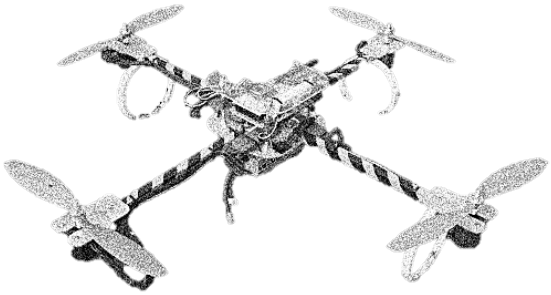
UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

CONSTRUCCIÓN Y CONTROL DE ESTABILIDAD DE UN CUADRICÓPTERO



TESIS

QUE PARA OBTENER EL TÍTULO DE

INGENIERO MECATRÓNICO

PRESENTA

LUIS ANTONIO JUÁREZ GARCÍA DE LEÓN

DIRECTOR DE TESIS

M. EN I. SERAFÍN CASTAÑEDA CEDEÑO

ENERO 2012

-Esta es la Tierra. No el eterno hogar de la Humanidad,
sino el punto de partida de una infinita aventura.
Todo lo que has de hacer para conseguirlo es tomar tu decisión.
Es sólo tuya. (...) Con aquella desaparición,
(...) había llegado el fin de la Eternidad... Y el comienzo del Infinito.
Isaac Asimov, "El fin de la Eternidad"

Para Mamá, Titila y Apá

Agradecimientos

Gracias a Dios por la oportunidad de la vida con todo lo que implica vivir y por haberme dado el honor de haberlos conocido y compartido tanta vida.

A mi Mamá por haber roto las reglas sociales, por haber seguido a tu corazón abanderada por el amor, por aventurarte a convertirte en mi Mamá; gracias por el amor, las enseñanzas y la confianza que desde siempre pusiste en mí. Gracias por tus esfuerzos, por tus horas, tus consejos, por las palabras de aliento, por la paciencia y por todo lo que en palabras sería motivo de un gran documental con muchos capítulos en los que siempre estarías de protagonista y aun así insuficiente para intentar expresar mi agradecimiento y mi amor por ti, porque gracias a ti soy y soy lo que soy.

A mi Titila, porque a pesar de las circunstancias de tu tiempo y tu historia, no dudaste en convertirte en mi abuelita, porque preparaste el camino, porque aún sin conocerme, cumpliste de manera excepcional el papel, más allá del protocolo, pero también por el asunto de ser mi segunda mamá; gracias por los consejos, las correcciones, la formación, las historias, tu historia. Gracias también por el apoyo, por las ideas, por la complicidad, la amistad, por estar ahí. Gracias porque sabes también que sin tu ayuda estelar, esto que leemos hoy no sería, gracias por el ejemplo de fortaleza y de vida que das a todos los que te conocemos.

A mi Apá, porque aún en la distancia, de alguna u otra forma siempre estuviste tan cerca y tan al pendiente. Gracias por el esfuerzo y por el amor que a pesar de tu trayectoria en esta vida, de tu tiempo, nunca disminuiste para con los tuyos. Porque para mí, tú también eres un ejemplo de profunda nobleza, de amistad y de compañía, detrás de ese carácter recio. Gracias por siempre estar ahí.

A usted Tío, porque desde otro tiempo y otra vida, sembraste las semillas del ejemplo de superación, de esas que contienen árboles fuertes y de mucho conocimiento, que rompen barreras y esquemas sociales por el simple gusto de crecer; porque con ella compartiste el tiempo difícil de la austeridad y la pobreza que los hizo grandes personas.

A mi Familia, la de lejos, la de mientras, la de un rato, la que está ahí sin convivir demasiado, pero que en algún momento compartieron tiempo conmigo y dejaron algún recuerdo y consejo como base de lo que soy.

A ti Papá, porque también accediste al amor, porque también te aventuraste a convertirte en mi Papá, y aunque no tuvimos mucho tiempo para compartir, sé que desde donde estás, participaste en el trazo del camino que me trajo hasta aquí, porque sé que lo seguirás haciendo hasta que nos volvamos a encontrar, porque sin ti, nada de esto hubiera ocurrido.

Quisiera compartirles lo que alguna vez leí, porque "A veces llegué a pensar que en el banco de niños soborné a algún ángel de poca vocación para que me brindara el milagro de poder ser hijo de alguien tan extraordinario como vos. De lo contrario, cómo explicar tantísima suerte... El otro día que a mí me toque, espero encontrarme al mismo ángel sin vocación, para sobornarlo de nuevo a cualquier precio y me dé la posibilidad de volverme a encontrar contigo, donde quiera que sea."... con ustedes.

A mis maestros de la vida, los de cada etapa, los que corrigieron mis primeros errores, los que me levantaron de mis tropiezos, los que me enseñaron lo necesario para llegar, los que me alentaron, incluso los que me enseñaron lo que no quería ser. Todos ustedes que confiaron, que me brindaron la oportunidad de comunicarme con el mundo y de poder recibir lo que otros querían comunicarme, en especial, gracias a mis maestros y maestras de las mejores etapas, aquellos de quienes guardo muy gratos recuerdos.

A mis profesores de la Facultad de Ingeniería, los Ingenieros, los que aportaron su valioso conocimiento de este arte de transformar el mundo, de conocerlo, de intentar explicarlo, de mejorarlo. A mi director de tesis, el Maestro Serafín por el apoyo y la fe que desde un inicio tuvo en este proyecto, gracias por el apoyo teórico, moral y económico. A mis sinodales por su conocimiento compartido, su paciencia, por su apoyo y sus palabras, por brindarme el honor de poder entregarles este trabajo.

A mis amigos, los de siempre, los incondicionales, los que soportan mi persona, los de los consejos y los momentos de alegría. Los que aclaran el rumbo cuando hay confusión. Los críticos de buena fe, los apoyos.

A las personas que de una u otra forma influyeron en alguna parte de mi vida, las que no menciono pero no olvido, las que detrás del telón, aportaron algo que me hiciera lo que soy. Las que estuvieron muy cerca, por mucho tiempo, las que me regalaron sonrisas y por qué no, también con las que sufrimos. Las personas que siempre querré pase lo que pase.

A todos los que han creído y apoyado de alguna manera mi proyecto de vida, la tesis, mi persona; a todos los demás que no menciono pero no olvido, por lo que dejaron en mí, por lo que aportaron para construirme así. A todos, de sinceramente, muchas gracias.

ToÑo JuÁrEz, México

Abstract

A quadcopter is a four-rotor aircraft in a cross configuration. It is a mechatronic device with many possibilities of study from the standpoint of control. It has a wide variety of applications ranging from entertainment, exploration and aerial photography, to the application of various control techniques as result of research and technology development, construction of simple structures and autonomous tours with real time video streaming. The main variables of a quadcopter are three navigation angles (α, β, γ) and three coordinates (x, y, z) that place it in a position and orientation relative to an inertial reference frame. In this paper, a quadcopter was built based on a design process. The characteristics of the physical model were used to determine the parameters of a mathematical model that takes certain considerations as that the operation point is close to the equilibrium point. Based on this mathematical model a PID control was designed and implemented to achieve angular stability of the physical model. Several tests were conducted at different constants and satisfactory results were obtained. The quadcopter was stable and floating in a fixed point. Then a state feedback control was designed, this one wasn't implemented, but it was simulated to observe the behavior of the quadcopter of the stability, performance and regulation of its states, including linear positions and velocities and angular positions and velocities. As a result of the simulation, several graphs were obtained showing the behavior of the angles, the system inputs and the quadcopter positions in response to certain references which initially were equal to zero to show the stability and performance and then were different from zero to show regulation.

Resumen

Un cuadricóptero es una aeronave de cuatro rotores en una configuración en cruz. Es un dispositivo mecatrónico con muchas posibilidades de estudio desde el punto de vista de control. Tiene una amplia variedad de aplicaciones que van, desde el entretenimiento, la exploración remota y fotografía aérea, hasta la aplicación de diversas técnicas de control como resultado de investigaciones y desarrollo tecnológico, construcción de estructuras simples y recorridos autónomos con transmisión de video en tiempo real. El cuadricóptero tiene como variables principales, tres ángulos de navegación (α, β, γ) y tres coordenadas (x, y, z) que lo ubican en una posición y orientación con respecto a una referencia inercial. En el presente trabajo, se construyó un cuadricóptero con base en un proceso de diseño. Las características del modelo físico sirvieron para determinar los parámetros de un modelo matemático que toma ciertas consideraciones como que el punto de operación es cercano al punto de equilibrio. Con base en el modelo matemático anterior, se diseñó e implementó un control PID para alcanzar la estabilidad angular del modelo físico. Se realizaron varias pruebas con diferentes constantes y se obtuvieron resultados satisfactorios, es decir, que el cuadricóptero se mantenía estable y flotando en un punto fijo. Posteriormente se diseñó un control por realimentación de estados, el cual no se implementó, pero se simuló para observar su comportamiento para la estabilidad, desempeño y regulación de los estados del cuadricóptero que incluían las posiciones y velocidades lineales y las posiciones y velocidades angulares. Como resultado de la simulación, se obtuvieron diversas gráficas que muestran el comportamiento de los ángulos, las entradas del sistema y las posiciones del cuadricóptero como respuesta a referencias que en un inicio eran iguales a cero para mostrar la estabilidad y desempeño y posteriormente eran diferentes de cero para mostrar regulación.

Contenido

Figuras

Símbolos y Unidades

Cd	Constante de descarga para batería
S	Número de celdas en serie para una batería tipo Polímero de Litio
kv	Constante que relaciona velocidad angular con voltaje
Hz	Hercios
MHz	Megahercios
Amin	Amperios minuto
mA	Miliamperios
mAh	Miliamperios hora
V	Voltios
N	Newtons
hp	Horse power
g	Gramos
kg	Kiligramos
°	Grado (Ángulo)
km	Kilómetros
m	Metros
mm	Milímetros
pulg	Pulgadas
rad	Radianes
Ø	Diámetro
A	Amperios
$\frac{m}{s}$	Metro en cada segundo
°/s	Grado en cada segundo
rpm	Revoluciones por minuto
$\frac{m}{s^2}$	Metro en cada segundo cuadrado
$°/s^2$	Grado en cada segundo cuadrado

μs Microsegundos

h Horas

min Minutos

ms Milisegundos

s Segundos

CAD Computer asisted drawing

DMIPS Dhrystone Million of Instructions Per Second

DSP Digital Signal Processor

GPS Geo-Positioning System

IMU Inertial Measurement Unit

LiPoly Lithium Polymer

MIPS Millones de instrucciones por segundo

PID Control proporcional, integral y derivativo

PVC Polyvinyl chloride

PWM Pulse Width Modulation

Rotor Conjunto motor-hélice

Capítulo 1

Introducción

1.1. Antecedentes

1.1.1. Historia

El origen de la idea de mantener un objeto en el aire debido a una fuerza de empuje generada por la rotación de unas alas o hélices se remonta hasta la antigua cultura China, aproximadamente en el año 400 a. C. En ese entonces, el artefacto se basaba en un palillo con un tipo de hélice en el extremo superior, colocada perpendicularmente; éste se hacía girar con las manos y era liberado.

Mucho tiempo después, en el Siglo XV, fue Leonardo Da Vinci quien realizó varios bocetos de máquinas diseñadas para vuelos verticales utilizando alas basadas en la función de un tornillo [1]. La palabra helicóptero, se forma a partir de las raíces "*helico*" que significa espiral y "*-'ptero*" que significa ala y se define según la Real Academia Española como una aeronave más pesada que el aire y que, a diferencia del avión, se sostiene merced a una hélice de eje aproximadamente vertical movida por un motor, lo cual le permite elevarse y descender verticalmente [2].

Algunos de los primeros helicópteros de cuatro rotores se inventaron en la década de 1900. En Francia, en 1904, apareció el Giro Plano 1 Breguet-Ritchet, el cual tenía cuatro rotores con cuatro aspas biplanas cada uno (los rotores tenían 8 m de diámetro, peso neto de 580 kg y un motor Antoinette de 45 hp). Realizó un vuelo cautivo con un pasajero a una altitud de cerca de 1 m por aproximadamente un minuto. También en Francia pero en 1924, el Ingeniero Etienne Oemichen, diseñó y cons-truyó la máquina mostrada en la Figura 1.1 con cuatro rotores con dos aspas cada uno para proporcionar sustentación (dos rotores de 7.6 m de diámetro y dos de 6.4 m de diámetro), cinco hélices horizontales para el control de posición, dos hélices para propulsión y una hélice frontal para el control de inclinación; todo esto impulsado por un motor LeRhone de 120 hp.

Este modelo impuso el primer récord de distancia recorrida con 360 m. Sin embargo, los diseños de uno y dos rotores resultaron ser más eficientes contemplando factores de

carga soportada, peso, distancia, tamaño y consumo de energía, ya que los diseños se centraron en resolver las necesidades de transporte de pasajeros y productos, especialmente durante las guerras mundiales; así pues, la invención del helicóptero puede considerarse como terminada alrededor de la década de 1950 [1].

