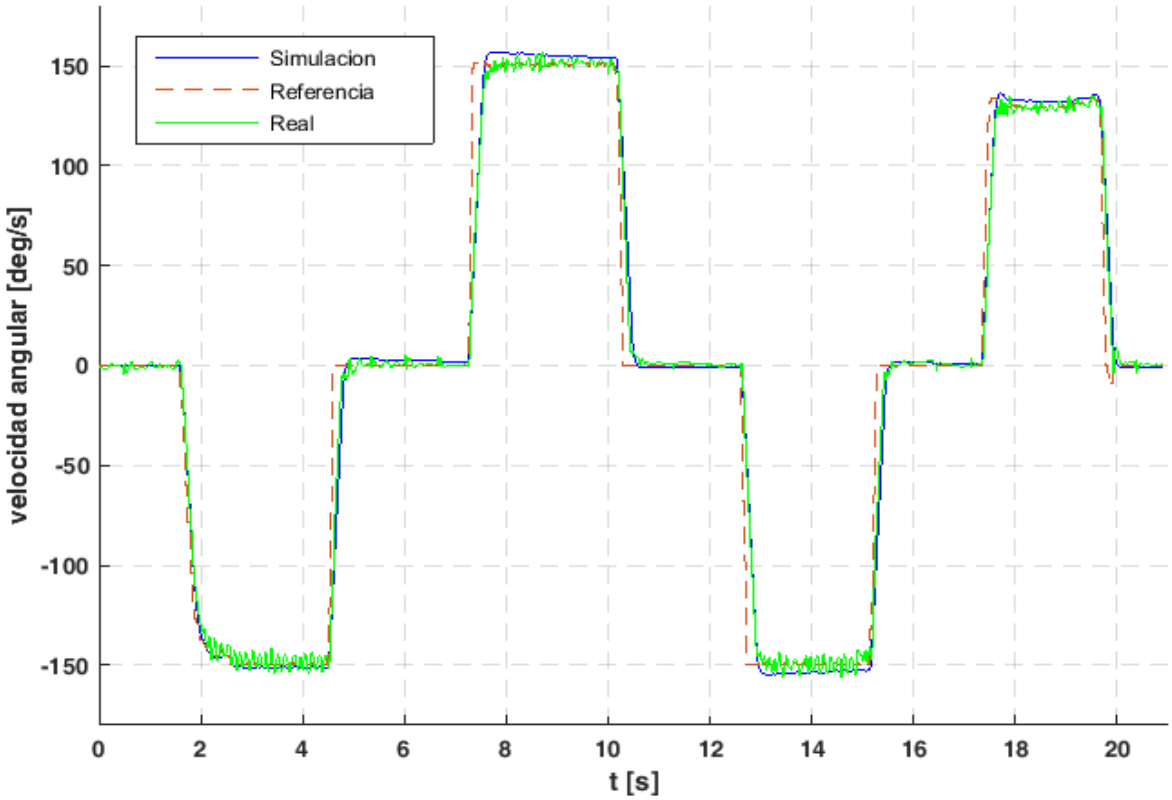


Figura 5.4: Gráfica de control para lazo interno de subsistema *yaw*

En la tabla 5.1 se presenta las constantes ajustadas de los controladores PID de lazo interno para los tres subsistemas que fueron implementados en el computador de vuelo.

Tabla 5.1: Parámetros de controladores de lazo interno

Controladores Lazo interno	K	T_i	T_d
Roll	0.1861	3.69	0.0957
Pitch	0.1861	3.69	0.0957
Yaw	2.48	3.77	0.0

Para mejorar la repuesta de los controladores, de tal manera que puedan funcionar correctamente aun cuando los actuadores saturen durante el vuelo y ante las perturbaciones, se implementó algunas mejoras al algoritmo PID que se detallan en la sección 5.1.1.

5.1.1 Anti-Windup

Una de las mejoras que se implementó fue la del *anti-windup* o anti saturador integral. Cuando el controlador envía una señal de control fuera de los límites de operación de los actuadores, éstos se saturan y la dinámica del Quadrotor no puede llegar a la referencia impuesta o demora mucho en hacerlo. Como consecuencia el error es considerable y el término integral se incrementa en el tiempo hasta crecer indefinidamente. A este efecto se le conoce como *Windup*.

Si el controlador impone señales de control por encima de los límites de los actuadores entonces está funcionando en lazo abierto pues éstos no logran ninguna acción correctiva, además que al intentar regresar a valores de operación normales el término integral debería disminuir tanto como aumento de valor para que el controlador pueda volver a enviar señales de control dentro de los rangos de operación de los actuadores y puedan ejercer acciones correctivas. El *anti-windup* evita este efecto del integrador normalizando el término integral (es decir manteniéndolo en valores normales de operación) cuando el actuador se satura.

El diagrama de bloques del controlador PID implementando el *anti-windup* se observa en la figura 5.5.

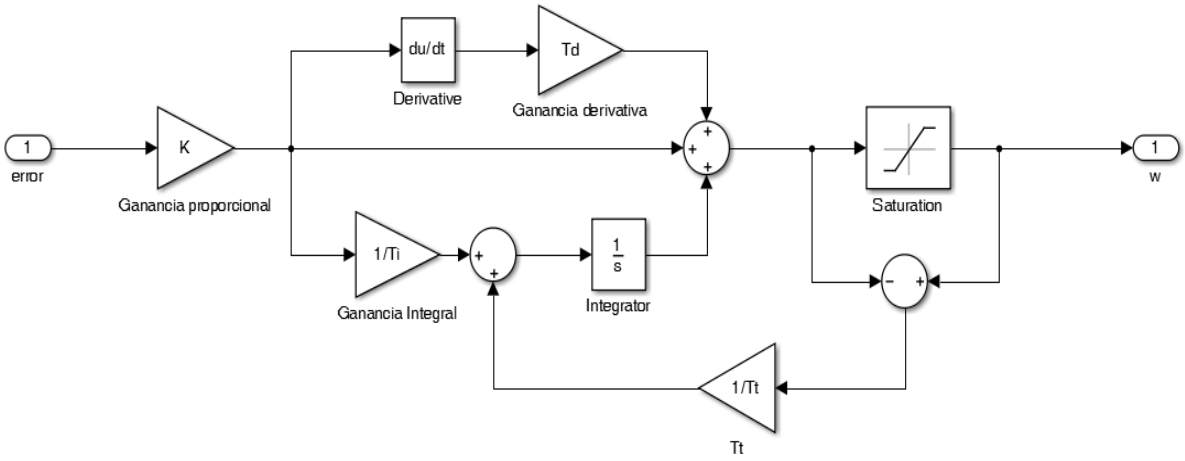


Figura 5.5: Diagrama de bloques de controlador con *anti-windup* mediante Simulink

En la figura 5.6 se muestra la acción del controlador PID sin *anti-windup* para el control de velocidad angular del ángulo *yaw* en el cual se muestra que al enviarle una magnitud de 350 deg/s en la referencia, los actuadores saturan ya que esto supera sus límites de operación y el término integral aumenta debido al error acumulado durante todo el tiempo que intenta alcanzar el valor de referencia. Luego al intentar regresar a un valor de referencia igual a 0 deg/s el controlador no funciona adecuadamente ya que el término integral retrasa la respuesta y el controlador demora más de seis segundos en estabilizar la respuesta e igualar al valor de la referencia.

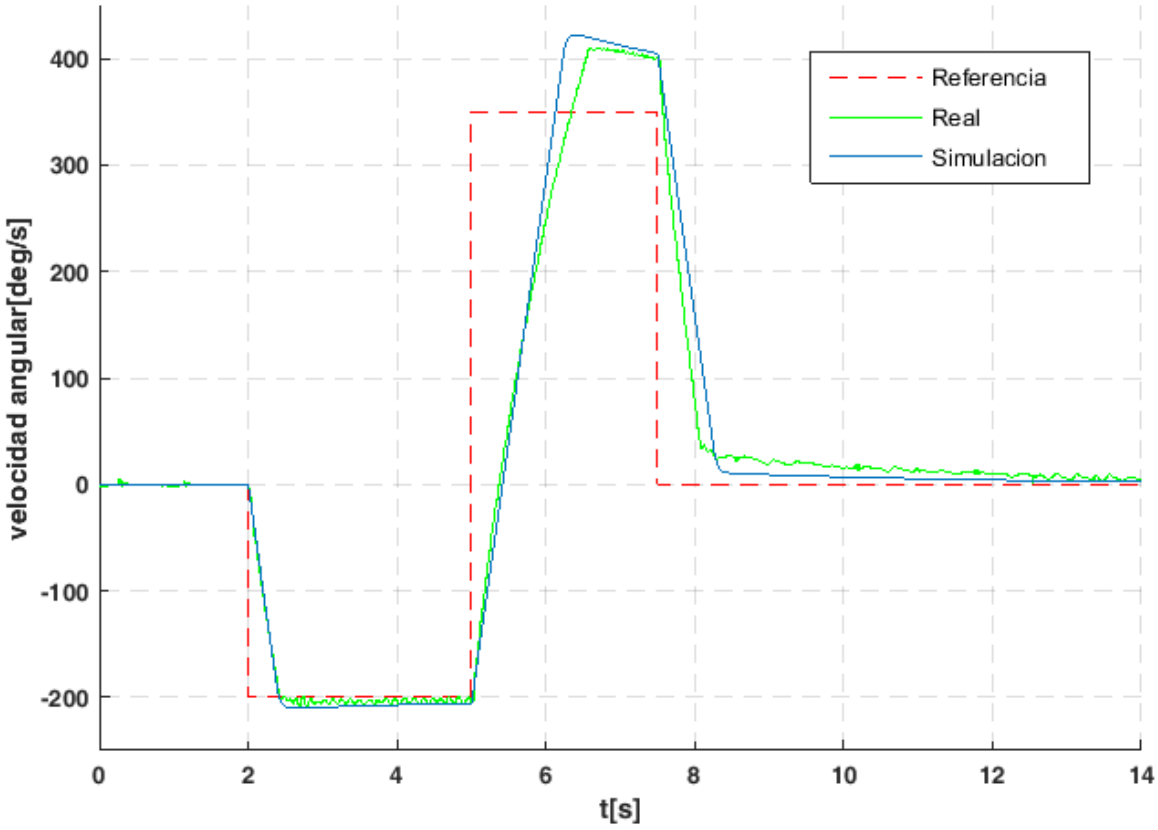


Figura 5.6: Gráfica de controlador sin *anti-windup*

El mismo experimento se realizó usando la implementación del *anti-windup*. En la figura 5.7 se observa que de igual manera que en el caso anterior los actuadores se saturan al tratar de alcanzar el valor de 350 deg/s, pero el término integral se está actualizando cuando esto sucede para no dejar que se incremente demasiado.

Cuando la señal de referencia vuelve a 0 deg/s en $t = 7.5\,s$ el controlador corrige la respuesta adecuadamente porque la magnitud del termino integral está en valores normales y no la retrasa.