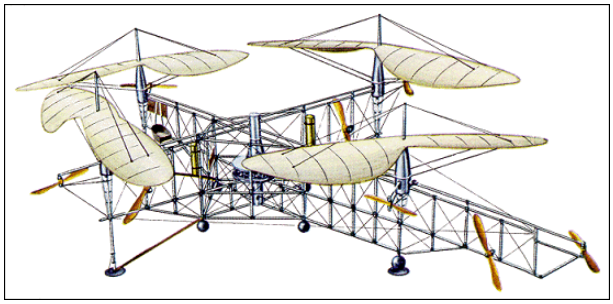


Figura 1.1: Oemichen 2

1.1.2. Definición

Recientemente, los robots aéreos cuatrirotor (*quadrotor aircraft*), los cuales forman parte de los vehículos aéreos no tripulados (*UAV* por sus siglas en inglés), o como serán mencionados en el presente trabajo: cuadricópteros; han emergido como una popular plataforma de investigación debido a la simplicidad de su construcción y mantenimiento, su habilidad para permanecer suspendidos en el aire y su capacidad de despegue y aterrizaje vertical. El cuadricóptero es un sistema mecatrónico, compuesto de una estructura mecánica, componentes electrónicos, programación y control. Utiliza cuatro motores en total, en una configuración en cruz. También cuenta con dos pares de hélices de paso fijo con rotación contraria ubicadas en las cuatro esquinas de la aeronave tal como se muestra en la Figura 1.2. Así, mientras que los rotores 1 y 3 giran en sentido antihorario, los rotores 2 y 4 giran en sentido horario lo que cancela casi en su totalidad el efecto giroscópico presentado en los helicópteros comunes pero que es utilizado en el cuadricóptero para cambiar su orientación.

Variando la velocidad angular de cada par de rotores, se puede cambiar la fuerza de empuje y se puede generar también, un movimiento vertical, frontal o lateral. El movimiento de cabeceo (*pitch*) es generado por una diferencia de velocidades entre el rotor 1 y 3 originando el ángulo β, de manera similar, el movimiento de alabeo (*roll*) se genera a partir de la diferencia de velocidades entre el rotor 2 y 4 originando el ángulo α. Tomando en cuenta el efecto giroscópico antes mencionado, el movimiento de guiño (*yaw*) es el que resulta de la diferencia de velocidades entre el par de rotores que giran en sentido horario y el par de rotores que gira en sentido antihorario originando el ángulo γ. La fuerza de empuje total es igual a la suma de las fuerzas de empuje de cada uno de los cuatro rotores. Los cuadricópteros presentan dos ventajas principales sobre los helicópteros convencionales. Primero, los cuadricópteros no requieren de vínculos mecánicos complejos entre el

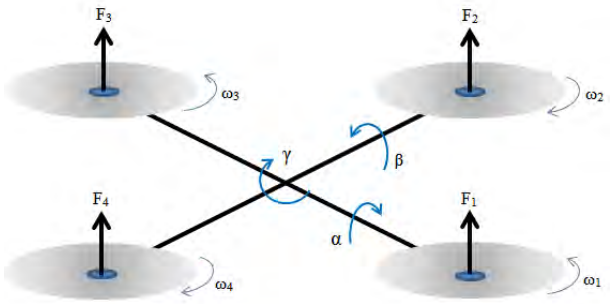


Figura 1.2: Configuración de un cuadricóptero

motor y las hélices, ya que estas últimas se acoplan directamente al eje de cada motor, simplificando así el diseño y el mantenimiento de la unidad. Segundo, el uso de cuatro rotores, asegura que cada uno de éstos es más pequeño en diámetro que el rotor principal equivalente en un helicóptero común con respecto al tamaño de la unidad. Sin embargo, algunas de las razones por las que no se cuentan con cuadricópteros del tamaño de los helicópteros actuales como los que tienen la finalidad de transportar pasajeros o mercancías son, por ejemplo, las longitudes de brazo, de la estructura mecánica y el suministro de combustible que se requeriría para actuar los cuatro rotores.

1.1.3. Desarrollos y aplicaciones

Los cuadricópteros tienen gran área de aplicación, sobre todo en exploración autónoma especialmente en ambientes peligrosos o de difícil acceso, vigilancia aérea local, exploración militar, aplicaciones comerciales ya sea en interiores o exteriores, grabación de video y toma de fotografías aéreas, apoyo a instituciones de seguridad pública [3]; incluso para organizar grupos o cuadrillas para la construcción de estructuras mecánicas simples de forma automática [4]. Estas aplicaciones han conducido a que los desarrolladores de este tipo de tecnología, tengan como metas principales el hacer dispositivos pequeños, estables, de tiempo de vuelo cada vez mayor, energéticamente más eficientes y que puedan llevar cada vez más carga. Sin embargo, algunos de los factores que mantienen en desarrollo e investigación este tipo de vehículos, son: el control de estabilidad y el seguimiento de trayectorias a partir de modelos completamente definidos, los cuales suelen ser no lineales y con parámetros desconocidos, de difícil obtención y/o despreciados para ciertos puntos de operación [5], así como el corto tiempo de vuelo que proporcionan las tecnologías actuales de suministro de energía por baterías [6, 3], lo que limita en cierta medida el comportamiento y las tareas que puede realizar un cuadricóptero. Es gracias a los avances en las capacidades de los microprocesadores, sistemas de control moderno, tecnología de motores y en los sistemas de sensores ambientales e inerciales que es posible encontrar de manera comercial dispositivos cuatrirotores tan avanzados como el Draganflyer® [7].

Figura 1.3: El cuadricóptero

Algunos cuadricópteros, incluyen sistemas avanzados de control de estabilidad y seguimiento de trayectorias para mejorar el aprendizaje y la experiencia de uso para pilotos tanto expertos como novatos. Actualmente entre las diversas técnicas de control aplicadas a este tipo de vehículos, están el control *backstepping* [8, 9], H∞ [10], control por sistemas de visión [11], control PID (*Proportional Integral Derivative*) [6, 12], redes neuronales [13] y sistemas RBF-ARX (*Radial Basis Function-Agressive External*) [5], entre otros. Algunos de los sensores más comunes que se han empleado en estos sistemas, son las Unidades Medidoras de Inercia (IMU, *Inertial Measurement Unit*) que integran acelerómetros, giró metros y magnetómetros de forma cada vez más compacta y con mayor sensibilidad; además, están también los sistemas de geoposicionamiento (GPS, *Geo-Positioning System*), las videocámaras, sonares y sensores láser entre otros. Así pues, el cuadricóptero es un sistema con muchas posibilidades de estudio desde el punto de vista de control; es un sistema rico en análisis teórico y aplicación de diversas técnicas de control tanto lineal como no lineal, clásico como moderno, así como un sistema con gran capacidad de instrumentación.

1.2. Objetivos

Después de describir el amplio panorama que abarca la construcción y el control de un cuadricóptero, se definen a continuación, los alcances y objetivos generales de este tipo de sistemas, además de limitar los objetivos particulares del presente trabajo de investigación.

1.2.1. Objetivos generales

Desde un punto de vista de versatilidad y funcionalidad, el cuadricóptero puede cumplir varios objetivos que comienzan con la construcción del mismo; pasando por otros como el de la capacidad de flotar en un punto fijo, el vuelo por radiocontrol, la toma de videos y fotografías aéreas de forma manual o automática, la recopilación de información ambiental

como temperatura, humedad y presión atmosférica; hasta la transmisión de video en tiempo real a la base de operación, transporte de pequeños objetos de manera autónoma e incluso el seguimiento de trayectorias definidas preprogramadas para la realización de recorridos autónomos o tareas y maniobras específicas.

Desde un punto de vista de control, para un cuadricóptero pueden existir tres objetivos generales principales:

1. Estabilidad angular y lineal del sistema.
2. Regulación a una referencia constante diferente de cero (velocidades lineales).
3. Seguimiento de una referencia variable (perfil de velocidades).

1.2.2. Objetivos Particulares

El objetivo del presente trabajo, es buscar el control de la estabilidad de un cuadricóptero con una referencia constante igual a cero. Para cumplir esto, primero es necesario diseñar y construir un prototipo que permita definir los parámetros y constantes requeridas para el desarrollo del controlador. Posteriormente, con los datos anteriores se diseñará y simulará una propuesta de control y se implementará en el prototipo para realizar algunas pruebas de funcionamiento.

Capítulo 2

Características, funciones y especificaciones

Como se mencionó en los objetivos tanto generales como en los particulares, es necesario basar el desarrollo del control en los parámetros definidos por un prototipo, por lo que para este trabajo, se decidió diseñarlo y construirlo. Esto a su vez requirió de seguir un breve proceso de diseño que incluyó entre sus etapas, la definición de las características, funciones y especificaciones del prototipo a construir.

2.1. Características deseadas

El proceso de diseño que se siguió, comenzó con la definición de las siguientes características deseadas del cuadricóptero:

1. El cuadricóptero vuela.
2. El cuadricóptero tiene control ante perturbaciones como el viento, es decir, mantiene su estabilidad.
3. El usuario conoce la información que el cuadricóptero envía.
4. El cuadricóptero mantiene fija una posición mientras no se le indique algún movimiento.
5. El cuadricóptero despega y aterriza de manera autónoma cuando el usuario lo solicite.
6. El cuadricóptero utiliza baterías recargables.
7. El cuadricóptero detecta la carga mínima de las baterías requerida para mantener el vuelo.
8. El cuadricóptero se transporta fácilmente.
9. La capacidad de carga del cuadricóptero permite variaciones en el peso y en el número de componentes a bordo.

2.2. Funciones

Siguiendo el proceso de diseño, se identificaron las siguiente funciones del cuadricóptero, separando así el sistema completo en sub-conjuntos de menor complejidad, sin perder de vista la correlación de estos últimos para definir y resolver el primero. Además, se organizaron de manera que se pudiera visualizar dicha correlación con respecto al flujo de información y energía, como se muestra en la Figura 2.1, donde como primera aproximación se tomaron en cuenta algunos componentes comerciales para conocer los requerimientos de energía y protocolos de comunicación que más adelante, definieron la selección final de dispositivos del cuadricóptero.

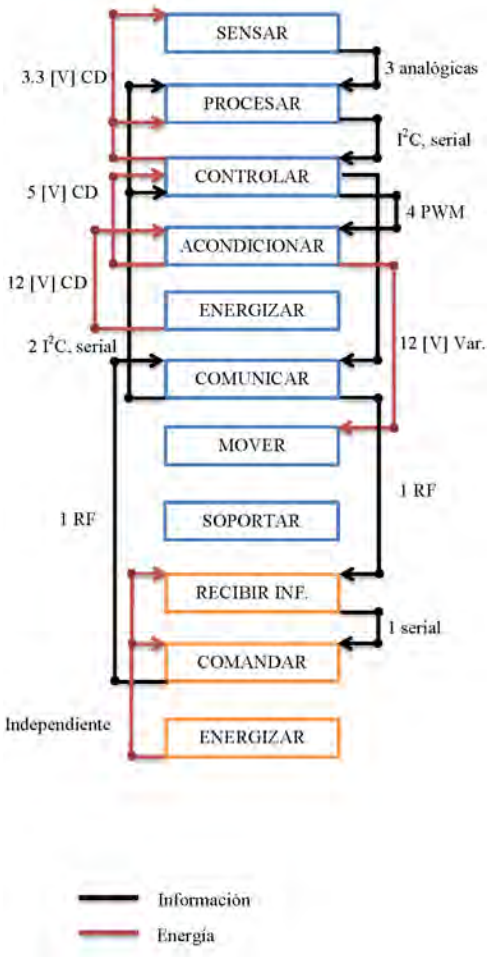


Figura 2.1: Interacción de funciones

Algunos de los objetivos de cada función se presentan a continuación:

- SENSAR (velocidad, posición)
- PROCESAR (información de sensores, control, comunicación)
- CONTROLAR (actuadores)

- ACONDICIONAR (señales de control, alimentación para electrónica)
- ENERGIZAR (actuadores, etapa de potencia, electrónica)
- COMUNICAR EN VUELO (al cuadricóptero con la base)
- MOVER (el cuadricóptero a las posiciones deseadas)
- SOPORTAR (los componentes del cuadricóptero)
- COMUNICAR EN BASE (a la base con el cuadricóptero)
- COMANDAR (generar comandos de control)
- ENERGIZAR (los componentes de la base)

2.3. Especificaciones

Después de haber definido las características deseadas, así como haber identificado las funciones del cuadricóptero, se seleccionaron las unidades físicas que caracterizan los elementos o dispositivos con los que debe contar el cuadricóptero; esta selección se encuentra en el Cuadro 2.1, donde se muestran las características deseadas y una o varias magnitudes físicas que permiten especificarla.

Cuadro 2.1: Características y especificaciones

Característica deseada	Especificación
Capacidad de Volar	Empuje vertical total mínimo de 39.2 N, posición entre 0 m y 200 m y tiempo de vuelo de 5 min.
Mantener horizontalidad, estabilidad y posición	Posición lineal fija y angular de $\pm 10°$, velocidad lineal y angular de 0 $^{m}/_{s}$ y 0 $^{\circ}/_{s}$ respectivamente y aceleración lineal y angular de 0 $^{m}/_{s^2}$ y $^{\circ}/_{s^2}$ respectivamente.
Envío y recepción de información	Alcance de 5 km
Uso de baterías	De 12 V, amperaje necesario para mantener el tiempo de vuelo
Aterrizaje y despegue autónomo	Menos de 5 intervenciones del usuario
Fácil transporte y capacidad de carga	Longitud de 0.25 m entre el centro del cuadricóptero y el eje del motor y peso máximo de 4000 g

Después de haber encontrado parámetros medibles que especifiquen al cuadricóptero, se procedió a buscar elementos como materiales, sensores, actuadores y procesadores, entre otros, que cubran las funciones de acuerdo a las especificaciones.

Capítulo 3

Selección de componentes electrónicos

Para llevar a cabo la construcción de un cuadricóptero con base en las funciones definidas anteriormente, es necesario contar con elementos físicos reales que permitan dar solución a dichas funciones. Por lo tanto, para poder sensar, controlar, procesar, acondicionar, energizar, comunicar y mover al cuadricóptero, sus datos y señales, se requiere definir actuadores, baterías, sensores, procesadores, controladores y dispositivos de comunicación como se detalla a continuación.

3.1. Actuadores

Con base en el peso de algunos cuadricópteros existentes como en [14, 6, 15] que va desde los 0.56 kg hasta los 4.34 kg, se determinó que el peso final del cuadricóptero podría oscilar cerca de los 3.5 kg. Con esta primera aproximación, se buscaron actuadores que pudieran proporcionar como mínimo el empuje necesario para mantener flotando en el aire al cuadricóptero, pero también el empuje necesario para acelerarlo aproximadamente dos veces su propio peso, por lo que se necesitaron actuadores que lograran levantar, entre los cuatro, una masa total de entre 6 kg y 7 kg.

Recordando la configuración de fuerzas de la Figura 1.2, cada rotor debería producir un empuje igual a un cuarto del peso total del cuadricóptero, así, si el peso total máximo son 7 kg, cada rotor debe producir 1.75 kg o 17.15 N de empuje vertical.

En el aeromodelismo, que es la actividad más relacionada con la construcción de un cuadricóptero, se ocupan, entre otros, motores del tipo "sin escobillas" o *brushless* en inglés [6, 15], de los cuales existen 2 subcategorías, los *inrunner* y los *outunner*. Los *inrunner*, están diseñados para altas velocidades (más de 10,000 rpm) y bajos torques (empujes menores a 15 N) y su funcionamiento es similar a los motores comunes de corriente directa (CD), en donde la parte rotatoria (con los imanes permanentes) junto con el eje está dentro de la carcasa del motor.

Los motores *outrunner* en cambio, están diseñados para bajas velocidades (me-

nos de 10,000 rpm) y altos torques (empujes mayores a 15 N), ya que la configuración de rotación es inversa a la de los *inrunner*, es decir, la parte rotatoria (con los imanes permanentes) junto con el eje, es la carcasa del motor [16].

Dado lo anterior y tomando en cuenta el empuje vertical necesario para cada rotor, se decidió que la fuerza de empuje del cuadricóptero fuera generada por los motores brushless outrunner KD 36-10XL que tienen por características: 900 kv, 18 V máximos de alimentación, 3 fases, 42 A de consumo de corriente máximo y 190 g de peso.

Figura 3.1: Motor KD 36-10-XL

Las hélices se seleccionaron de acuerdo a la configuración del giro de los rotores de un cuadricóptero en donde dos giran en sentido horario y dos en sentido antihorario pero las cuatro generan un empuje vertical, por lo que fue necesario contar con dos tipos de hélices: *Flyer* y *Pusher* como las que se muestran en la Figura 3.2.

Figura 3.2: Hélices Flyer y Pusher

El motor seleccionado, incluía una recomendación de hélices de 13 pulg de diámetro por 6.5 pulg de paso, sin embargo, comercialmente, se encontraron hélices de 14 pulg x 4.7 pulg y 12 pulg x 3.8 pulg que son las hélices usadas en el cuadricóptero. En conjunto, una hélice de 12 pulg x 3.8 pulg y un motor de los seleccionados, alcanzan, según pruebas

realizadas, un empuje de aproximadamente 2.8 kg o 27.44 N cuando éstos giran a cerca de 10,000 rpm. Cada motor junto con su respectiva hélice, tiene un peso de 200 g.

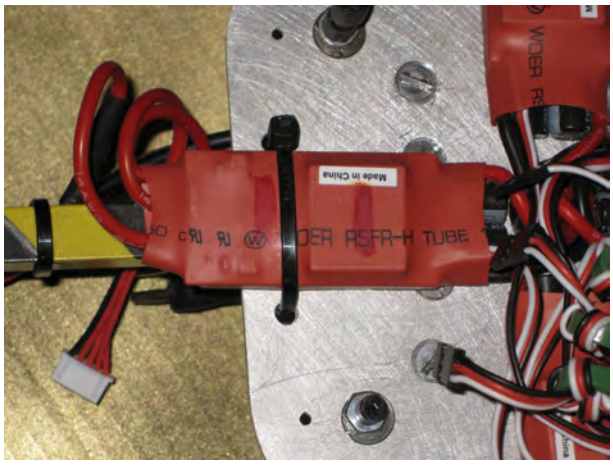


Figura 3.3: ESC Turnigy Plush 60 *A*

Otra característica de los motores *brushless*, es que para su funcionamiento, requieren de controladores de velocidad electrónicos (ESC*, Electronic Speed Controllers*), que transforman una señal PWM (*Pulse Width Modulation*) parecida a la que controla un servomotor, en una variación de corriente en los transistores que alimentan las bobinas del motor.

La señal PWM de entrada tiene una frecuencia de 500 Hz y un ancho de pulso reconocido por los ESC de entre 1 ms y 2.5 ms, donde 1 ms es para una velocidad mínima en el motor y 2.5 ms para una velocidad máxima. Una de las características de selección de estos controladores de velocidad, es la corriente máxima de operación. Ya que los motores seleccionados, consumen como máxima corriente hasta 42 A, se eligieron los ESC Turnigy Plush de 60 A los cuales cuentan con características adicionales como: dos tipos de freno de motor (rápido y progresivo), tres diferentes voltajes de corte que permiten detener la alimentación a los motores cuando el voltaje en las baterías llega a un determinado límite, arranque programable (rápido, lento o progresivo), entre otros. Su peso es de 60 g cada uno y cuentan con una tarjeta de programación para acceder a las diferentes características mencionadas.

3.2. Energía

La energía requerida, depende del voltaje y la corriente requerida tanto por los actuadores, como por los demás componentes electrónicos, sin embargo, el mayor consumo se encuentra en los actuadores.

Para los motores seleccionados, el voltaje máximo de alimentación es de 18 V, los picos de corriente se encuentran entre 25 A y 42 A y la corriente nominal es de entre 17 A y 32 A, sin embargo, tomando en cuenta el peso deseado del prototipo, para que los cuatro rotores tengan la capacidad de elevar el cuadricóptero, éstos operan aproximadamente

Figura 3.4: Batería Rhino 2250 mAh

por debajo de la mitad de su velocidad máxima permitida, lo que en corriente significa un consumo de aproximadamente 56 A para los cuatro motores. Se deseó que las baterías no superaran un peso de 300 g cada una y que la energía proporcionada sea suficiente para mantener un vuelo de aproximadamente 5 min, que es un tiempo también suficiente en el que se pueden mostrar resultados con repsecto al cumplimiento del objetivo de estabilidad antes mencionado, es decir que para cuatro de los motores seleccionados se requiere una batería de 18 V y que soporte 56 A por 5 min; con estos datos y para calcular los mAh deseados en las baterías, se realizó el siguiente cálculo:

$$56[A] \cdot 5[min] = 280[Amin]$$

$$\frac{280[Amin] \cdot 1000[mA] \cdot 1[h]}{1[A] \cdot 60[min]} = 4666[mAh]$$

Con el resultado anterior, se propuso usar, dos baterías LiPoly RHINO 4S 40Cd de 14.8 V, 2250 mAh y 275 g cada una, donde 4S es el número de celdas conectadas en serie y 40C es una constante máxima de descarga sin que sufra daño la batería, dicha constante es calculada como:

$$nCd = n \cdot Capacidad\, en\, [A] \tag{3.1}$$

Por lo que para la batería seleccionada se tiene que la corriente máxima que se le puede solicitar es de: $40Cd = 40 \cdot 2.25[A] = 90[A]$. Si un motor KD 36-10XL requiere eventualmente 20 A, los cuatro requieren 80 A, que es un valor dentro del rango de descarga máxima de corriente de la batería.

3.3. Sensores

Los sensores necesarios para cumplir el objetivo de estabilidad del cuadricóptero son aquellos relacionados con las variables que influyen en la dinámica del mismo, como las posiciones, velocidades y aceleraciones angulares, así como las posiciones, velocidades y aceleraciones lineales; también se pueden incluir variables que mantengan fija su orientación y su posición geoespacial. Actualmente, existen sensores inerciales relacionados con algunas de las variables anteriores y sistemas de geoposicionamiento que permiten conocer la ubicación de algún objeto con respecto a la longitud, latitud y altitud terrestre [17]. Ejemplos de este tipo de sensores son los acelerómetros, los girómetros, los magnetómetros, los barómetros y los sistemas GPS.

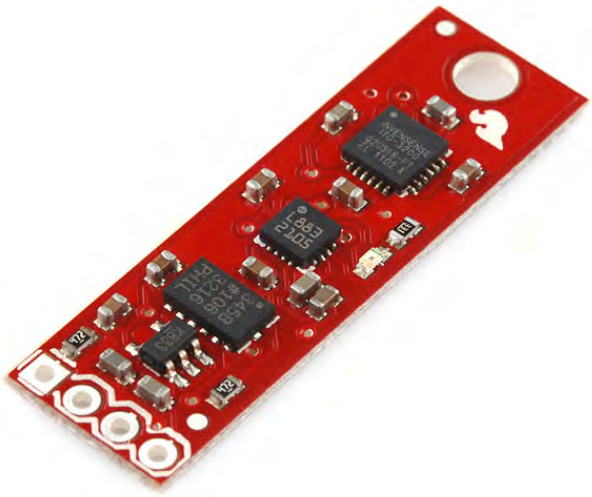


Figura 3.5: 9 DOF Sensor Stick

Todos los anteriores pueden interactuar ya sea de manera analógica (variaciones de voltaje) o digital con algún procesador de información que utilice ésta, para aplicaciones que van desde monitoreo hasta control. Se puede encontrar una gran variedad de este tipo de sensores, incluso, algunos integrados en sistemas embebidos controlados por algún microprocesador y de tamaños muy pequeños en un solo circuito integrado, como se menciona en [18].

Como se mencionó anteriormente, IMU se refiere a los sistemas inerciales embebidos o unidades medidoras de inercia, a los sistemas inerciales embebidos pero con características adicionales como orientación y posición geoespacial, se les conoce como AHRS (*Attitude Heading Reference System*).

Por sus características de simplicidad de funcionamiento y comunicación, se eligió el sensor de 9 Grados de Libertad (DOF, *Degrees Of Freedom*): 9 *DOF Sensor Stick* de SparkFun Electronics [19], que contiene un acelerómetro ADXL345 [20], relacionado con aceleraciones lineales estáticas y dinámicas, un girómetro ITG-3200 [21], relacionado con

velocidades angulares, un magnetómetro HMC-5883L [22], relacionado con la orientación respecto de los ejes magnéticos terrestres, y un circuito integrado que fusiona las señales y las arroja en una salida digital por medio de la tecnología I^2C.

3.4. Procesamiento

De los sensores, se obtiene la información necesaria para alimentar el algoritmo de control, sin embargo, para llevar a cabo la acción de control, es necesario tomar en cuenta que la variación de la velocidad de los motores también está involucrada con los estados del sistema, por lo que se requiere contar con un dispositivo que pueda llevar a cabo dicha tarea, así como poder realizar los cálculos propios de la ley de control. Los datos enviados por el sensor, la ley de control, el algoritmo de despegue y aterrizaje, así como las señales que requieren los ESC para aumentar o decrementar la velocidad de los motores, se encuentran alojados en el microcontrolador ATMega 1280 instalado en una tarjeta de desarrollo llamada Arduino Mega®. Dicha arquitectura trabaja con un oscilador de 16 MHz y alcanza una velocidad de procesamiento de 16 MIPS. La programación del microcontrolador se realiza por medio del lenguaje *Wiring* [23], a través del ambiente de desarrollo basado en *Processing* [24], los cuales en su conjunto forman la plataforma denominada Arduino® [25].

Figura 3.6: Arduino Mega® 1280

3.5. Comunicación Inalámbrica

Para cualquier dispositivo de radio control o de monitoreo remoto, es necesario contar con una comunicación inalámbrica robusta que permita enviar señales de control, monitoreo o de ejecución. Además, por seguridad del usuario, es indispensable mantenerse a una cierta distancia alejado del dispositivo a controlar, para disminuir la probabilidad de sufrir algún accidente. Existen varias arquitecturas de comunicación inalámbrica entre las

que se encuentran: Radio Frecuencia Simple, Bluetooth®, WiFi®, Luz infrarroja, etc. Considerando el tipo de aplicación y el ambiente de operación, además de su sencillez de funcionamiento y configuración, se eligió la radiofrecuencia como medio de comunicación entre la base y el cuadricóptero. Además, existen dispositivos que hacen aún más sencilla la comunicación entre dos o más dispositivos por este medio. Uno de estos dispositivos es el creado por Digi®: el XBee®, el cual es un módulo de comunicación por radiofrecuencia que trabaja bajo el protocolo zigbee® [26]; esta arquitectura requiere de un módulo Xbee® por cada dispositivo a comunicar.

Figura 3.7: Módulo XBee Pro 900®

A estos módulos, dependiendo del modelo, se les pueden modificar algunos parámetros de comunicación como: dirección de envío, canal de comunicación, velocidad de comunicación, bits de paro, entre otros. Por las características de operación mencionadas y las distancias de trabajo del cuadricóptero, se eligió el Xbee Pro 900 XSC RPSMA® [27]. Este modelo de Xbee® trabaja con alimentación de 3.3 V, tiene capacidad de configuración de modo en reposo, tiene un alcance de aproximadamente 24 km con antenas de alta ganancia y existen tarjetas de conexión para puerto USB para comunicación directa con una computadora y el dispositivo a controlar. Los componentes electrónicos de procesamiento, sensado y comunicación, junto con los cables y conectores, tienen un peso de 275 g.