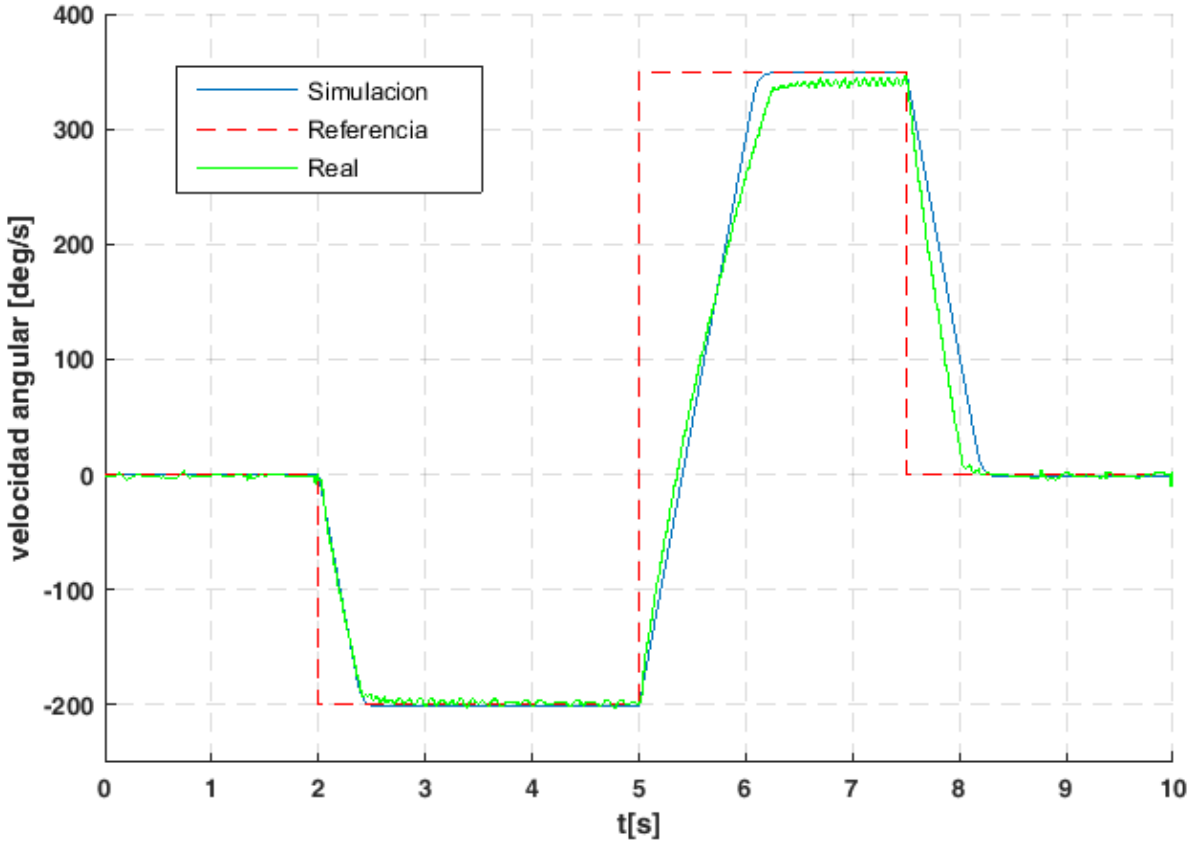


Figura 5.7: Gráfica de controlador con *anti-windup*

5.1.2 Filtro en la acción derivativa

El problema de usar el término derivativo del controlador PID es que la acción derivativa tiende a amplificar el ruido. Para resolver este problema el término derivativo se implementa con forma de un filtro como se puede observar en la ecuación (5.2).

$$H_D = \frac{s.K.T_d}{1 + s.T_d/N} \tag{5.2}$$

En esta modificación se puede considerar que la derivada ideal es filtrada por un sistema de primer orden con una constante de tiempo igual a T_d/N. La aproximación actúa como

un derivador para bajas frecuencias, pero limita la ganancia asociada a las componentes de alta frecuencia a un valor máximo de $K.N$. En la práctica, los valores de N están comprendidos entre 8 y 20 (Aström & Murray, 2002).

El algoritmo de *anti-windup* y el filtro derivativo se simularon e implementaron en conjunto con el controlador PID diseñado previamente para comprobar el progreso en el rendimiento del controlador. En la figura 5.8 se muestra el diagrama de bloques final del controlador con las mejoras diseñadas.

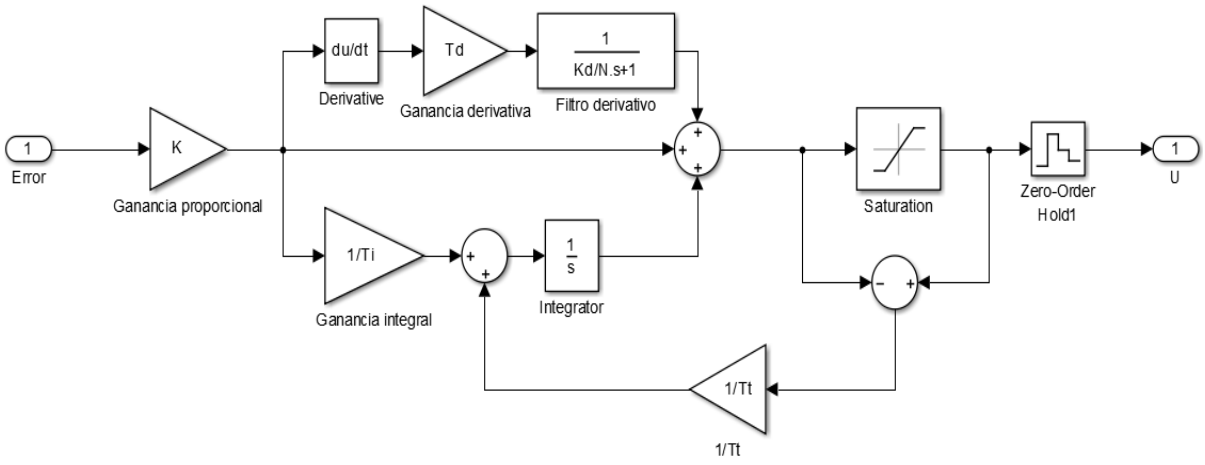


Figura 5.8: Diagrama de bloques de controlador PID mejorado en Simulink

En las figuras 5.9, 5.10 y 5.11 se observa la validación de los controladores mejorados usando entradas de referencia enviadas con el mando de control remoto además del error en el tiempo.