Después de haber seleccionado los componentes electrónicos con los que contará el prototipo de cuadricóptero, a continuación se procede con el diseño de la estructura sobre la que se fijarán dichos componentes, lo que incluye el diseño de algunas propuestas de configuración y el análisis de la estructura final para corroborar que cumple con las especificaciones definidas previamente.

Capítulo 4

Diseño de la estructura mecánica

En este capítulo, se muestran las diferentes propuestas de configuración del prototipo de cuadricóptero con base en sus funciones como se muestra en la Figura 4.1, así como su respectiva evaluación y evolución hasta llegar a la configuración final. A esta última se le realizó un análisis de deformación por elemento finito para determinar los materiales y sus características que permitan llevar el diseño a su realización y construcción.

Figura 4.1: Configuración de funciones

4.1. Configuración

Dentro del proceso de diseño se realizaron diferentes propuestas de configuración de la estructura y el acomodo de partes y componentes del cuadricóptero, donde se tomaron en cuenta las características positivas de cada diseño y propuesta para lograr un cuadricóptero ligero, funcional, y robusto, dichas propuestas fueron dibujadas con ayuda del software de dibujo asistido por computadora (CAD, *Computer Asisted Drawing*) Solid Edge ST2® [28].

4.1.1. Propuesta 1

Desde el comienzo de la investigación sobre diversas configuraciones de cuadricópteros, se encontró que la mayoría de los diseños convergían a uno cuya configuración es en cruz, con

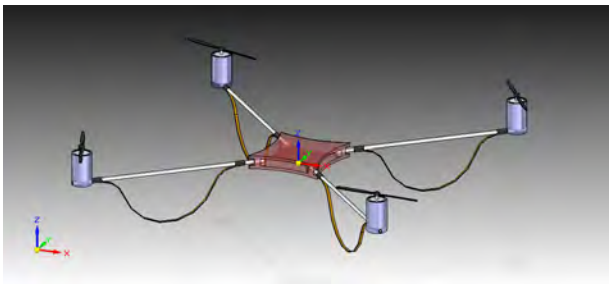


Figura 4.2: Propuesta 1

los motores en los extremos y una plataforma central, sin embargo existen otros como el Draganflyer X8® [29] de 8 rotores o un cuadricóptero con tres rotores de empuje vertical y uno para cambio de dirección horizontal [5], de 3 rotores en los vértices de un triángulo, de 6 rotores, etc. En este trabajo se decidió mantener la configuración en cruz con 4 motores en los extremos y una plataforma central.

En esta primera propuesta, se comenzó a trabajar con la arquitectura del cuadricóptero para tener una idea inicial de los acoplamientos, la ubicación de elementos base, entre otras. De forma más específica, en esta propuesta no se contempló alguna dimensión comercial de tubos, los motores solo estaban contenidos en un cilindro de material desconocido y la placa central era una geometría irregular de material no definido; además no existía tornillería ni propuestas de unión, pero permitió comenzar a crear nuevas ideas de reducción de espacios, aumento de robustez y sobre todo reconocer la necesidad de comenzar a contar con análisis físicos que dieran certeza a la elección de materiales y dimensiones. Cabe mencionar que desde esta propuesta, se pensó en ocupar un diseño de tamaño relativamente grande ya que esto permitiría brindar mayor estabilidad en ambientes exteriores.

4.1.2. Propuesta 2

Entre la primera propuesta y ésta, se generaron varios bocetos con más o menos componentes, nuevas configuraciones, pequeños cambios de geometría, etc. La segunda propuesta es el resultado de la suma de todas estas ideas; es un modelo más trabajado, con dimensiones reales de los motores seleccionados, las baterías, los controladores de velocidad, los módulos de comunicación y los sensores. También se visualizan las características dimensionales aproximadas de las hélices para corroborar su compatibilidad con los motores y así comprobar que las dimensiones geométricas de los motores no fueran mucho más grandes o chicas que las de las hélices.

En esta segunda propuesta, se incluyó una ubicación real de componentes, lo que ayudó a limitar algunas longitudes y espacios como la altura entre las placas centrales para ubicar las baterías, la longitud de los brazos para evitar interferencias entre las hélices, la manera de sujetar los motores, entre otras. En esta propuesta, se utilizó tubo redondo

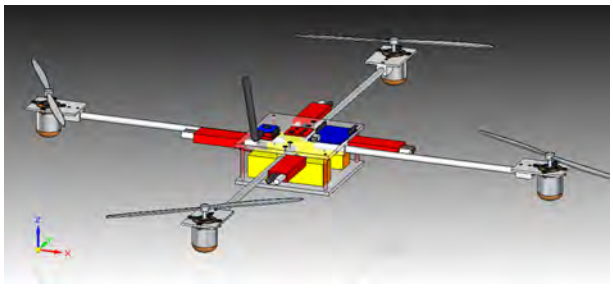


Figura 4.3: Propuesta 2

con acoplamientos cuadrados, también comenzó a aparecer la tornillería y surgió la idea de ocupar aluminio para las placas centrales para aumentar la robustez de la estructura.

4.1.3. Propuesta 3

Finalmente, entre la segunda propuesta y la propuesta final, se siguieron una numerosa serie de pasos para obtener este último diseño, entre ellos, se encuentran análisis de deformación de la estructura que se presentan más adelante, la elección final de materiales, los espesores definitivos de lámina, varios rectificados de las dimensiones reales de los componentes, las geometrías finales de las láminas centrales y las que sujetan los motores, la elección del método de sujeción de las baterías, la reducción de elementos como por ejemplo, la segunda lámina central de aluminio y la colocación de una lámina de acrílico para reducir el peso total del cuadricóptero. Además, en ésta, se propuso una configuración y ubicación final de los componentes electrónicos y se generó el dibujo CAD de las hélices con dimensiones reales.

Se encontró conveniente el uso de perfil cuadrado de aluminio para los brazos del cuadricóptero debido, en parte, a la facilidad de sujeción de éstos con la estructura y por último, se tomó en cuenta la estética del cuadricóptero para el acabado final. Con lo anterior, se logró un diseño robusto, simple y funcional que se puede utilizar para cumplir tanto el objetivo particular de este trabajo, como la posibilidad de utilizarlo para alcanzar los demás objetivos generales tanto de control como de funcionalidad.

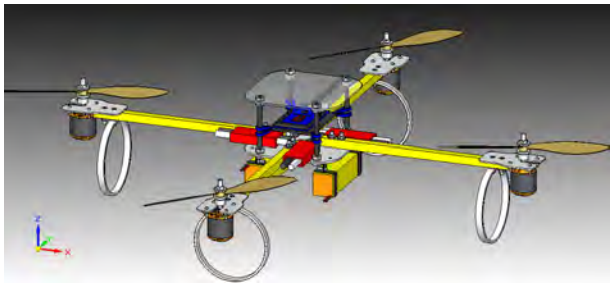


Figura 4.4: Propuesta final

4.2. Análisis y realización

El marco del cuadricóptero está inscrito en un cuadrado de 0.50 m x 0.50 m, donde el eje de cada motor coincide con el vértice de dicho cuadrado. El cuadricóptero está compuesto por una estructura en cruz hecha de perfil cuadrado de aluminio de 12.7 mm de lado exterior con 1.65 mm de espesor. Este perfil se eligió con base en un análisis de deformación como el de la Figura 4.5.

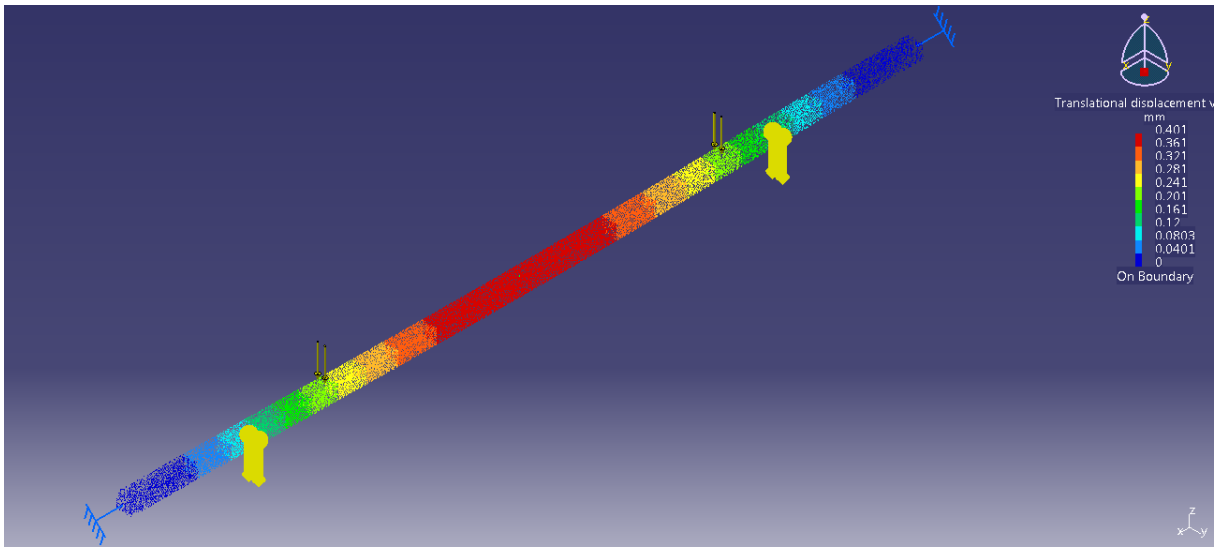


Figura 4.5: Análisis de deformación para perfil cuadrado de aluminio

Las pruebas se realizaron por elemento finito con ayuda del software Catia V5R19® [30] con muestras de distintas dimensiones comerciales [31], fijas por los extremos; se mantuvo constante tanto la longitud en 700 mm (que es aproximadamente la longitud total del eje de un motor al eje del motor contrario), como la carga distribuida en 39.2 N (que es aproximadamente el peso total máximo del cuadricóptero) y la aceleración de la gravedad en 9.8 m/s, los resultados registrados se muestran en el Cuadro 4.1.

Cuadro 4.1: Resultados del primer análisis de deformación

Muestras de aluminio	Deformación máxima
Tubo redondo 9.5 mm Ø x 1.24 mm espesor	2.25 mm en el centro
Tubo redondo 12.7 mm Ø x 1.24 mm espesor	0.68 mm en el centro
Perfil cuadrado con esquinas redondeadas 12.7 mm lado x 1.65 mm espesor	0.408 mm en el centro
Perfil cuadrado 12.7 mm lado x 1.65 mm espesor	0.401 mm en el centro

La selección se basó en el perfil o tubo con menor deformación máxima pero también con mayor facilidad de manufactura e instalación, es decir, con base en el menor número

tanto de piezas requeridas para fijar los tubos o perfiles al resto de la estructura, como de herramientas utilizadas para manufacturar dichas piezas.

De manera análoga, para la lámina central que sujetaría cada uno de los cuatro perfiles cuadrados con la configuración mostrada en la Figura 4.6, se realizaron diferentes análisis de deformación. Probando con diferentes muestras, se mantuvo constante tanto la longitud de la lámina central en 150 mm de lado, así como los radios y distancias entre barrenos, la carga distribuida en 39.2 N y la aceleración de la gravedad en 9.8 m/s; la variable es el espesor comercial [31] de cada lámina de aluminio.

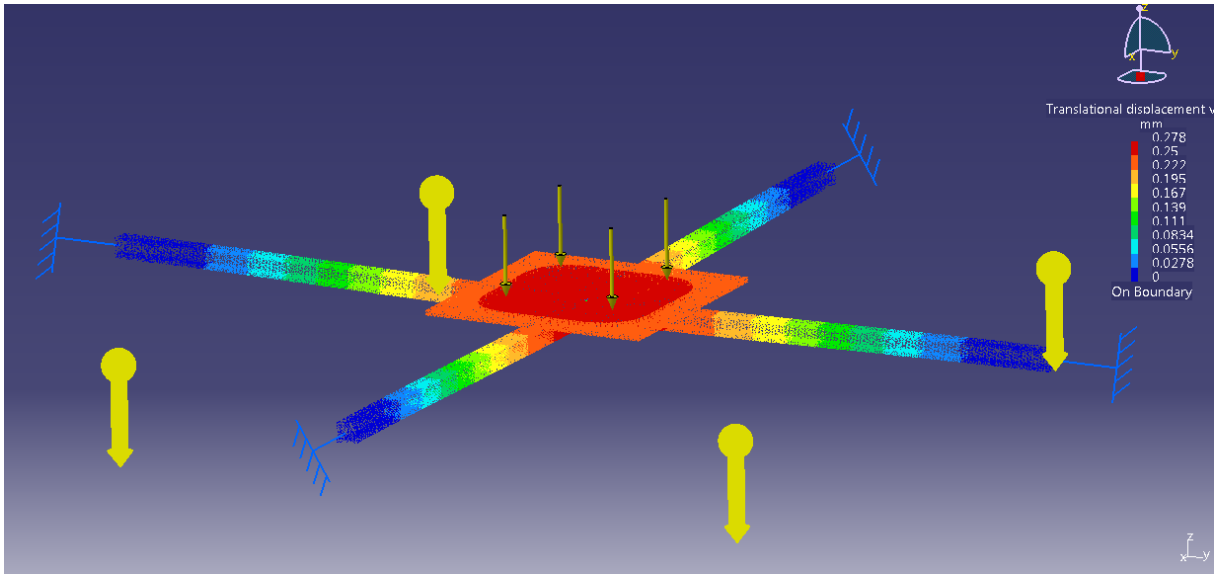


Figura 4.6: Análisis de deformación para lámina de aluminio central

Los resultados registrados se muestran en el Cuadro 4.2 para cada espesor de lámina utilizado en las pruebas.