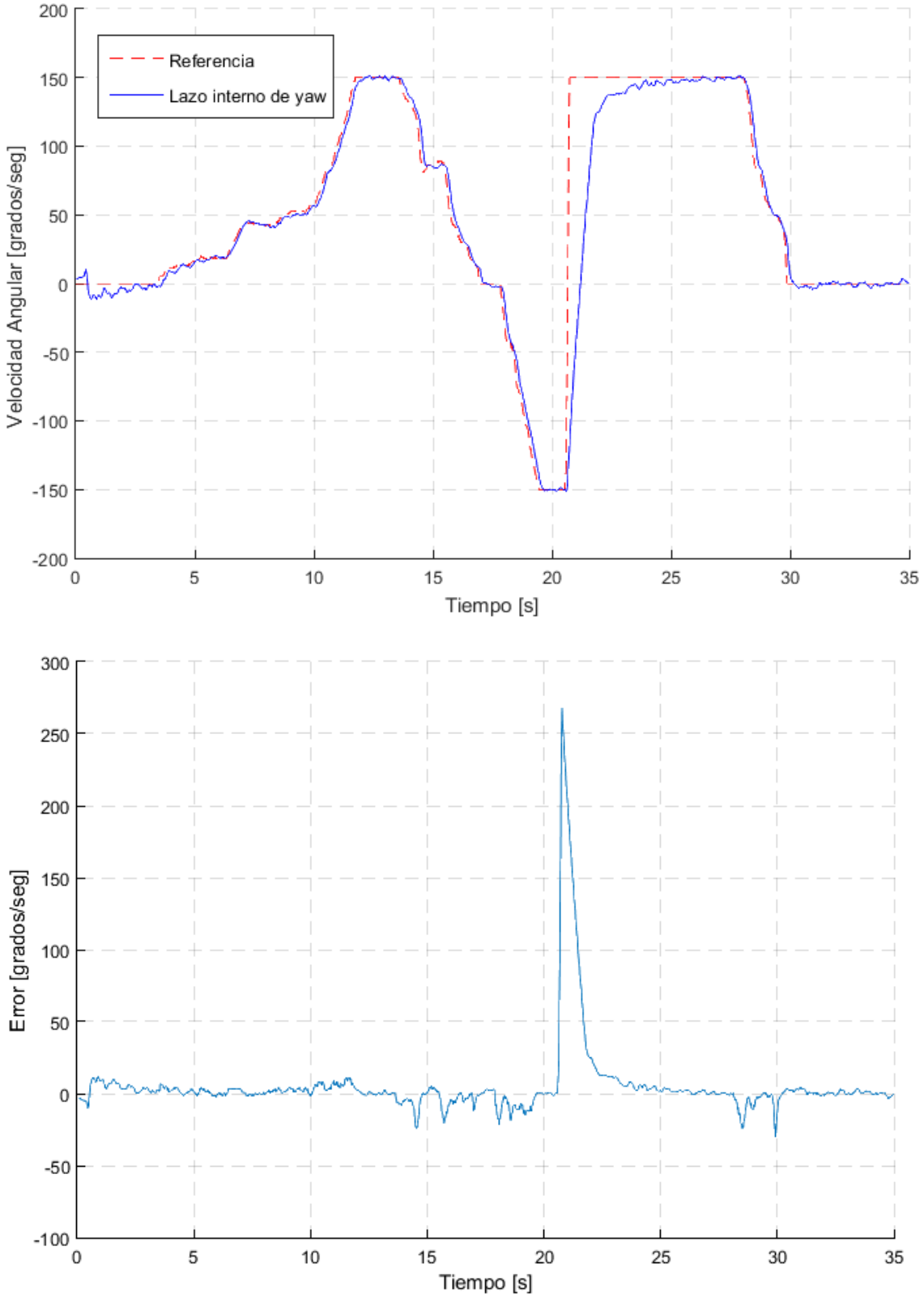


Figura 5.9: Validación de controlador de velocidad angular de *yaw*

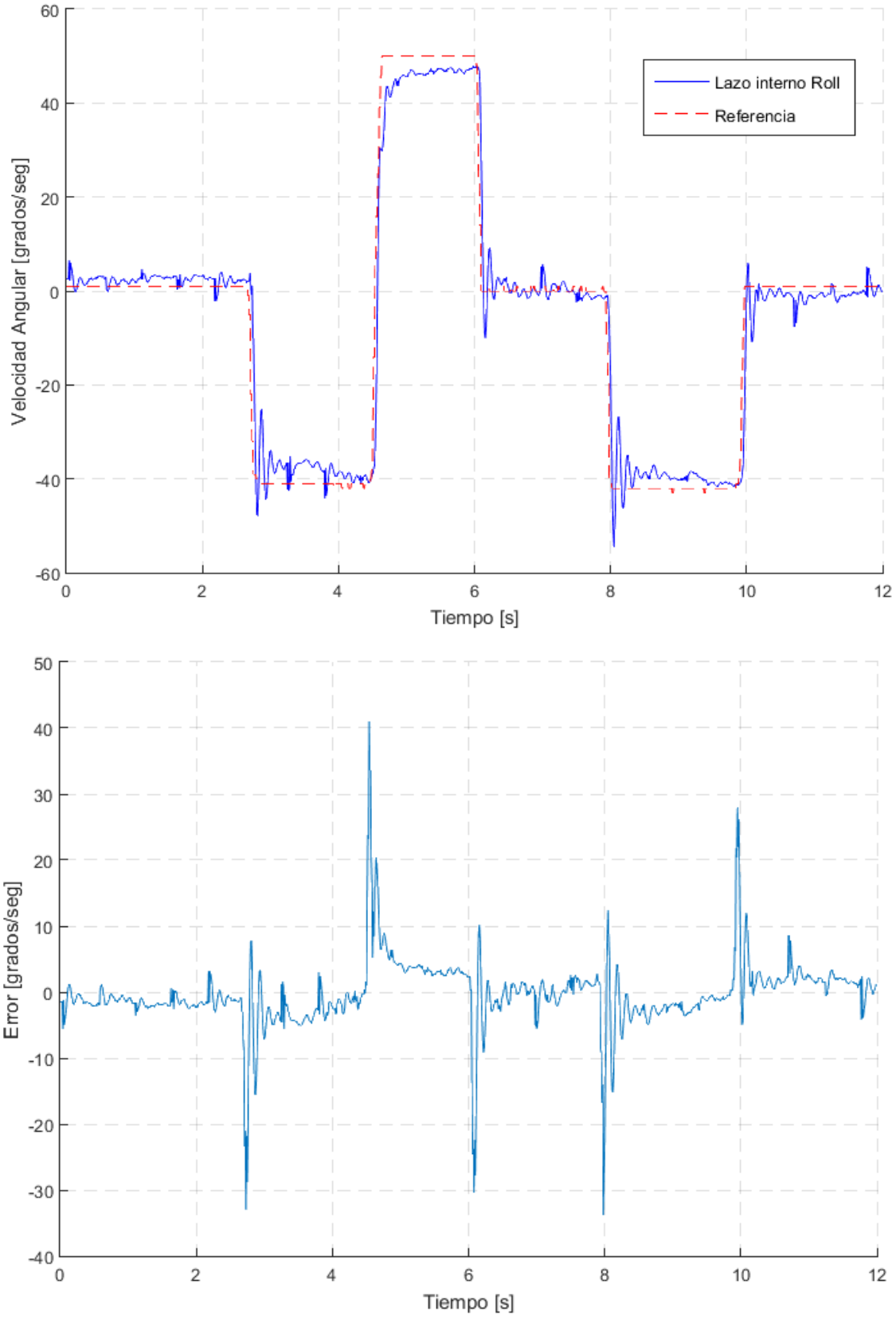


Figura 5.10: Validación de controlador de velocidad angular de *roll*

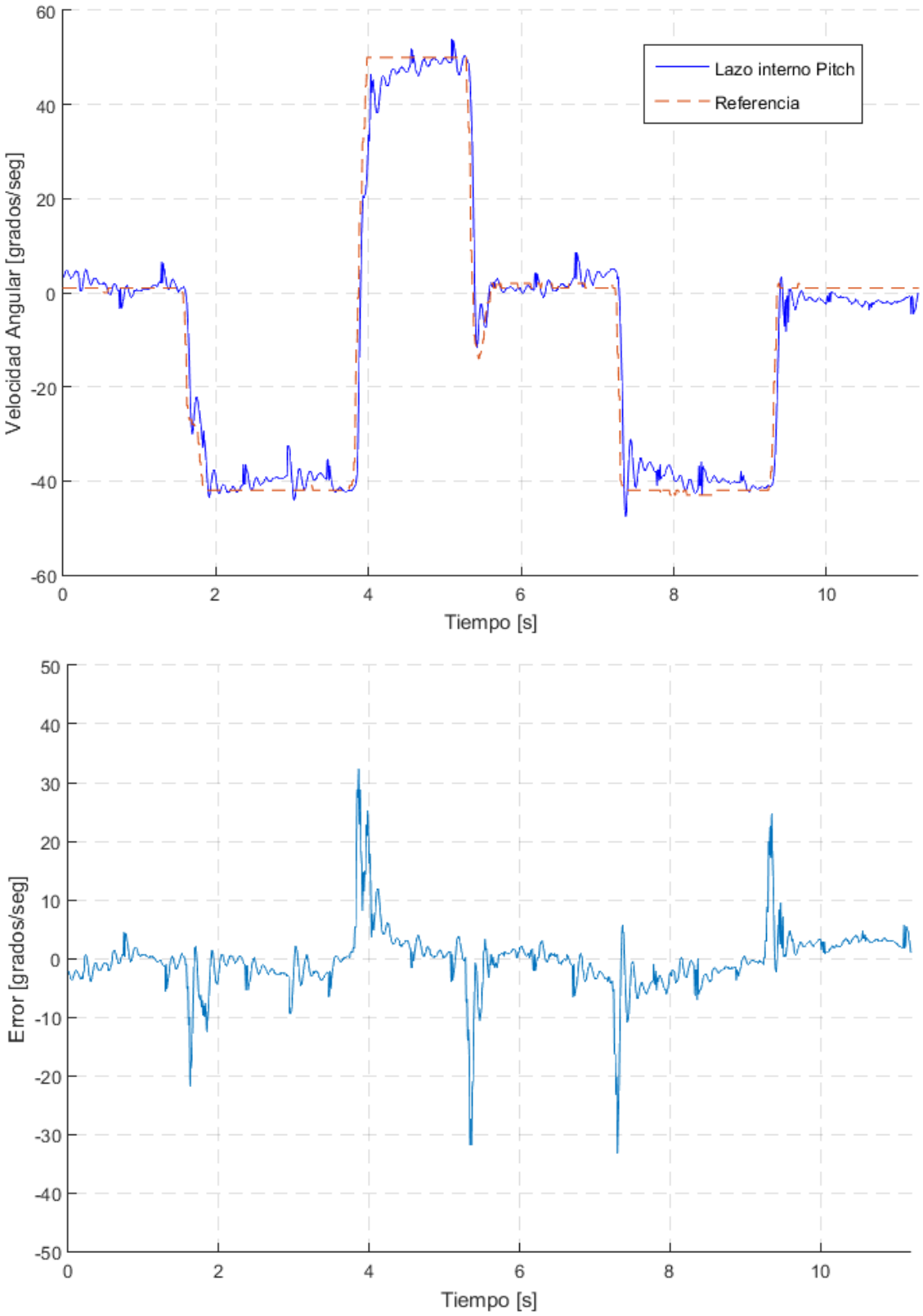


Figura 5.11: Validación de controlador de velocidad angular de *pitch*

5.2 Lazo externo

En este lazo de control, también llamado lazo primario, se implementó un controlador PI suficiente para que el Quadrotor logre la actitud deseada con un buen desempeño. El lazo externo recibe como entrada la señal de referencia que representa la actitud deseada comparándose con la actitud real para obtener el error. El controlador actúa sobre el error y genera acciones correctivas que representan la velocidad angular deseada que envía al lazo interno como se observa en la figura 5.12. Se observa además que a la salida del lazo interno se integra la velocidad angular para obtener la actitud como salida del lazo externo.

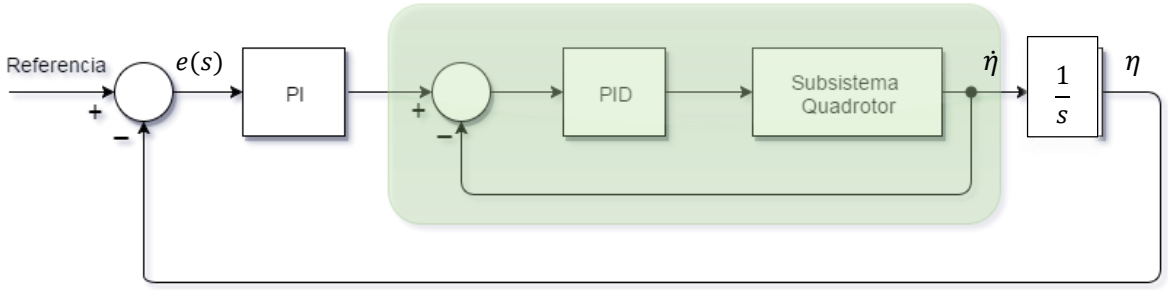
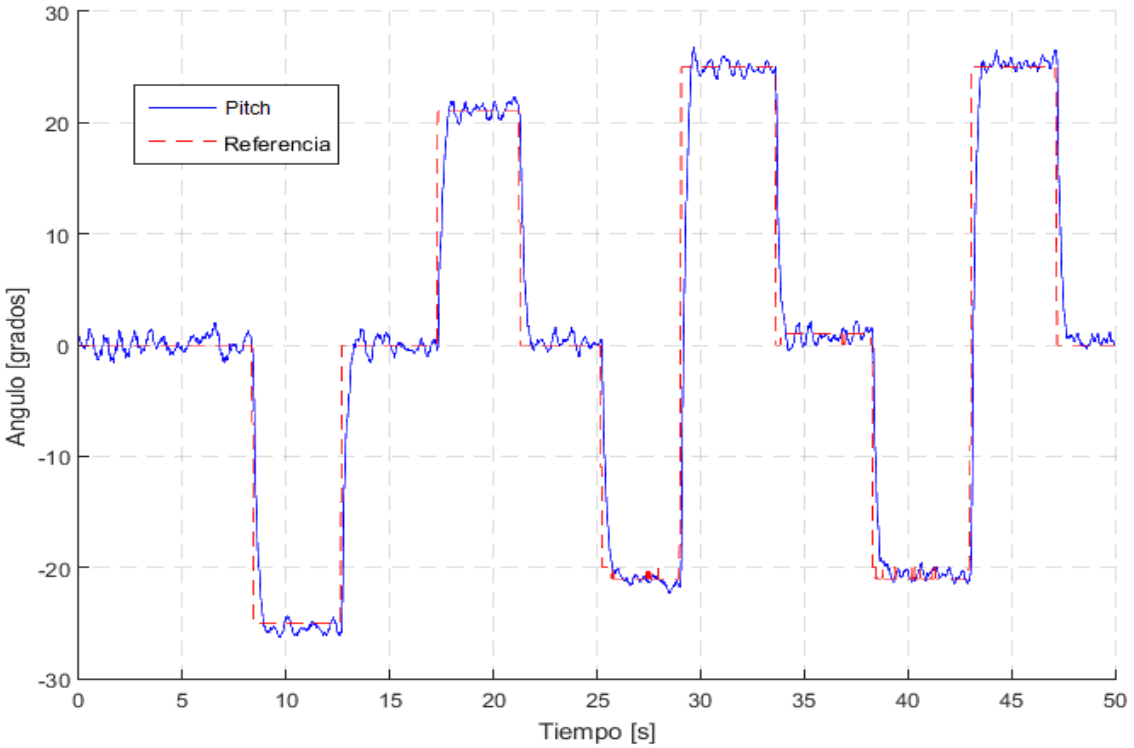


Figura 5.12: Esquema de control de lazo externo

De igual manera que en el lazo interno, se realizó simulaciones en Simulink para evaluar el controlador con los parámetros ajustados. Se ajustó los parámetros del controlador para que el tiempo de establecimiento sea menor a 1 segundo, debido a la rápida dinámica del Quadrotor, para evitar sobre impulsos ya que esto dificulta el manejo del operador de vuelo con el control remoto y para que mantenga al sistema en *hovering* a pesar de las perturbaciones.