Cuadro 4.2: Resultados del segundo análisis de deformación

Muestras de aluminio	Deformación máxima
Calibre 10 con 3.4 mm espesor	0.251 mm en el centro
Calibre 12 con 2.8 mm espesor	0.278 mm en el centro
Calibre 14 con 2.1 mm espesor	0.309 mm en el centro

La selección de la lámina se hizo con base en los resultados anteriores de deformación, pero tomando en cuenta también, el peso de cada muestra, así pues entre la lámina de calibre 10 y 12, existe una diferencia pequeña en la deformación máxima (0.027 mm), sin embargo tienen una diferencia de aproximadamente 80 g lo que hace a la lámina de calibre 12 una mejor opción. Entre la lámina de calibre 12 y 14, existe una diferencia de aproximadamente 10 g en peso, y la diferencia en deformación es también pequeña,

aunque un poco mayor (0.031 mm) por lo que no se obtiene mayor ventaja en el peso a comparación de la deformación obtenida; por lo anterior, la lámina calibre 12 fue elegida como la mejor opción y es la que se seleccionó para la construcción de la estructura.

Finalmente para la selección de las láminas que sujetarán los motores al resto de la estructura, se usó un criterio de selección también basado en los resultados anteriores, bajo la premisa de tener una fuerza distribuida igual a un cuarto del total de la carga usada en el análisis anterior, por lo que se eligió lámina de aluminio calibre 14 para esta parte de la estructura. El corte tanto de ésta pieza como de la lámina central se realizó por chorro de agua para mantener un buen acabado en el contorno de éstas.

Para sujetar las baterías se utilizaron 2 cinturones de aluminio por batería que simplemente la rodean por su perímetro y las sujetan a la lámina central por medio de cinturones de plástico ajustables. Para sujetar los controladores de velocidad a la lámina central y para agrupar los cables se utilizaron también, cinturones de plástico ajustables.

Como base para los componentes electrónicos, se ocupó una tabla de madera sujeta a cuatro postes de varilla roscada de $\frac{1}{4}$ " por medio de ligas para disminuir el ruido mecánico producido por la vibración de los motores; éste diseño estuvo inspirado en [18] y se muestra en la Figura 4.7. Para la cubierta superior se ocupó una placa de acrílico de 3 mm de espesor de las mismas dimensiones que la lámina central de aluminio en donde también se colocó un interruptor general para controlar la alimentación de energía del cuadricóptero.

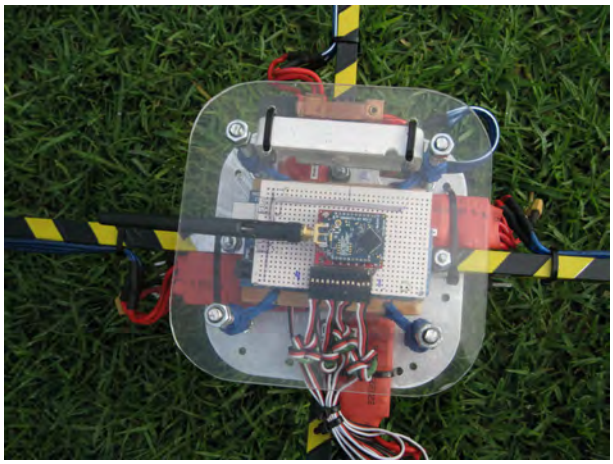


Figura 4.7: Plataforma

El soporte que funciona como tren de aterrizaje del cuadricóptero, está compuesto por cuatro aros de PVC de 10 mm de ancho y 6 mm de espesor colocados debajo de la lámina que sujeta a los motores, sobre el perfil cuadrado. El peso de la estructura de aluminio más la tornillería es de 1,135 g. El peso total del cuadricóptero con todos los componentes es de 3,000 g. Los planos de cada una de las piezas generadas para la construcción de la estructura del cuadricóptero y la lista de material, se encuentran en el *Apéndice A: Planos y Materiales*.

Capítulo 5

Diseño y simulación del sistema de control

Con los capítulos anteriores, se detalló el proceso de diseño y construcción del prototipo de cuadricóptero. A continuación se detalla el proceso de diseño del sistema de control basado en las especificaciones del prototipo construido, así como diversas simulaciones para visualizar, antes de la implementación, una aproximación a la respuesta del sistema con el sistema de control seleccionado.

5.1. Modelado matemático

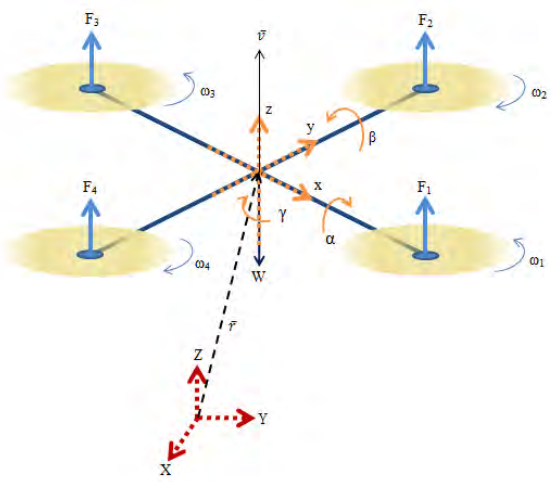


Figura 5.1: Configuración dinámica

Después de haber mostrado las características, los elementos y la configuración del cuadricóptero, así como la construcción del modelo físico en cuyas características se basará el diseño del controlador, se procedió a analizar el sistema dinámico. La configuración dinámica del cuadricóptero se muestra en la Figura 5.1. Existen dos marcos de referencia, el primero se encuentra fijo en el cuadricóptero y está definido por x, y, z y es donde se originan los ángulos de navegación (*roll*, *pitch* y *yaw*). El segundo es un marco de

referencia inercial al que están referidos todos los movimientos del cuadricóptero y está definido por X, Y, Z. Estos dos marcos de referencia están relacionados por un vector $\overline{r}$ y la matriz de rotación $\underline{R}$ mostrada en la Ecuación 5.1, donde c representa la función coseno y s la función seno.

$$\underline{R} = \begin{bmatrix} c\gamma c\beta & c\gamma s\beta s\alpha - s\gamma c\alpha & c\gamma s\beta c\alpha + s\gamma s\alpha \\ s\gamma c\beta & s\gamma s\beta s\alpha + c\gamma c\alpha & s\gamma s\beta c\alpha - c\gamma s\alpha \\ -s\beta & c\beta s\alpha & c\beta c\alpha \end{bmatrix} \tag{5.1}$$

El movimiento angular, como se indicó anteriormente, está dado por los ángulos: α, sobre el eje x resultado del movimiento de alabeo; β, sobre el eje y resultado del movimiento de cabeceo y γ, sobre el eje z resultado del movimiento de guiño. La fuerza de empuje total es generada por la suma de cada una de las cuatro fuerzas de empuje proporcionada por cada motor.

La fuerza del peso actúa en el centro del cuadricóptero considerando que es simétrico con respecto al eje x e y. Los motores sobre el eje x giran en sentido antihorario y los que se encuentran sobre el eje y giran en sentido horario, W equivale a la masa m_a total del cuadricóptero multiplicada por la aceleración de la gravedad G.

Las entradas del sistema dependen directamente de las velocidades angulares de los motores, el cambio de estas velocidades genera una variación en la fuerza de empuje y en los ángulos del cuadricóptero; lo anterior trae como consecuencia un desplazamiento sobre un vector $\overline{v}$ con componentes en X, Y, Z. Dichas entradas se calcularon tomando en cuenta lo siguiente:

$$F_i = b \cdot w_i^2 \tag{5.2}$$

Donde $i = \{1, 2, 3, 4\}$, F_i es la fuerza de empuje de cada rotor, w_i es la velocidad angular de cada rotor y b es una constante de proporcionalidad de los rotores que relaciona el empuje generado con la velocidad de giro que, para éste trabajo, se toma como igual para los cuatro motores con base en pruebas experimentales registradas en el Capítulo 7. Así pues, se tiene lo siguiente:

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} F_1 + F_2 + F_3 + F_4 \\ F_4 - F_2 \\ F_3 - F_1 \\ F_1 - F_2 + F_3 - F_4 \end{bmatrix} \tag{5.3}$$

Donde u_1 es responsable del empuje vertical, u_2 del alabeo, u_3 del cabeceo y u_4 del guiño. Reescribiendo las ecuaciones, se tiene que:

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -1 & 0 & 1 \\ -1 & 0 & 1 & 0 \\ 1 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} \tag{5.4}$$

Que equivale a la Ecuación 5.5.

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} b(w_1^2 + w_2^2 + w_3^2 + w_4^2) \\ b(w_4^2 - w_2^2) \\ b(w_3^2 - w_1^2) \\ d(w_1^2 + w_3^2 - w_2^2 - w_4^2) \end{bmatrix} \tag{5.5}$$

Donde d es una constante de arrastre que relaciona dos fuerzas horizontales con las velocidades del par de motores que giran en sentido horario y las del par que gira en sentido antihorario respectivamente.

Además, retomando la Ecuación 5.4, se obtuvo la matriz para determinar cómo se calculan las fuerzas F_i que deben actuar en cada motor a partir de las entradas u_i cuando se obtenga su valor en cualquier instante de la operación del control para, a su vez, conocer las velocidades w_i de los motores y determinar el PWM correspondiente que se debe comandar. Dado lo anterior, se obtuvo la Ecuación 5.6.

$$\begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} = \begin{bmatrix} 0.25 & 0 & -0.5 & 0.25 \\ 0.25 & -0.5 & 0 & -0.25 \\ 0.25 & 0 & 0.5 & 0.25 \\ 0.25 & 0.5 & 0 & -0.25 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \tag{5.6}$$

Las variables involucradas en el modelo matemático utilizado son:

$$M^T = \{ \; x \quad \dot{x} \quad y \quad \dot{y} \quad z \quad \dot{z} \quad \alpha \quad \dot{\alpha} \quad \beta \quad \dot{\beta} \quad \gamma \quad \dot{\gamma} \; \} \tag{5.7}$$

El modelo en variables de estado, incluyendo las derivadas del vector de estados está representado en la Ecuación 5.8. Donde I_x, I_y e I_z son las inercias sobre cada eje, L es la longitud de cada brazo entre el eje del motor y el centro del cuadricóptero, G es la aceleración de la gravedad y m_a es la masa total del cuadricóptero. Este modelo es una de las versiones reducidas del modelo completo de un cuadricóptero como en [15, 6] ya que no considera algunos efectos como en [14] tales como el aleteo producido en las hélices ante el cambio de dirección del cuadricóptero, o el complemento de las inercias producidas ante el giro del cuadricóptero sobre sus ejes.

$$\dot{M} = \begin{pmatrix} \dot{x} \\ (cos\alpha sin\beta cos\gamma + sin\alpha sin\gamma)u_1/m_a \\ \dot{y} \\ (cos\alpha sin\beta sin\gamma - sin\alpha cos\gamma)u_1/m_a \\ \dot{z} \\ -G + (cos\alpha cos\beta)u_1/m_a \\ \dot{\alpha} \\ \frac{L}{I_x}u_2 \\ \dot{\beta} \\ \frac{L}{I_y}u_3 \\ \dot{\gamma} \\ \frac{d}{I_z}u_4 \end{pmatrix} \tag{5.8}$$

Estas consideraciones y el modelo reducido, son válidas para puntos de operación cercanos al punto de equilibrio del cuadricóptero, es decir, que factores como el viento, maniobras agresivas y cambios de dirección, pueden ser despreciados mientras el cuadricóptero se mantenga flotando en el aire, que es parte del objetivo principal de este trabajo.

Retomando el modelo de la Ecuación 5.8, se puede observar que es un sistema no lineal y las primeras seis ecuaciones del mismo, corresponden únicamente a los movimientos de desplazamiento lineal, las cuales dependen de los ángulos (α, β, γ), de la entrada u_1, la masa y la gravedad; mientras que las últimas seis variables de estado, resuelven los movimientos angulares y no dependen directamente de los ángulos ni de la entrada u_1. Por lo anterior y a manera de simplificación, se consideró útil separar el sistema en dos subsistemas, uno para los movimientos angulares y otro para los movimientos lineales. El esquema que representa este concepto, se visualiza en la Figura 5.2.