Debido a que se requiere controlar la orientación del UAV para que permanezca en *hovering* solo es necesario diseñar el lazo externo de los subsistemas *roll* y *pitch*, debido a que el subsistema *yaw* no influye en el vuelo en *hovering* del Quadrotor, además que

resulta más práctico para el operador que el subsistema *yaw* solo cuente con un controlador de velocidad angular y no de ángulo, pues si el operador requiere ingresar comandos de actitud mediante control remoto para el subsistema *yaw* tendría que mantener todo el tiempo el control en la posición deseada. En cambio con el controlador de velocidad angular solo se envía señales de referencia para que gire hacia el ángulo deseado y cuando alcance ese ángulo se establece una señal de referencia en 0 deg/s para que se mantenga en esa posición. En las figuras 5.13 y 5.14 se observa el desempeño de los controladores para los ángulos *pitch* y r*oll*.



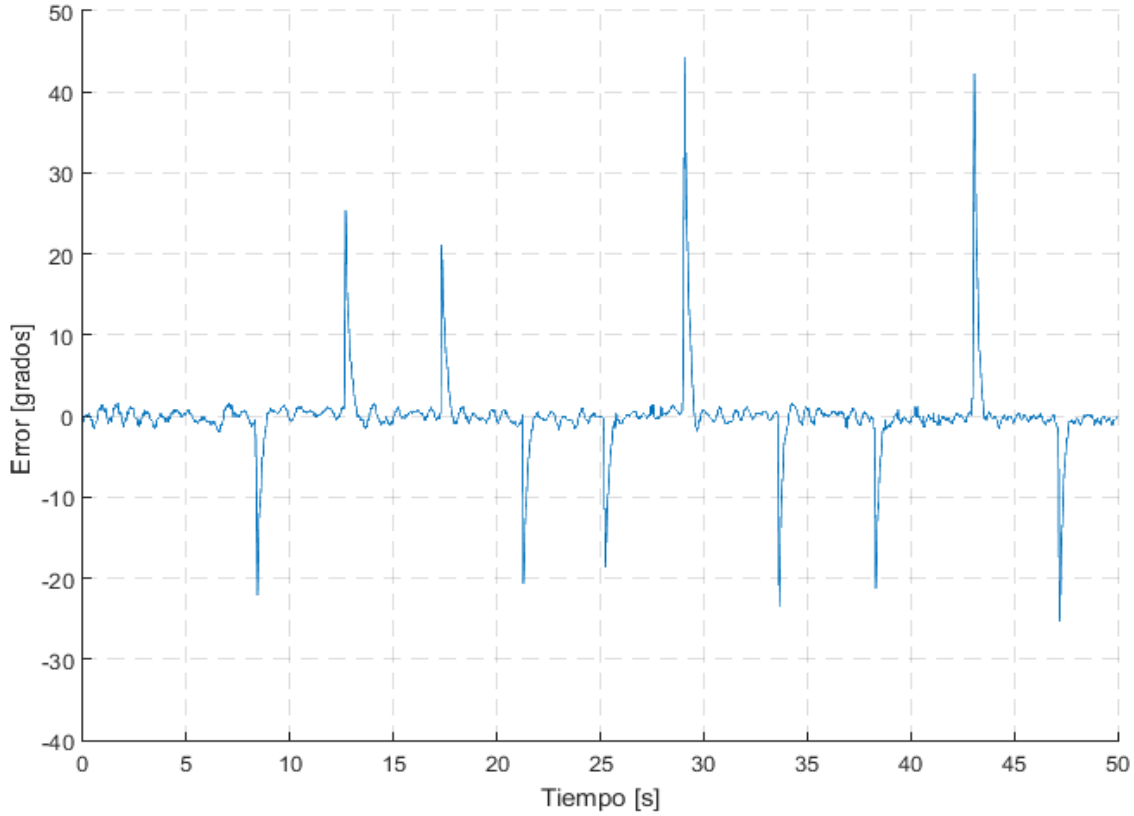


Figura 5.13: Gráfica de control de ángulo *Pitch*

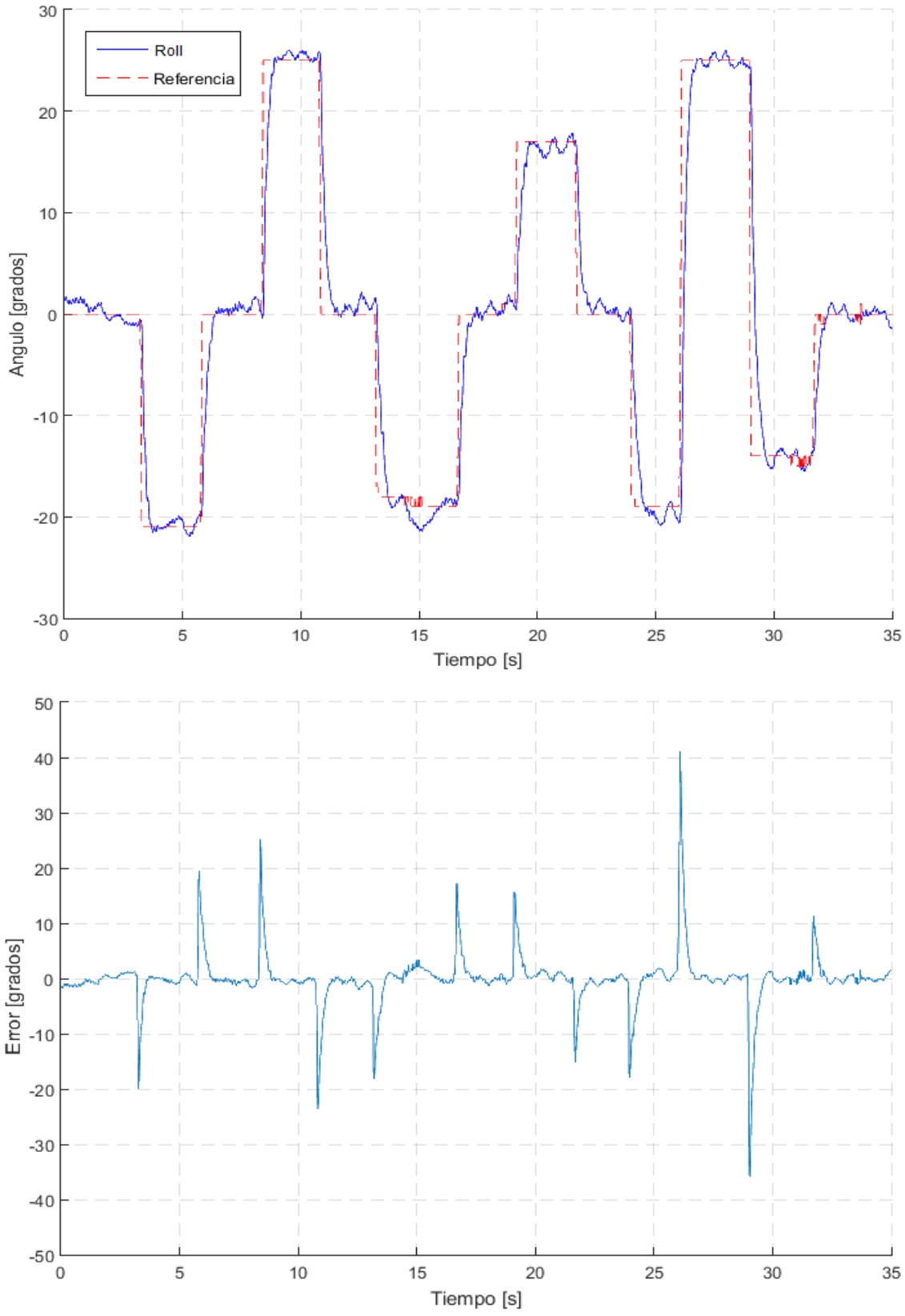


Figura 5.14: Gráfica de control de ángulo *Roll*

Las magnitudes de los parámetros ajustados de los controladores PI para los ángulos *roll* y *pitch* se muestran en la tabla 5.2.

Tabla 5.2: Parámetros de los controladores PI del lazo externo

Controlador	K_p	K_I
Roll	6	0.5
Pitch	6	0.5

En la figura 5.15 se observa el desempeño del controlador de ángulo *pitch* ante perturbaciones que se generaron empujando uno de los brazos mientras el Quadrotor estaba atado en el banco de pruebas con los rotores encendidos a una potencia alrededor del punto de operación. Las perturbaciones se generan en los segundos 19, 31 y 47. Se puede observar que a pesar que las fuerzas externas perturban al Quadrotor desviándolo en distintos instantes de tiempo el controlador es suficientemente robusto para rechazar las perturbaciones presentes pues reacciona rápidamente ante las fuerzas externas corrigiendo la actitud del Quadrotor para estabilizarlo en cero grados en un tiempo relativamente corto.

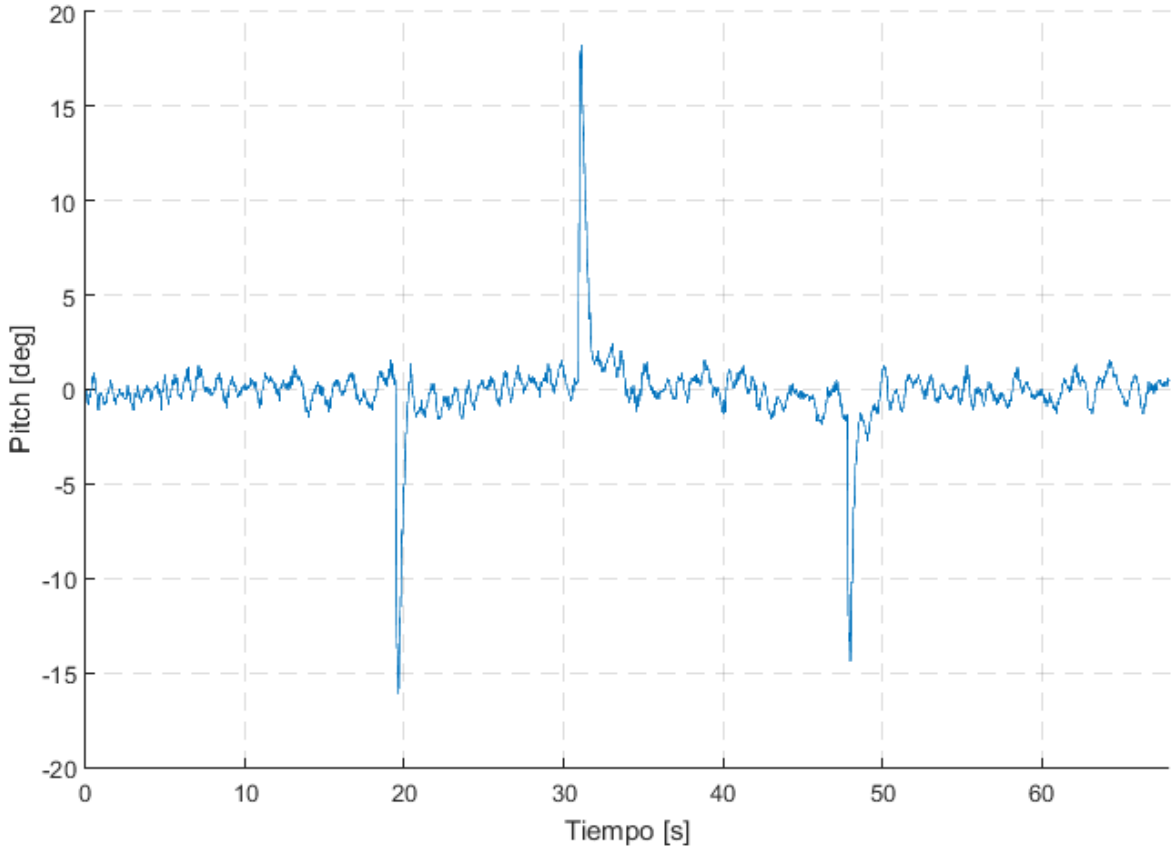


Figura 5.15: Controlador de *Pitch* ante perturbaciones

5.3 Implementación del controlador

El software implementado en la FPGA DE0-Nano se programa utilizando el entorno de desarrollo NIOS II Software Build Tool para Eclipse el cual permite realizar tareas de desarrollo de software como la creación, edición, compilación y depuración del programa. El resultado del desarrollo se descarga a la FPGA a través del cable USB-Blaster. Para la implementación del software se ha utilizado programación estructurada con lenguaje C cuyas líneas de código se ejecutan de manera secuencial, a diferencia del lenguaje VHDL para el hardware. Se ha distribuido el programa en cabeceras que son llamadas desde la hoja principal para hacer el programa legible. En la figura 5.16 se presenta el entorno de desarrollo utilizado para la programación del software mostrando parte del código.

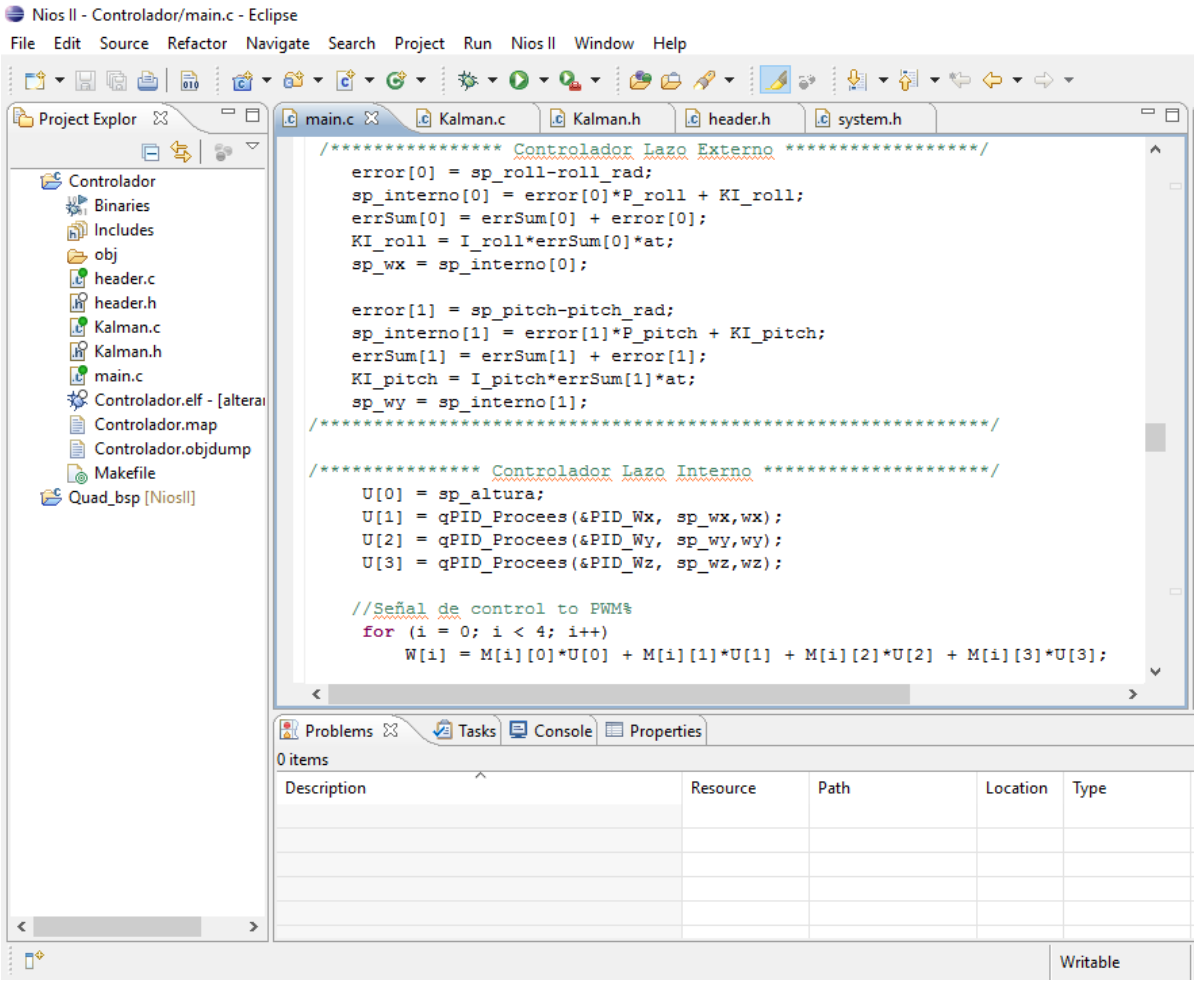


Figura 5.16: Entorno de desarrollo NIOS II Software Build Tool

El diagrama de flujo del programa implementado en la FPGA se observa en la figura 5.17 donde se presenta las principales funciones que realiza el programa. Se observa que después de realizar algunas configuraciones iniciales ingresa a un bucle infinito donde se llama constantemente a la función de estimación de estados, adquisición de referencias y al algoritmo de control que calcula las señales correctivas. Además se observa el diagrama de la interrupción generada por el timer que se desborda cada 10ms, esto es el tiempo de muestreo.

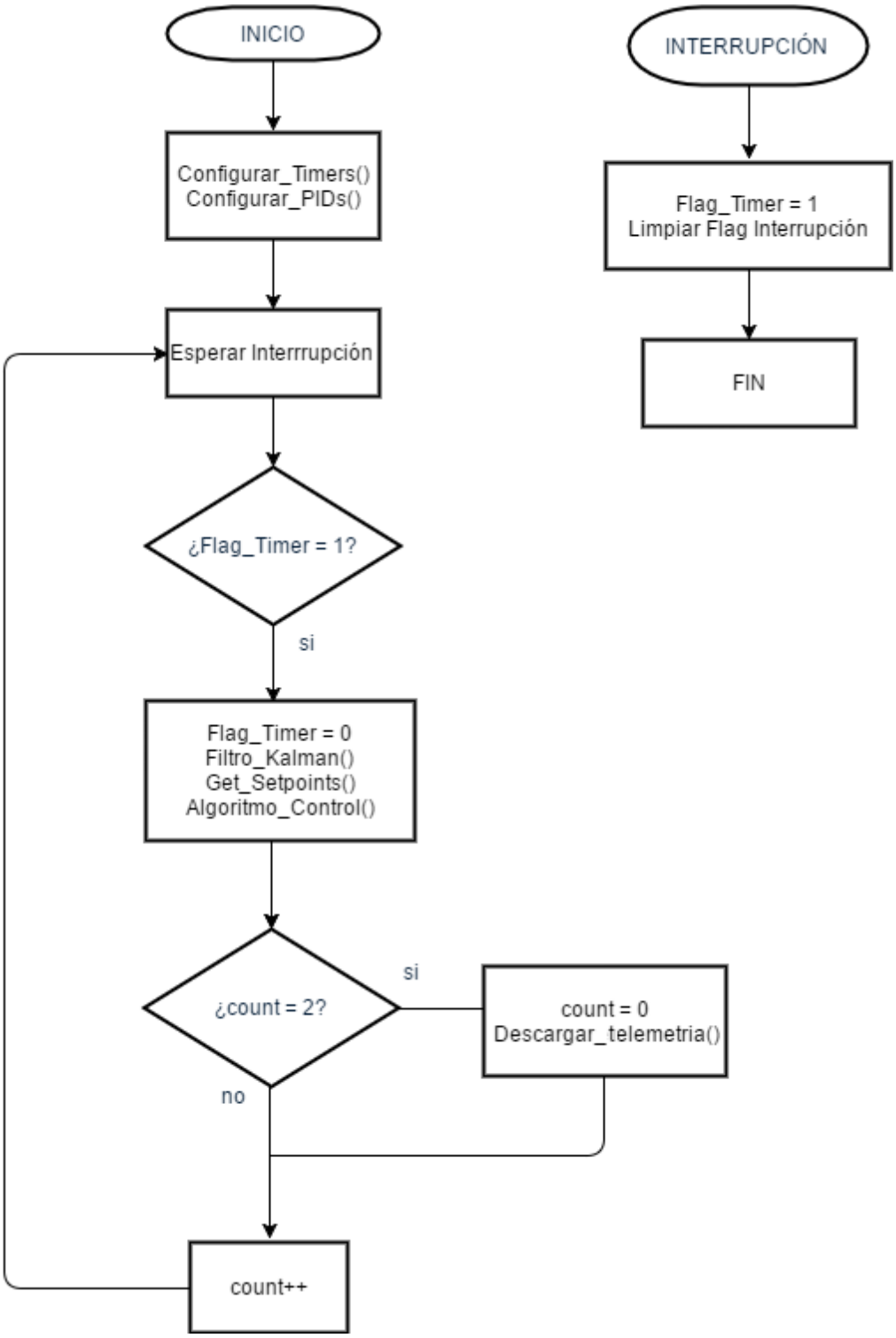


Figura 5.17: Diagrama de flujo de software del sistema de control

La FPGA DE0-Nano dispone de una memoria FLASH para poder cargar el programa de manera permanente, para esto se utiliza el núcleo controlador de memoria FLASH provisto

por Altera e implementado con el SOPC Builder como visto en la sección 4.1.4. En la figura 5.18 se observa el asistente para la descarga del programa a la memoria FLASH en el software NIOS II Software Build Tool.