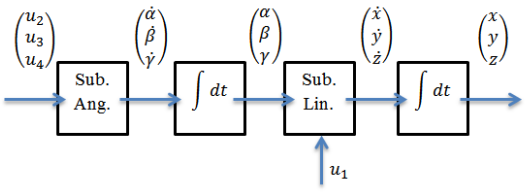


Figura 5.2: Modelo separado

5.2. Esquema general de control

Tomando en cuenta el objetivo de control relacionado con la estabilidad y una entrada de referencia igual a cero, en éste trabajo se entiende que el cuadricóptero está estabilizado cuando todas las derivadas del movimiento lineal y angular sean cero, y cuando todos los ángulos tomen también el valor de cero, quedando sólo el peso del cuadricóptero com-

pensado por el sistema y generando el empuje necesario para mantener el cuadricóptero flotando. Esto se demuestra como sigue:

$$\begin{aligned} si\ \alpha = \beta = \gamma = 0; \\ \dot{x} = \dot{y} = \dot{z} = 0; \\ \ddot{x} = \ddot{y} = \ddot{z} = 0; \end{aligned} \tag{5.9}$$

Entonces el el modelo del susbsitema lineal para valores de cero queda como:

$$\dot{M}_{Lo} = \left\{ \begin{array}{c} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -G + u_1/m = 0 \end{array} \right\} \tag{5.10}$$

$$\therefore u_1 = mG = W \tag{5.11}$$

Los vehículos aéreos autónomos, generalmente cuentan con dos lazos de control, el primero y principal, se le conoce como el lazo de control primario y se encarga de conseguir la estabilidad de los movimientos del sistema en cualquier punto del vuelo. El segundo se le conoce como lazo de tarea o misión, que se encarga de conseguir el control del sistema visto como una plataforma general y controla las trayectorias de vuelo del mismo. Siguiendo éste orden de ideas, se propone que, para el cuadricóptero, el control de tarea arroje como resultado un vector de velocidades deseado, que se utilizará para calcular los ángulos deseados, los cuales son la información de entrada para el algoritmo del lazo de control primario. Así también se toma como consideración importante que el lazo de control primario responda lo suficientemente más rápido que el control de tarea para garantizar la estabilidad en todos los puntos del vuelo, ya sea flotando o realizando algún seguimiento de trayectoria.

En este trabajo, como una primera aproximación al control de estabilidad, y ya que de la IMU se pueden obtener datos de aceleración lineal y velocidad angular, mismos que se fusionan por medio de algoritmos como el *filtro Kalman* para obtener el valor del ángulo de inlinación de cada eje, se diseñó un algoritmo de control con la técnica PID y se implementó en el modelo físico para visualizar el comportamiento exclusivamente para el subsistema angular y corroborar la estabilidad del mismo. Esto funcionó como el lazo de control primario del cuadricóptero y se omitió el lazo de tarea o misión ya que para efectos de la estabilidad, no se requiere de una trayectoria o tarea más que el mantenerse flotando en el aire.

Posteriormente, de manera adicional y para analizar el uso de otra técnica de control, se diseñó y simuló (sin implementar) otro algoritmo, ahora con la técnica de re-

alimentación de estados para el subsistema angular y un algoritmo de control proporcional para el subsistema de movimientos lineales, ambos trabajando en conjunto en un mismo sistema anidado en el que la respuesta de uno sirve como entrada para la otra parte, con esto se deseó observar también el control de regulación y seguimiento del cuadricóptero. El diagrama de bloques que representa los lazos de control principal y secundario junto con los algoritmos de control Con.1 (realimentación de estados) y Con. 2 (proporcional) y las entradas y salidas de cada bloque está representado en la Figura 5.3.

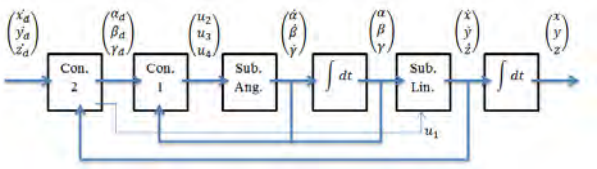


Figura 5.3: Control en bloques para los tres objetivos

5.3. Propuesta 1: Control PID para el subsistema angular

5.3.1. Diseño del control PID

El control Proporcional Integral Derivativo o *PID* se basa en realimentar información de salida de un sistema, compararla con una referencia y con base en el error entre la referencia y la señal de realimentación, calcular la señal de entrada del sistema. Como su nombre lo indica, el control PID se compone de tres "sub-controladores", los cuales trabajan con diferentes funciones del error. Así entonces, el control proporcional, se refiere a una constante o ganancia multiplicada por el error, el control integral, se refiere a la integral del error multiplicada por otra constante y, finalmente el control derivativo, se refiere a la derivada del error multiplicada por una tercera constante. La suma de las tres señales del controlador PID es la entrada del sistema. Cada parte del control PID tiene su propia función dentro de la respuesta deseada del sistema, por ejemplo el control integral ayuda a reducir a cero el error en estado estable del sistema y el control derivativo, ayuda a mejora el tiempo de asentamiento de la respuesta del sistema.

Existen diferentes versiones del control PID como en algunas aplicaciones que requieren o permiten utilizar sólo partes del control, por ejemplo, PI o PD. Lo anterior, aunado con la facilidad de diseño de las constantes desde el punto de vista del procedimiento para calcularlas, hacen del control PID una herramienta versátil y adaptable a las necesidades de cada aplicación, por lo que es ampliamente utilizado en el área industrial. Algunos de los problemas que existen al usar un control PID son por ejemplo, que la versión lineal del controlador, tiene una respuesta robusta en sistemas lineales, lo puede no suceder con la respuesta en sistemas no lineales o linealizados, además el diseño de

las constantes, en ocasiones es por prueba y error haciendo uso de la incertidumbre, lo que puede provocar demoras en encontrar las constantes adecuadas para un determinado funcionamiento del control y la posibilidad de encontrar la configuración no óptima de dichas constantes. El controlador en un diagrama de bloques sencillo se expresa en la Figura 5.4.

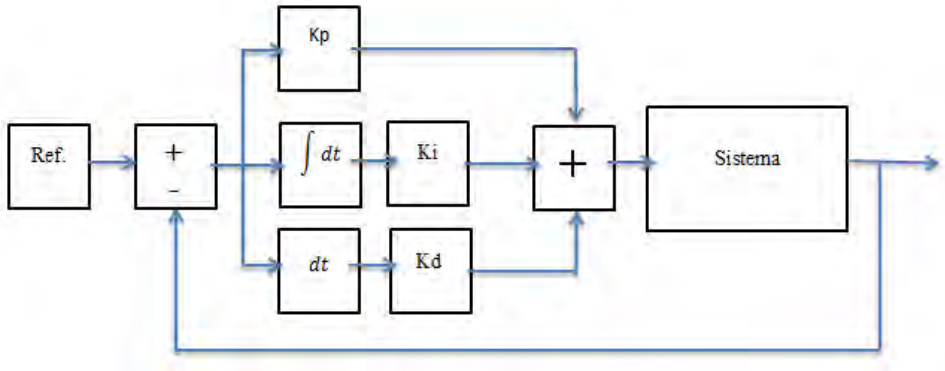


Figura 5.4: Algoritmo básico del control PID

Para la primera aproximación al control de la estabilidad del cuadricóptero, se utilizó este algoritmo de control, en donde únicamente interviene la parte angular ya que como se demostró anteriormente, el mantener los ángulos de navegación en cero, así como sus derivadas, trae como resultado la estabilidad del cuadricóptero manteniéndolo flotando en algún punto de operación. Por lo anterior y retomando del modelo en variables de estado las últimas seis ecuaciones, se tiene lo siguiente:

$$\dot{M}_A = \left\{ \begin{array}{c} \dot{\alpha} \\ \frac{L}{I_x} \cdot u_2 \\ \dot{\beta} \\ \frac{L}{I_y} \cdot u_3 \\ \dot{\gamma} \\ \frac{d}{I_z} \cdot u_4 \end{array} \right\} \tag{5.12}$$

Como se puede notar, es un sistema lineal, por lo que reescribiendo el modelo matemático, en la forma general de variables de estado, se tiene lo siguiente:

$$\dot{M}_A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} M_A + \begin{bmatrix} 0 & 0 & 0 \\ \frac{L}{I_x} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \frac{L}{I_y} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \frac{d}{I_z} \end{bmatrix} U$$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} M_A + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} U \tag{5.13}$$

Donde $M_A^T = \left\{\alpha, \dot{\alpha}, \beta, \dot{\beta}, \gamma, \dot{\gamma}\right\}$ y $U^T = \{u_2, u_3, u_4\}$, de tal manera que se obtienen las matrices A, B, C y D del modelo en variables de estado. Tomando en cuenta la independencia de los movimientos angulares y su representación matemática, se decidió buscar la estabilidad de cada uno de los ángulos por separado, ocupando el mismo algoritmo de control PID.

Así entonces, para α y para β, el modelo sería idéntico, mientras que para γ, únicamente es necesario cambiar en la matriz B del modelo, la constante L por d, por lo que se puede generalizar para los tres ángulos independientes, el siguiente modelo:

$$\dot{M}_{AS} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \dot{\alpha} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{L}{I_x} \end{bmatrix} [u_2]$$
$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \dot{\alpha} \end{bmatrix} \tag{5.14}$$

Para el cálculo de las inercias sobre cada eje, se ocuparon las siguientes ecuaciones, basadas en [32].

$$I_x = I_y = \frac{m_0 \cdot r_0^2}{2} + \frac{m_0 \cdot h_0^2}{6} + 2 \cdot m_0 \cdot L^2 + \frac{m_1 \cdot r_1^2}{4} + \frac{m_1 \cdot h_1^2}{12}$$
$$I_z = \frac{m_1 \cdot r_1^2}{2} + 4 \cdot m_0 \cdot L^2 \tag{5.15}$$

Donde m_0 corresponde a la masa de cada rotor (0.2 kg), r_0 al radio de cada motor (0.05 m), h_0 a la altura de cada rotor (0.07 m), m_1 a la masa de un cilindro que contiene la parte central del cuadricóptero (0.8 kg), r_1 al radio de dicho cilindro (0.08 m) y h_1 a la altura del mismo (0.16 m).

Para tener una idea más clara de estas dimensiones, se puede tomar como referencia la Figura 5.5.

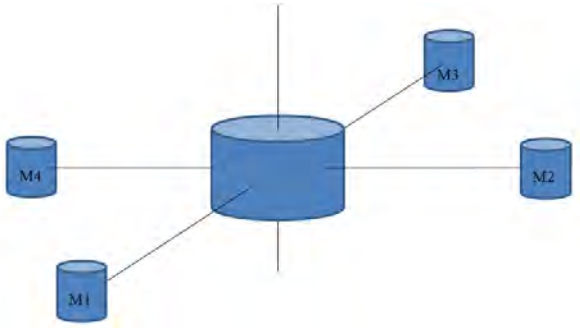


Figura 5.5: Cilindros inerciales

Finalmente, los parámetros del cuadricóptero utilizados son los siguientes:

$$m = 3.0\,kg$$
$$G = 9.8\,m/s^2$$
$$I_x = I_y = 0.0524\,kg \cdot m^2$$
$$I_z = 0.10056\,kg \cdot m^2$$
$$L = 0.35\,m$$
$$d = 1.5$$
$$b = 3.5\,x\,10^{-7}$$

Con base en la estructura del control PID, los parámetros y el modelo anterior, se determinó la función de transferencia del sistema y a su vez, la ecuación característica de lazo cerrado del mismo como sigue:

Se utilizó la siguiente ecuación, para convertir el espacio de estados del modelo simplificado para un ángulo, en función de transferencia $Q(s)$.

$$Q(s) = C(sI - A)^{-1}B \tag{5.16}$$

Obteniendo la siguiente ecuación:

$$Q(s) = \frac{6.68}{s^2} \tag{5.17}$$

De la teoría de control, se sabe que el controlador PID, tiene por función de transferencia, la siguiente:

$$H(s) = \frac{K_d s^2 + K_p s + Ki}{s} \tag{5.18}$$

La cual se utilizó para calcular la ecuación característica de lazo cerrado del sistema realimentado, tomando en cuenta el controlador PID. Por lo anterior, se tiene que dicha función de transferencia es equivalente a:

$$\frac{Q(s)H(s)}{1+Q(s)H(s)} = \frac{\frac{6.68}{s^2}\left(\frac{K_d s^2+K_p s+K_i}{s}\right)}{1+\frac{6.68}{s^2}\left(\frac{K_d s^2+K_p s+K_i}{s}\right)} = T(s) \tag{5.19}$$

Simplificando la ecuación anterior y tomando en cuenta su denominador, se obtiene la ecuación característica de lazo cerrado mostrada en Ecuación 5.20:

$$s^3 + 6.68 \cdot K_d \cdot s^2 + 6.68 \cdot K_p \cdot s + 6.68 \cdot K_i = 0 \tag{5.20}$$

En donde K_d, K_p y K_i son las constantes que definen el comportamiento del control PID. Para determinar esas constantes, fue necesario recurrir a la selección de una ecuación característica deseada que proporcionara el desempeño o la respuesta transitoria también deseada y comparar ambas ecuaciones término a término, resolviendo los coeficientes correspondientes de ambas. Para la ecuación característica deseada basada en la forma estándar de un sistema de segundo orden [33], se eligieron los siguientes parámetros de diseño iniciales: porcentaje de sobrepaso ($\%sp$) y tiempo de asentamiento (ts) con los siguientes valores basados en una primera aproximación aleatoria:

$$\%sp = 5\,\%$$

$$ts = 1\,s$$

Con los valores anteriores y utilizando las siguientes ecuaciones:

$$ts = \frac{4}{\zeta \cdot w_n} \tag{5.21}$$

$$\%sp = e^{-(\sigma/w_d)\pi} \cdot 100\% \tag{5.22}$$

$$w_d = w_n\sqrt{1-\zeta^2} \tag{5.23}$$

$$\sigma = \zeta \cdot w_n \tag{5.24}$$

Donde para sistemas subamortiguados de segundo orden, ζ es el factor de amortiguamiento, W_n es la frecuencia natural del sistema, W_d la frecuencia amortiguada y σ el factor de atenuación; se obtuvo la siguiente ecuación característica:

$$s^2 + 8s + 33.6 = 0 \tag{5.25}$$

Con los siguientes polos dominantes:

$$P_{1,2} = -4 \pm 4.195j$$

Sin embargo, se requiere de un polo más para poder definir la ecuación característica deseada. Este polo se asignó de acuerdo al criterio de selección en el que se busca un polo sobre el eje real negativo no dominante para evitar que influya significativamente en la respuesta transitoria pero no tanto como para producir picos grandes en la oscilación de la misma que pudieran afectar la respuesta de los motores, por lo tanto se eligió como tercer polo a: $P_3 = -14$.