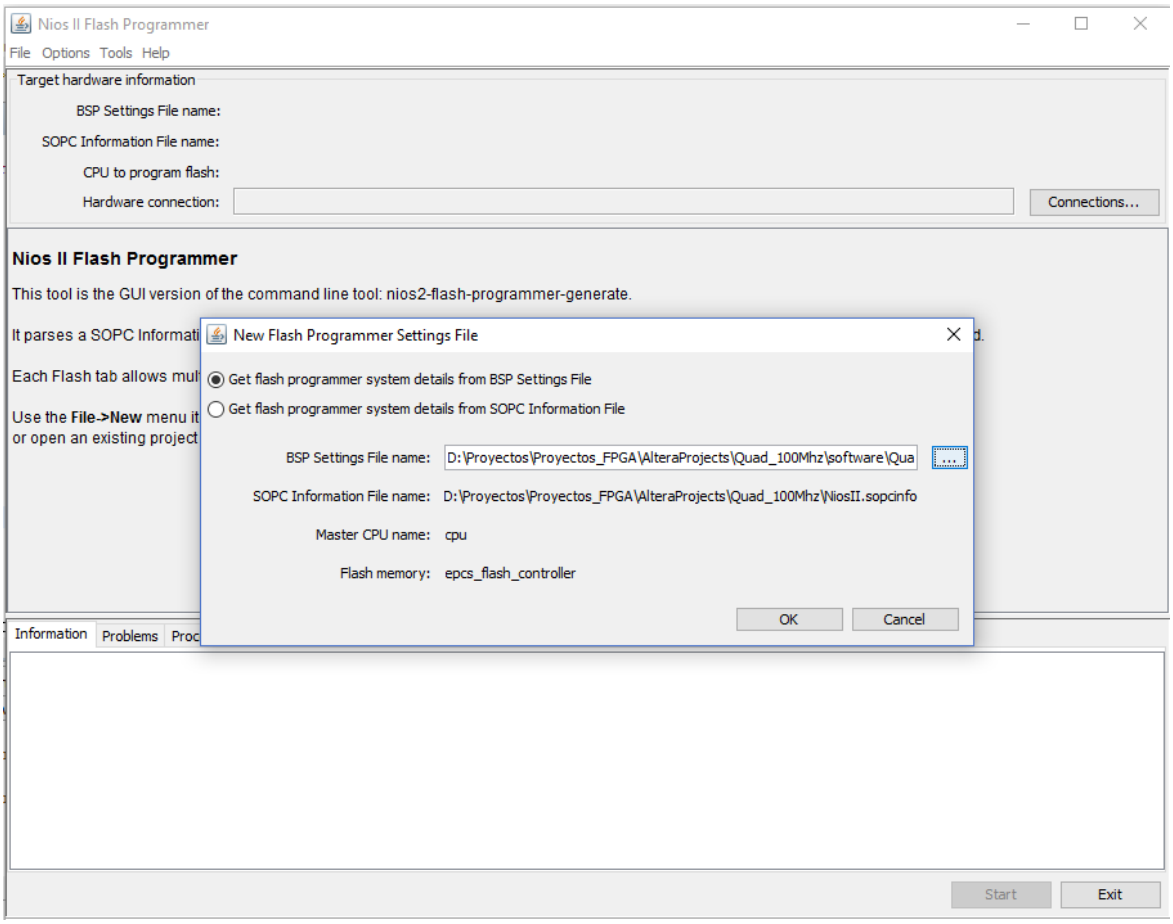


Figura 5.18: Herramienta para programar en la memoria FLASH

CAPÍTULO 6

RESULTADOS Y CONCLUSIONES

En este capítulo se presenta los resultados de la implementación de los algoritmos de control en el computador de vuelo. Se realizaron pruebas de vuelo real usando el mando a control remoto para establecer las referencias de ángulos de orientación.

6.1 Pruebas de vuelo real

Las pruebas de vuelo se realizaron en un campo abierto sometido a perturbaciones externas como el viento. Se realizaron pruebas tanto para *hovering* (es decir para valores de referencia de 0 grados), y también para valores de referencia de los ángulos distintos de cero grados para ver el desempeño del controlador y se comprobó que la respuesta sigue a las magnitudes de referencia enviadas con el mando a control remoto.

En la figura 6.1 se muestra la gráfica de vuelo obtenida en una de las pruebas, en la cual se observa el despegue del Quadrotor en $t = 5s$ y luego se mantiene en *hovering* el resto del tiempo comprobando el funcionamiento del controlador. La señal de referencia se mantiene en 0 grados para los tres ejes por lo que el controlador permite mantener el sistema en esa actitud pero se observa que la inherente inestabilidad del sistema sumado a las perturbaciones generadas por el ambiente genera cierto error en el vuelo.

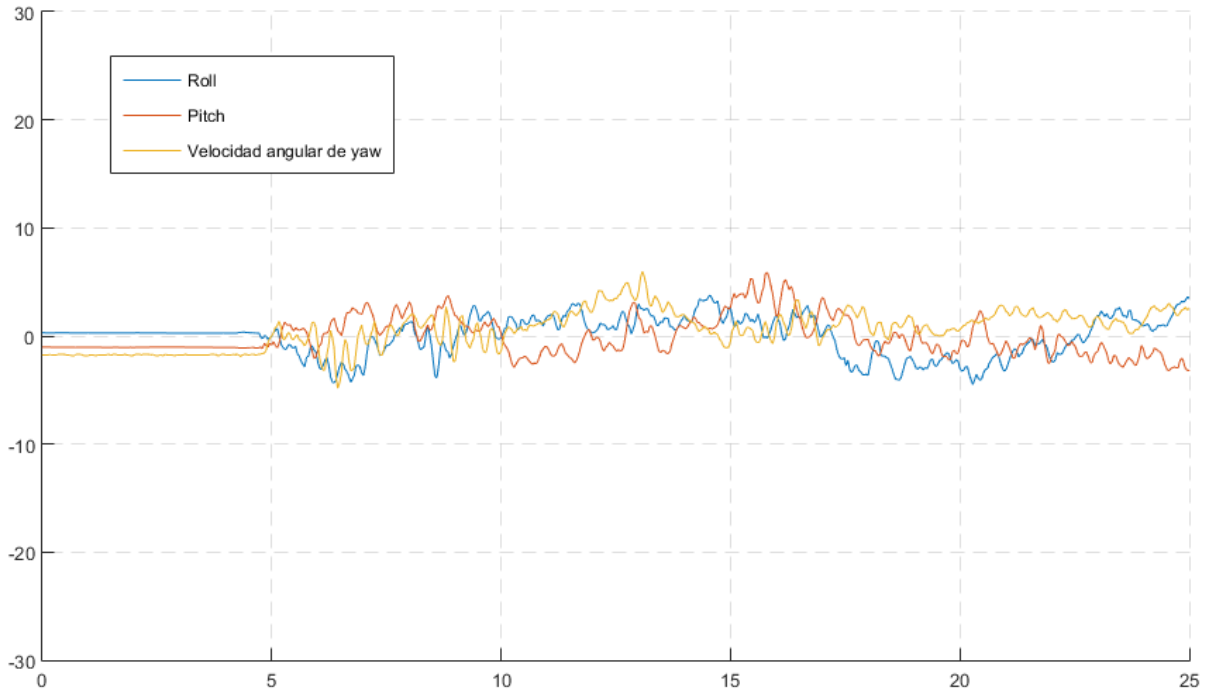


Figura 6.1: Resultados de vuelo para caso *hovering*

Para una mejor visualización se presenta las gráficas de control por separado.

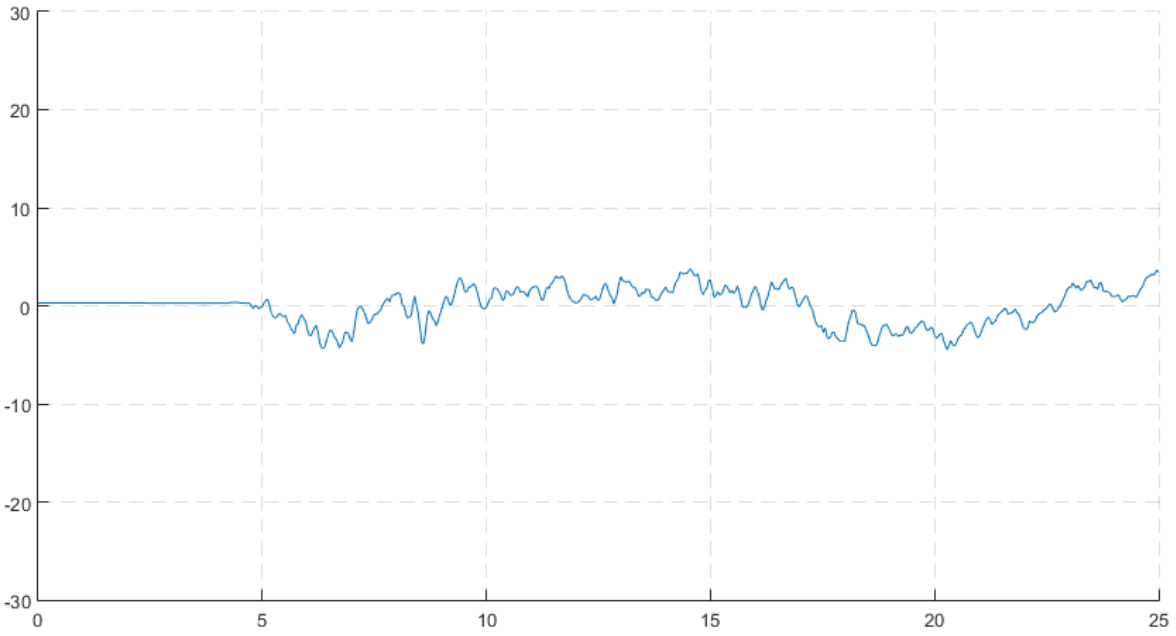


Figura 6.2: Controlador del subsistema *Roll* en *hovering*

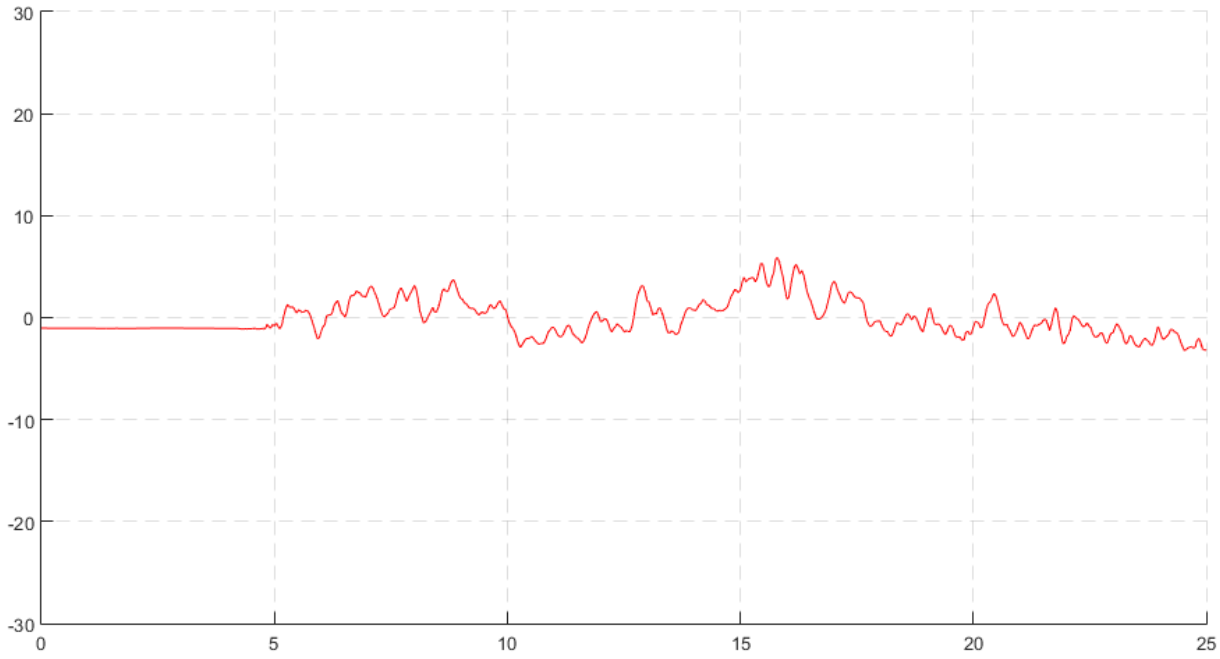


Figura 6.3: Controlador del subsistema *Pitch* en *hovering*

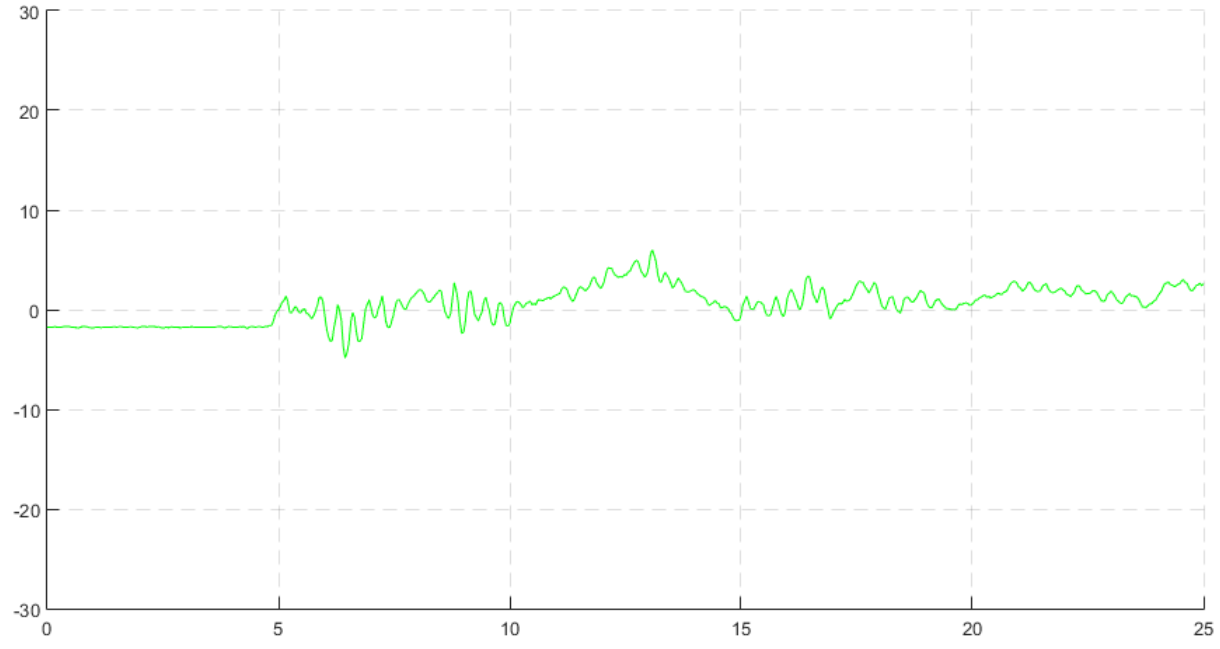


Figura 6.4: Controlador de velocidad angular del eje de *Yaw* para caso *hovering*

Para medir el margen de error en el controlador se utilizó una unidad de medición basado en el error cuadrático medio (RMS) el cual se midió para cada eje de rotación. La definición del error cuadrático medio se observa en la ecuación (6.1):

$$errorRMS = \sqrt{\frac{\sum_{i=1}^{N}(\widehat{x_i} - y_i)^2}{N}} \tag{6.1}$$

Donde:

$\widehat{x_i}$: es el valor estimado.

y_i : es el valor verdadero o referencial.

N : el número de muestras.

El error RMS medido se basó en información de 20 segundos de vuelo con una muestra de 1000 datos para cada eje de rotación. En la tabla 6.1 se observa la magnitud del error RMS para cada eje.

Tabla 6.1: Tabla de error RMS para caso hovering

Ángulo	Error RMS
Roll	2.0140
Pitch	1.9013
Yaw	1.7107

También se realizaron pruebas estableciendo cambios en la actitud y para así lograr desplazamiento en el vehículo. Si bien es cierto que el controlador está diseñado para un modelo matemático que no considera ángulos grandes en la actitud, pero se comprobó en las experiencias que ante cambios en la actitud los controladores PID se adaptan a pesar de las imperfecciones del modelo matemático y logran seguir las variables de referencia, es decir el controlador también logra corregir la actitud aun en casos distintos al *hovering* donde las variables de referencias son distintos de cero grados de inclinación. En las figuras 6.5 y 6.6 se observan el resultado de las pruebas ante diversos valores de referencias.

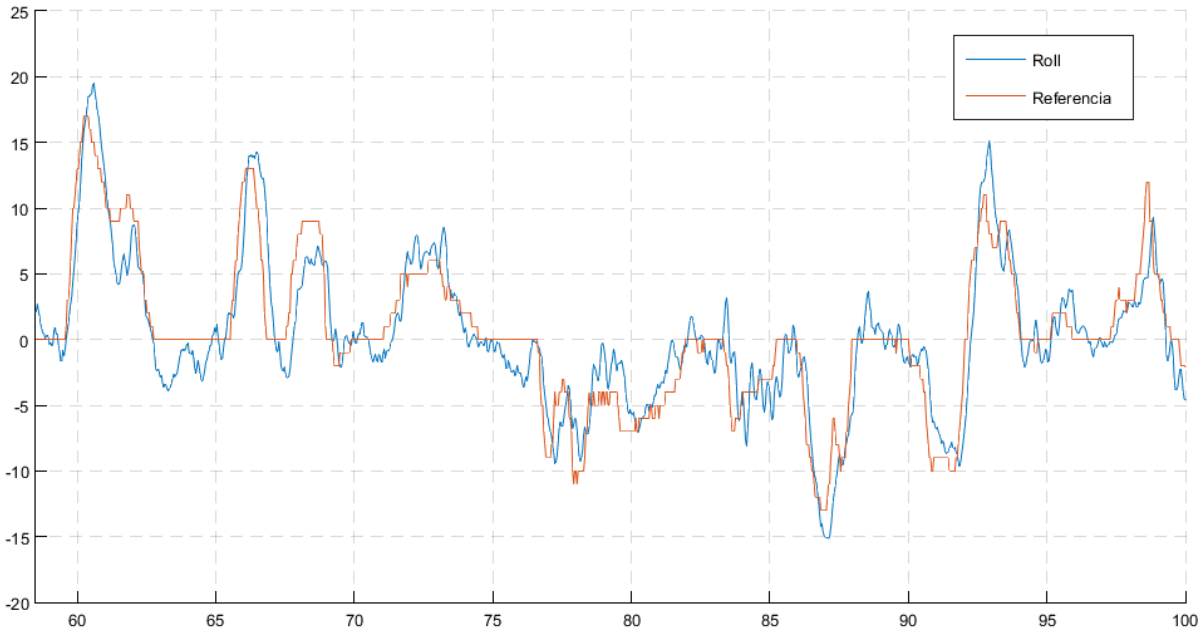


Figura 6.5: Gráfica de control del ángulo *Roll* ante diversas referencias

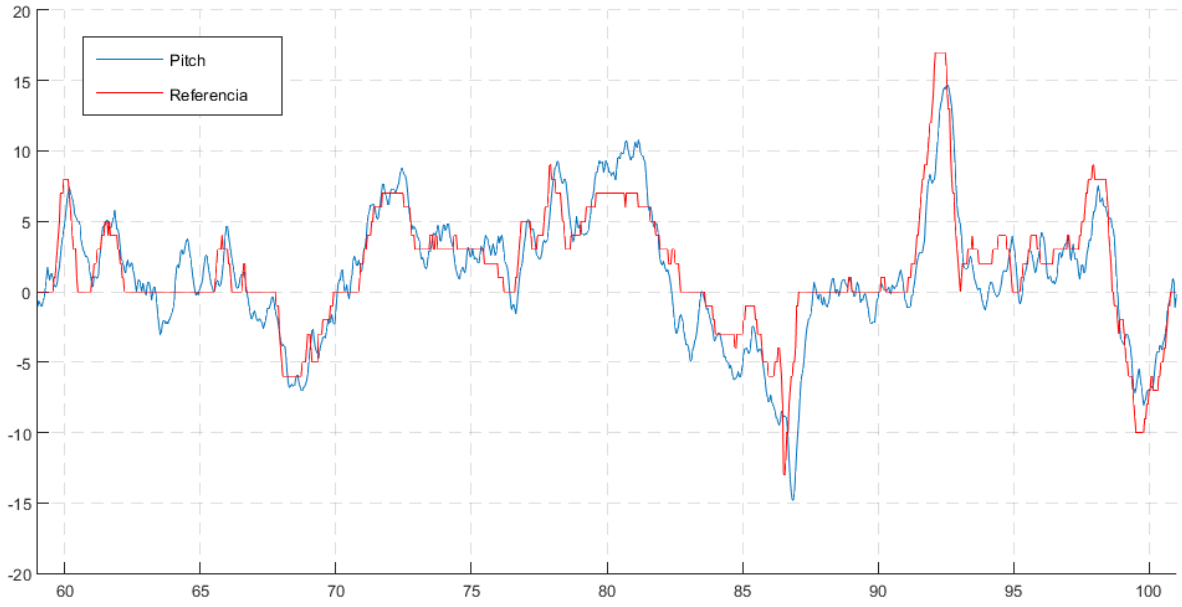


Figura 6.6: Gráfica de control del ángulo *Pitch* ante diversas referencias

En la tabla 6.2 se presenta el error RMS medido en el vuelo con señales de referencia diferentes de cero, se puede notar que el error ha aumentado de magnitud con respecto a las pruebas realizadas en *hovering* esto debido a que el controlador fue diseñado para un punto de operación en el estado de *hovering*, pero aun así los resultados presentados son aceptables. Se observa también que el error RMS en el ángulo *roll* es mayor, esto debido a que el brazo del Quadrotor que cruza el eje del ángulo *roll* lleva más pesos que el eje de *pitch* al ubicar componentes que generan perturbaciones con su peso.

Tabla 6.2: Tabla de error RMS para diversas referencias

Ángulo	Error RMS
Roll	5.51
Pitch	4.70
Yaw	2.95

6.2 Computador de vuelo

Se comprobó el buen funcionamiento del diseño hardware y software implementado en el computador de vuelo al ejecutar el sistema de control de actitud del Quadrotor en la experiencia realizada al aire libre, además que la FPGA usada como controlador tiene suficientes recursos sobrantes para implementar otras tareas de modo que el computador de vuelo puede ser mejorado para agregar aplicaciones de mayor nivel. En la figura 6.7 se observa que se utilizaron 8913 elementos lógicos reconfigurables que equivalen al 40% del total, es decir sobra suficientes recursos para realizar otras tareas específicas a nivel de hardware e incluso agregar otro procesador NIOS II si se requiere. Además solo se utilizaron 14 pines de la tarjeta de desarrollo DE0-Nano de los 80 disponibles, es decir se puede utilizar 66 pines más para conectar cualquier sensor o dispositivo que se desee controlar.