Con los datos anteriores, se obtuvo la siguiente ecuación característica deseada:

$$s^3 + 22s^2 + 145.6s + 470.4 = 0 \tag{5.26}$$

Igualando los coeficientes de la Ecuación 5.20 con los de la Ecuación 5.26, se obtuvieron los siguientes valores de Kp, Ki y Kd:

$$Kp = 21.8$$
$$Ki = 70.42$$
$$Kd = 3.24$$

De manera análoga y con base en el comportamiento y respuesta del sistema, se calcularon otras series de constantes. Estas pruebas quedaron registradas en el Capítulo 7 y la implementación junto con la programación del microcontrolador con la ley de control y algunos filtros, están registrada en el Capítulo 6.

5.3.2. Simulación del control PID para el subsistema angular

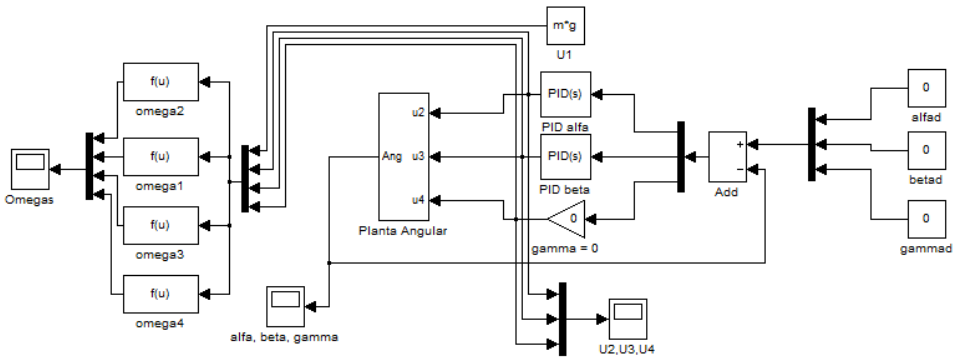


Figura 5.6: Diagrama de bloques para el control PID

Habiendo diseñado el controlador y sus parámetros, además de contar con el modelado del sistema, se procedió a verificar funcionamiento y el comportamiento de dicho controlador junto con el sistema del cuadricóptero por medio de una simulación en el programa MatLab® [34] con ayuda de su biblioteca SimuLink®. Esto permitió ingresar por medio

de bloques, las ecuaciones del modelo matemático del sistema de movimientos angulares y la ley de control.

El modelo utilizado se muestra en la Figura 5.6 en donde aparece la planta angular cuyas entradas son u_2, u_3 y u_4 y como salida, está el vector de ángulos que se realimenta para restarlo a la referencia y así obtener el error; posteriormente se hace pasar el error para α y para β por el control PID con las constantes obtenidas anteriormente y para γ, debido a que el sensor y la biblioteca utilizada para fusionar sus señales y calcular los ángulos, presenta al momento de éstas pruebas, inexactitudes para dicho ángulo, se asume que el error es igual a cero. Las salidas de cada controlador PID α y para β, así como la señal de error igual a cero para para γ, son las entradas correspondientes mencionadas para la planta angular mientras que la entrada u_1 se asume constante e igual al peso del cuadricóptero al no tener control sobre el desplazamiento vertical.

5.3.3. Resultados de la simulación del sistema angular y el control PID

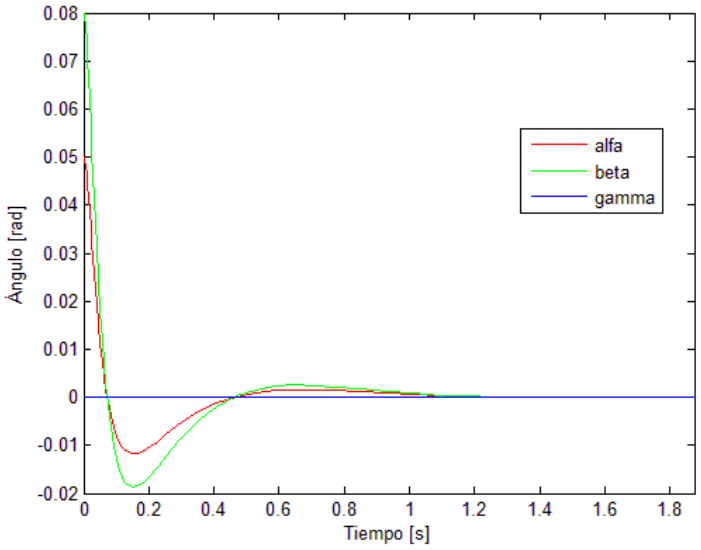


Figura 5.7: Respuesta angular con el PID para estabilidad

Se obtuvieron diversas gráficas para observar el comportamiento del sistema y el control PID para el objetivo de estabilidad, por lo que como condición inicial en α se eligió un valor de 0.05 rad y para β un valor de 0.08 rad, mientras que para γ se eligió el valor de cero. La Figura 5.7 muestra el comportamiento de los ángulos en el tiempo, con lo que se observa que para el controlador elegido, los ángulos sí se estabilizan según los parámetros elegidos que son 1 s para el tiempo de asentamiento y 5 % de sobrepaso.
La Figura 5.8 muestra el comportamiento de las entradas u_2, u_3 y u_4 que también se estabilizan al pasar el tiempo, con lo que también para el control diseñado se obtiene la respuesta deseada. Estas entradas, como se ha mencionado corresponden a los movimientos de alabeo, cabeceo y guiño.

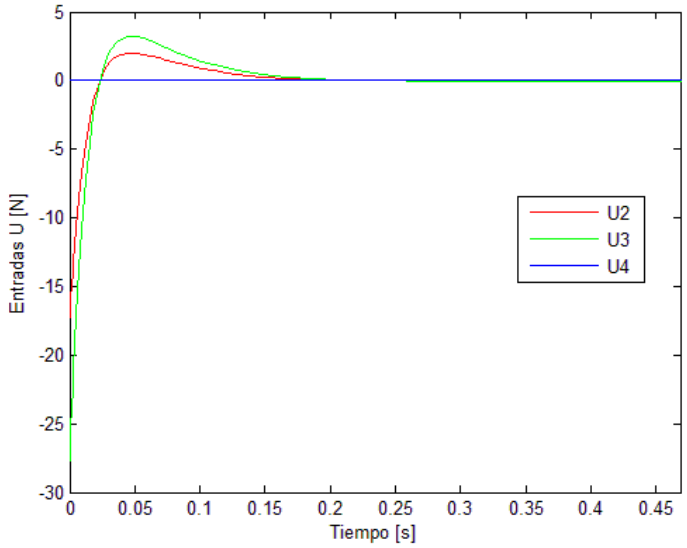


Figura 5.8: Respuesta de u_2, u_3 y u_4 con el PID para estabilidad

En la Figura 5.9 se observa el comportamiento en el tiempo de las velocidades angulares de cada motor, éstas también se estabilizan después de un cierto tiempo pero no a cero sino al valor necesario para mantener flotando el cuadricóptero.

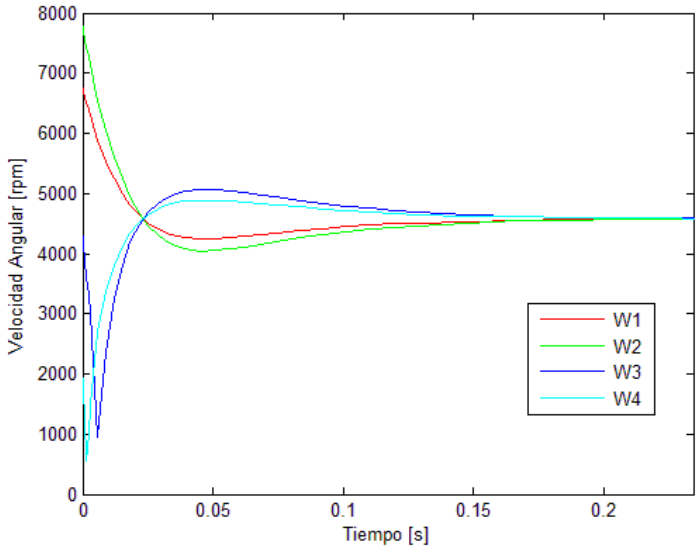


Figura 5.9: Respuesta de las velocidades de cada motor para estabilidad

Con las gráficas anteriores se puede concluir que el control PID diseñado sirve para estabilizar el subsistema angular del cuadricóptero, por lo que prosiguió su implementación registrada en el Capítulo 6 y observar los resultados para determinar la necesidad de cambiar los parámetros de diseño del controlador siguiendo el mismo procedimiento y según

lo requiera el sistema, tomando en cuenta de manera especial, el tiempo de asentamiento del mismo; dichos resultados están registrados en el Capítulo 7.

5.4. Propuesta 2: Control por realimentación de estados y control proporcional para el sistema completo

5.4.1. Diseño del control por realimentación de estados para el subsistema angular

Retomando el subsistema angular, su modelo en variables de estado de orden seis descrito en la Ecuación 5.13, los mismos parámetros del cuadricóptero y el mismo desempeño deseado, se procedió a diseñar un nuevo controlador, tomando en cuenta ahora, la técnica de realimentación de estados.

Al seleccionar ahora la técnica de realimentación de estados para todo el subsistema angular, es conveniente verificar que dicho subsistema es controlable y observable. Primero, para verificar que dicho subsistema es controlable, se ocupó la matriz de controlabilidad:

$$Ctrb = \begin{bmatrix} B & AB & A^2B & ... & A^{n-1}B \end{bmatrix};\ n \triangleq orden\ del\ sistema \tag{5.27}$$

Como resultado se obtuvo que:

$$rango(Ctrb) = 6 = orden\ del\ sistema$$

Por lo tanto, el subsistema angular es controlable.

Como segundo paso, para verificar que el subsistema fuera observable, en caso de que alguno de los estados no se pudiera conocer directamente de los sensores. Se utilizó la matriz de observabilidad siguiente:

$$Obsv = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix} \tag{5.28}$$

Como resultado se obtuvo que:

$$rango(Obsv) = 6 = orden\ del\ sistema$$

Por lo tanto, el subsistema angular es observable.

El control por realimentación de estados, también se basa en la elección de polos que utilizan características de desempeño deseadas del sistema, sin embargo, a diferencia

del control PID, el lazo de realimentación no utiliza las salidas del sistema, sino que realimenta los estados del sistema como se muestra en la Figura 5.10.

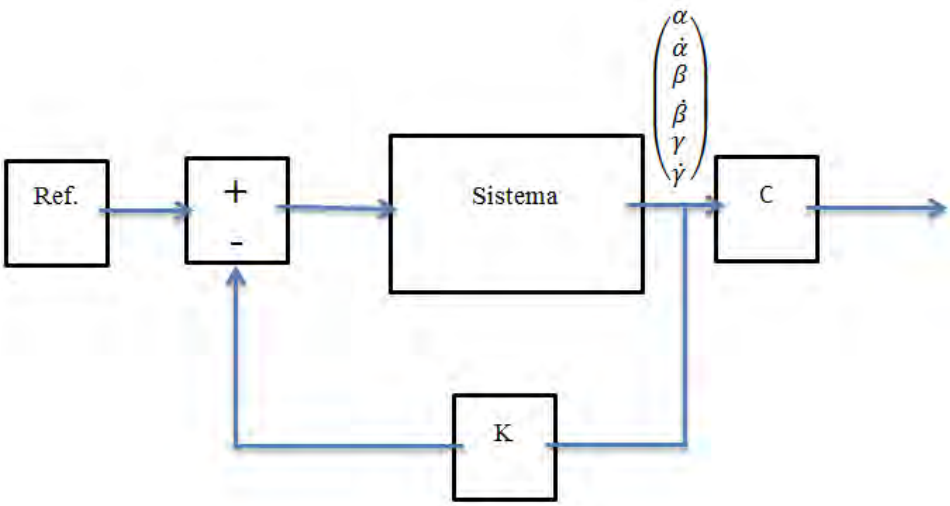


Figura 5.10: Realimentación de estados

Una de las posibles ventajas del control por realimentación de estados es que los grados de libertad del controlador, son iguales al número de variables de estado que se realimentan, es decir, que existe una constante o matriz de constantes de realimentación de dimensión proporcional al número de estados que permite modificar el comportamiento de la respuesta con base en la selección del mismo número de polos y si el sistema es de orden mayor que dos, se eligen los polos de manera similar al procedimiento realizado para obtener las constantes en el PID, donde los dos primeros atienden a una ecuación característica deseada de lazo cerrado y los restantes al criterio en el que se eligen polos no dominantes, sobre el eje real negativo. Dado que el orden del sistema del modelo de la Ecuación 5.13 es seis, se eligió como se mencionó anteriormente, el mismo desempeño de la respuesta deseada del PID para los dos primeros polos (dominantes) y los cuatro polos restantes se eligieron con base en la respuesta del sistema en simulación. La selección final fue la siguiente:

$$P_{deseados} = [-4 \pm 4.195j,\ -144.2,\ -144.21,\ -144.22,\ -144.23]$$

Con los polos definidos, se procedió a analizar el sistema en lazo cerrado; la ecuación característica deseada se comparó con la ecuación característica de lazo cerrado y se resolvieron las constantes de esta última en MatLab®, las cuales forman parte del arreglo K, mostrado a continuación:

$$K = \begin{bmatrix} 1411.0 & 31.4 & 1228.1 & 8.5 & -841.5 & -5.8 \\ 1330.1 & 9.2 & 1239.8 & 30.2 & -741.6 & -5.1 \\ -306.9 & -2.1 & -405.1 & -2.8 & 284.7 & 11.6 \end{bmatrix}$$

Sin embargo, según la teoría del control por realimentación de estados, dicho controlador cumple por sí solo, únicamente los objetivos de estabilidad y desempeño, sin embargo para el objetivo de regulación y seguimiento, es necesario agregar un grado más de libertad, el cual se encuentra en una constante de precompensación que toma por nombre $K0$ como se muestra en la Figura 5.11.