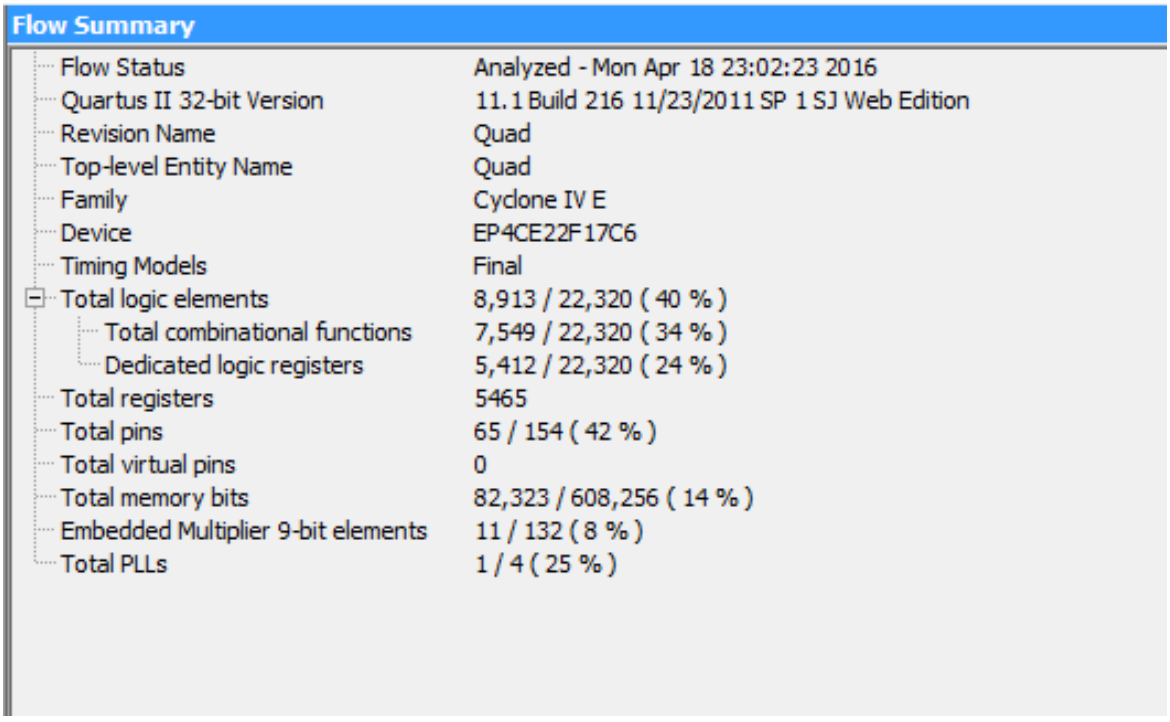


Figura 6.7: Resultados de diseño de arquitectura hardware en el computador de vuelo

6.3 Conclusiones

- Tal como se aprecia en las secciones 3.5.1 y 3.5.2 se presentó dos maneras de calcular el modelo matemático del Quadrotor, lo cual dio como resultado una función de transferencia. Además se observó que el modelo dinámico hallado experimentalmente caracteriza mejor los efectos dinámicos del Quadrotor, por lo tanto es posible utilizar este método práctico de identificación de sistemas para el diseño de controladores de vuelo en Quadrotores.
- El simulador de vuelo desarrollado en el software Matlab presentando en la sección 3.6 permitió validar el modelo matemático calculado y probar los algoritmos de control antes de realizar pruebas de vuelo real por lo cual es una herramienta vital para el desarrollo del sistema de control en este proyecto.
- El desarrollo de la arquitectura hardware y el software implementado en el computador de vuelo basado en FPGA cumplió satisfactoriamente las funciones de vuelo implementadas utilizando pocos recursos hardware, por lo cual se concluye que es una herramienta potente para el desarrollo de controladores en vehículos aéreos.
- Se comprobó el buen funcionamiento del Quadrotor en hovering con un diseño de control clásico basado en un punto de operación en hovering, se concluye entonces que es posible utilizar modelos matemáticos lineales que sean aproximados al modelo real para realizar el diseño de control en Quadrotores.
- A pesar de las limitaciones del modelo matemático que es válido para pequeños ángulos de desviación, se demostró que el Quadrotor puede ser controlado a desviaciones de ángulos grandes por lo cual se concluye que el controlador cumple satisfactoriamente su función y es posible utilizar modelos simples a pesar de imperfecciones en el modelo matemático.

6.4 Trabajos futuros

Algunos trabajos a futuro son los siguientes:

- ✓ Implementación de un control de altitud usando sensores barómetro y ultrasonido.
- ✓ Implementación de un control de trayectoria usando GPS.
- ✓ Implementación de otros algoritmos de control de actitud como el LQG para comparación de algoritmos.
- ✓ Implementar un segundo procesador que se ejecute en paralelo y realice aplicaciones de visión artificial.

BIBLIOGRAFÍA

Altera. (02 de 04 de 2015). *Nios II Gen2 Processor Reference Guide.* Recuperado el 07 de 11 de 2015, de https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/hb/nios2/n2cpu-nii5v1gen2.pdf

Altera. (s.f.). *Nios II/f*. Recuperado el 16 de 06 de 2016, de https://www.altera.com/products/processors/benefits/nios-ii-processor-cores.html#fast

ArduPilot Dev Team. (07 de 11 de 2015). Obtenido de http://ardupilot.com/ardupilot/index.html

Aström, K. J., & Murray, R. M. (2002). *Feedback Systems An Introduction for Scientists and Engineers.* Princeton University Press.

Berkeley. (s.f.). *XC2000 Logic Cell Array Family.* Recuperado el 07 de 11 de 2015, de http://www-inst.eecs.berkeley.edu/~cs294-59/fa10/resources/Xilinx-history/xc2000.pdf

Bouabdallah, S. (2011). *Design and Control of Quadrotors with Application to Autonomous Flying.* École Polytechnique Fédérale de Lausanne.

Bresciani, T. (2008). *Modelling, Identification and Control of a Quadrotor Helicopter.* Lund University.

Castro, R. G. (2001). *Sistemas Mecanicos SubActuados.* Universidad Nacional de Ingenieria.

DFRobot. (s.f.). *10 DOF Mems IMU Sensor*. Recuperado el 07 de 11 de 2015, de http://www.dfrobot.com/index.php?route=product/product&product_id=818#.V6p0Bph97IU

Digi International Inc. (s.f.). *XBee/XBee-PRO RF Modules.* Recuperado el 11 de 07 de 2015, de https://www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-Datasheet.pdf

DJI. (s.f.). *DJI Phantom 4*. Recuperado el 11 de 08 de 2016, de http://www.dji.com/product/phantom-4/info#specs

DJI. (s.f.). *FLAME WHEEL ARF KIT*. Recuperado el 09 de 08 de 2016, de http://www.dji.com/product/flame-wheel-arf/feature

DJI Wiki. (s.f.). *ESC*. Recuperado el 07 de 11 de 2015, de http://wiki.dji.com/en/index.php/ESC

Draganfly Innovations Inc. (s.f.). *Draganfly*. Recuperado el 07 de 11 de 2015, de http://www.draganfly.com/products/x4-es/overview

García, F. M. (s.f.). *Controladores PID.* Recuperado el 11 de 08 de 2016, de http://www.dia.uned.es/~fmorilla/MaterialDidactico/Aspectos%20practicos.pdf

Guilherme V. Raffo, M. G. (2008a). Backstepping/Nonlinear H¥ Control for Path Tracking of a QuadRotor. *2008 American Control Conference*, 3356-3361.

Guilherme V. Raffo, M. G. (2008b). MPC with Nonlinear H¥ Control for Path Tracking of a Quad-Rotor Helicopter. *17th International Federation of Automatic Control*, 8564-8569.

Hirschberg, M. J. (2000). *The American Helicopter an Overview of Helicopter Developments in America 1908-1999.* Anser Analytic Services.

Hobbyking. (s.f.). *Turnigy 4500mAh*. Recuperado el 07 de 11 de 2015, de http://www.hobbyking.com/hobbyking/store/__10283__Turnigy_4500mAh_3S_30C_Lipo_Pack.html

Huerta, P. (2009). *Sistemas de Multiprocesamiento Simétrico Sobre FPGA.* Universidad Rey Juan Carlos.

Huerta, R. (2015). *Implementación de un Sistema de Determinación de Orientación Mediante Diseño SOPC en un FPGA para un Vehículo Aéreo No Tripulado del Tipo Quadrotor.* Universidad Tecnologica del Peru.

I. Kuon, R. T. (2008). *FPGA Architecture: Survey and Challenges.* now Publishers Inc.

Jaramillo, F., & Gómez, A. (2013). *Sistema de Control para la Estabilidad y Orientación de un helicóptero Quadrotor.* Escuela de Ingeniería de Antioquia.

Jorge M. Cotter, A. F. (2010). *Diseño de Control Robusto de Velocidad de Motores Brushless para Robótica Aérea.* Universidad Nacional de Colombia.

Kharsansky, A. (2013). *Diseño e Implementación de un Sistema Embebido de Control de Actitud para Aeronaves No Tripuladas.* Universidad de Buenos Aires.

Leishman, J. G. (2000). *A History of Helicopter.* University of Maryland.

MathWorks Inc. (s.f.). *PID Tuning*. Recuperado el 11 de 08 de 2016, de http://www.mathworks.com/discovery/pid-tuning.html

MathWorks Inc. (s.f.). *System Identification Toolbox*. Recuperado el 19 de 10 de 2015, de http://www.mathworks.com/help/ident/index.html?requestedDomain=www.mathworks.com

MIT. (s.f.). *Aerospace Control Laboratory*. Obtenido de http://acl.mit.edu/

Muñoz, M. A. (s.f.). *Manual de vuelo*. Recuperado el 23 de 12 de 2015, de http://www.manualvuelo.com/PBV/PBV12.html

Najib Metni, T. H. (2007). Visual Tracking Control of Aerial Robotic Systems with Adaptive Depth Estimation. *International Journal of Control, Automation, and Systems*, 51-60.

Nonami, K., Kendoul, F., Suzuki, S., Wang, W., & Nakazawa, D. (2010). *Autonomous Flying Robots.* Springer.

Parrot SA. (s.f.). *Parrot AR Drone*. Recuperado el 07 de 11 de 2015, de http://www.parrot.com/usa/products/ardrone-2/

Risco, M. (2014). *Arquitectura de Bus Simple, un Conjunto de Herramientas para el Desarrollo Portable de Sistemas en Chip.* Recuperado el 10 de 06 de 2016, de https://drive.google.com/file/d/0B4ddOwd1tkyZYmthVUd4RnpHWmc/edit

S. Bouabdallah, A. N. (2004b). PID vs LQ Control Techniques Applied to an Indoor Micro Quadrotor. *IEEE International Confererence on Intelligent Robots and Systems, 2*, 2451–2456.

S. Bouabdallah, P. M. (2004a). Design and Control of an Indoor Micro to Indoor Micro Quadrotor. *IEEE International Conference on Robotics and Automation*, 4393–4398.

S. Bouabdallah, R. S. (2005). Backstepping and Sliding-mode Techniques Applied to an Indoor Micro Quadrotor. *IEEE International Conferecence on Robotics and Automation*, 2259-2264.

Stanculeanu, I., & Borangiu, T. (2011). Quadrotor Black-Box System Identification. *International Journal of Mechanical, Aerospace, Industrial, Mechatronic and Manufacturing Engineering*, 1025-1028.

Szafranski, G., & Czyba, R. (2011). Different Approaches of PID Control UAV Type Quadrotor. *Proceedings of the International Micro Air Vehicles conference 2011 summer edition*, 70-75.

Terasic. (2012). DE0-Nano User Manual. Recuperado el 07 de 11 de 2015, de http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&No=593&PartNo=4

Terasic. (s.f.). *DE0-Nano Development and Education Board*. Recuperado el 07 de 11 de 2015, de http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=139&No=593

University of Pennsylvania. (s.f.). *GRASP - general robotics, automation, sensing and perception laboratory*. Obtenido de https://www.grasp.upenn.edu/

Zurich, ETH. (s.f.). *Flying Machine Area*. Obtenido de http://flyingmachinearena.org/