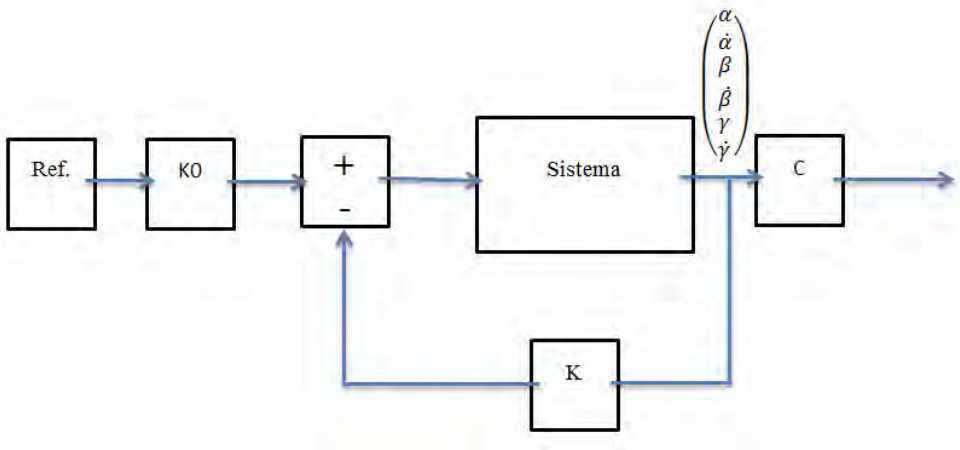


Figura 5.11: Realimentación de estados con precompensación

Esta constante de precompensación se calculó con la ecuación:

$$K0 = \left[C(-A + BK)^{-1}B\right]^{-1} \tag{5.29}$$

Que proviene de obtener la función de transferencia del sistema desde la entrada de referencia previa al precompensador hasta la salida del sistema. Dando como resultado la siguiente matriz de constantes:

$$K0 = \begin{bmatrix} 1411.0 & 1228.1 & -841.5 \\ 1330.1 & 1239.8 & -741.6 \\ -306.9 & -405.1 & 284.7 \end{bmatrix}$$

Quedando así, completamente definidos los bloques del control por realimentación de estados para el subsistema angular. De manera similar, se puede seguir nuevamente el mismo procedimiento de selección de polos y cálculo de constantes, pero con características del desempeño de la respuesta diferen-tes, así como polos no dominantes distintos sobre el eje real, para determinar nuevas constantes de realimentación y precompensación dependiendo del comportamiento de la respuesta del sistema.

5.4.2. Diseño del control proporcional para el subsistema lineal

Con el control por realimentación de estados, se cumplen los objetivos planteados para el subsistema angular, sin embargo, recordando que existe también el subsistema de movimientos lineales, es necesario atender el control de éste. Para tal efecto, se diseñó un controlador proporcional bajo las siguientes suposiciones:

- El subsistema angular responde lo suficientemente rápido como para asegurar la estabilidad angular del cuadricóptero en todos los puntos de la trayectoria definida por el subsistema de movimientos lineales
- Se considera como punto de operación del cuadricóptero, un intervalo cerca del punto en el que éste flota en el aire, es decir, ángulos cercanos a cero. Por lo que el coseno del ángulo se considera casi igual a uno y el seno del ángulo se considera casi igual a cero, es decir al mismo ángulo, por lo que $cos(ang) \approx 1$ y $sen(ang) \approx ang$
- Los ángulos necesarios para el subsistema lineal son aproximadamente iguales a los ángulos deseados del subsistema angular.

Por lo anterior, evaluando las funciones seno y coseno con las consideraciones mencionadas, el modelo del subsistema lineal queda como sigue:

$$\dot{M}_L = \begin{pmatrix} \dot{x} \\ (\overset{\nearrow 1}{cos}\alpha \overset{\nearrow \beta}{sin}\beta \overset{\nearrow 1}{cos}\gamma + \overset{\nearrow \alpha}{sin}\alpha \overset{\nearrow \gamma}{sin}\gamma)u_1/m_a \\ \dot{y} \\ (\overset{\nearrow 1}{cos}\alpha \overset{\nearrow \beta}{sin}\beta \overset{\nearrow \gamma}{sin}\gamma - \overset{\nearrow \alpha}{sin}\alpha \overset{\nearrow 1}{cos}\gamma)u_1/m_a \\ \dot{z} \\ -G + (\overset{\nearrow 1}{cos}\alpha \overset{\nearrow 1}{cos}\beta)u_1/m_a \end{pmatrix} \tag{5.30}$$

$$\dot{M}_L = \begin{Bmatrix} \dot{x} \\ (\beta + \alpha \cdot \gamma)u_1/m_a \\ \dot{y} \\ (\beta \cdot \gamma - \alpha)u_1/m_a \\ \dot{z} \\ -G + u_1/m_a \end{Bmatrix} \tag{5.31}$$

Además, con base en este nuevo modelo, se crearon tres entradas "artificiales" $\tilde{u}_1$, $\tilde{u}_2$ y $\tilde{u}_3$ que provienen del cálculo del control proporcional y que sirven para el cálculo de la entrada u_1 del subsistema lineal así como para el cálculo de los ángulos deseados, con los cuales se alimenta la referencia del subsistema angular y se definen las variables del subsistema lineal, como en [35].

$$\dot{M}_L = \left\{ \begin{array}{c} \dot{x} \\ (\beta_d + \alpha_d \cdot \gamma_d)u_1/m_a = \tilde{u}_1 \\ \dot{y} \\ (\beta_d \cdot \gamma_d - \alpha_d)u_1/m_a = \tilde{u}_2 \\ \dot{z} \\ -G + u_1/m_a = \tilde{u}_3 \end{array} \right\} \tag{5.32}$$

Así, se genera entonces el control proporcional con base en el error dado por la diferencia entre las velocidades lineales medidas y las velocidades de referencia del sistema multiplicadas por una constante proporcional como se detalla a continuación:

$$\begin{array}{c} \tilde{u_1} = kp_1 \cdot (\dot{x_d} - \dot{x}) \\ \tilde{u_2} = kp_2 \cdot (\dot{y_d} - \dot{y}) \\ \tilde{u_3} = kp_3 \cdot (\dot{z_d} - \dot{z}) \end{array} \tag{5.33}$$

Aquí cabe mencionar que las velocidades lineales que sirven de referencia para este controlador, son en la realidad, velocidades comandadas por un control remoto tal como funciona un helicóptero de radio control a escala en donde se le indica al helicóptero moverse con una velocidad determinada por el "joystick" en una cierta dirección, mientras el control interno del helicóptero realiza los cálculos necesarios para incrementar o decrementar la velocidad de los rotores para inclinar el helicoptero o el rotor según el ángulo requerido para producir tal efecto.

Considerando a $\gamma = 0$ por la baja confiabilidad en el valor obtenido del sensor para este ángulo, como se mencionó anteriormente, se puede entender que el cuadricóptero no cambia su orientación al no tener movimiento de guiño. Así entonces, las ecuaciones correspondientes a las entradas artificiales del modelo de la Ecuación 5.32:

$$\begin{array}{c} (\beta_d)u_1/m_a = \tilde{u_1} \\ (-\alpha_d)u_1/m_a = \tilde{u_2} \\ -G + u_1/m_a = \tilde{u_3} \end{array} \tag{5.34}$$

Además, ya que de estos se pueden obtener los ángulos deseados para utilizarlos como referencia en el subsistema angular, se procede a despejar las variables de interés como lo son u_1, α_d, y β_d.

$$\begin{array}{c} u_1 = m_a \cdot (\tilde{u_3} + G) \\ \alpha_d = \frac{-\tilde{u_2}}{\tilde{u_3}+G} \\ \beta_d = \frac{\tilde{u_1}}{\tilde{u_3}+G} \\ \gamma_d = 0 \end{array} \tag{5.35}$$

Finalmente, las constantes kp_1, kp_2 y kp_3 se definieron para una primera aproximación con valor unitario, el cual, se puede modificar basado en simulaciones y el comportamien-

to de la respuesta de este controlador, tomando en cuenta que se desea que el control proporcional tenga un tiempo de asentamiento mayor al del control por realimentación de estados del subsistema de movimientos angulares para asegurar la consideración en la que los ángulos deseados son iguales a los ángulos medidos.

$$kp_1 = 1$$
$$kp_2 = 1$$
$$kp_3 = 1$$

5.4.3. Simulación del sistema completo con control

Habiendo diseñado el nuevo controlador, se procedió a verificar funcionamiento en el sistema del cuadricóptero, nuevamente por medio de una simulación en el programa MatLab® [34] con ayuda de su biblioteca SimuLink®. El diagrama de bloques final que se utilizó para realizar las pruebas de simulación fue el mostrado en la Figura 5.12.

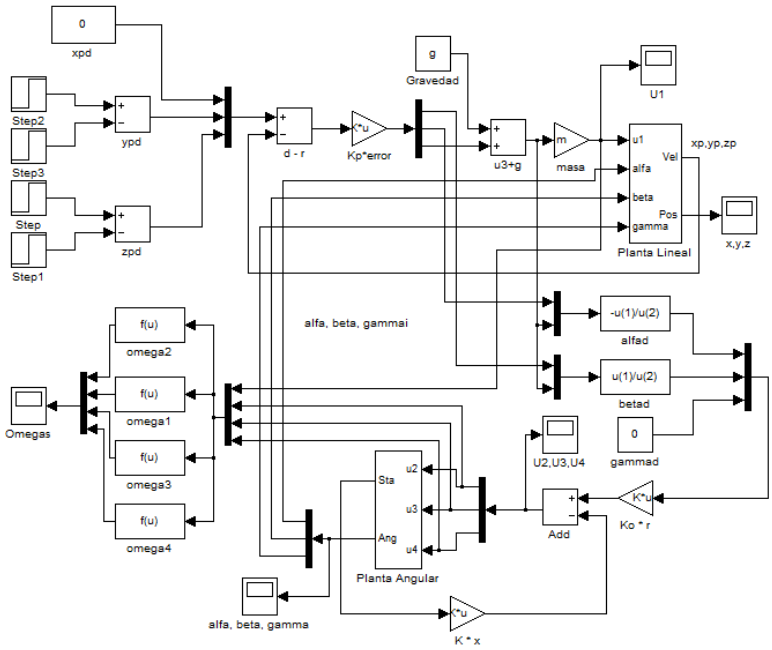


Figura 5.12: Control en bloques detallado

5.4.4. Resultados de la simulación con control de estabilidad

Se obtuvieron gráficas, primero para observar la estabilidad de los ángulos y las posiciones lineales, las entradas del sistema y las velocidades de giro de cada rotor ante una referencia con condición inicial diferente de cero y segundo para la regulación de dichas variables ante una referencia variable con condición inicial igual a cero. Para la estabilidad, y con la condición inicial del ángulo α igual a 0.05 rad así como la posición (x, y, z) en (5, 5, 5), se obtuvo para la respuesta angular la Figura 5.13 en la que se observa el efecto

que produce el movimiento de un ángulo en los demás, así como que después de 1 s, los ángulos convergen a cero comprobando así la estabilidad del sistema.

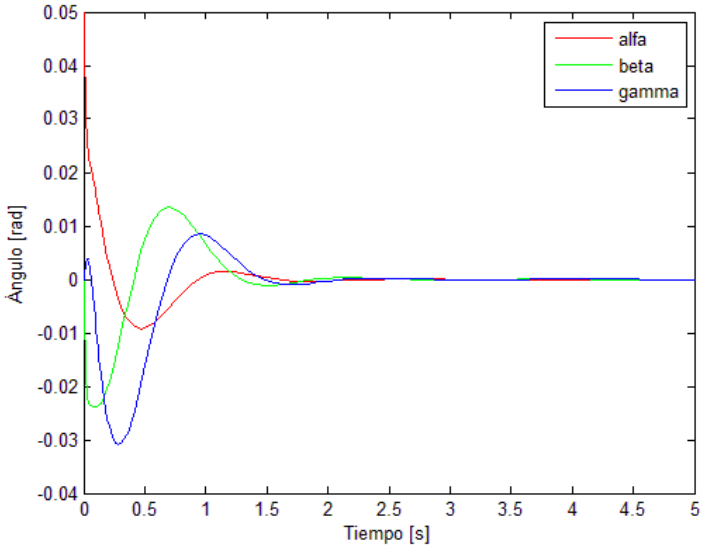


Figura 5.13: Respuesta angular para la estabilidad

De manera similar, en la Figura 5.14, se observa el comportamiento de las fuerzas relacionadas con los movimientos de alabeo, cabeceo y guiño en los que también se produce un efecto de estabilidad en un tiempo aproximado de 0.1 s.

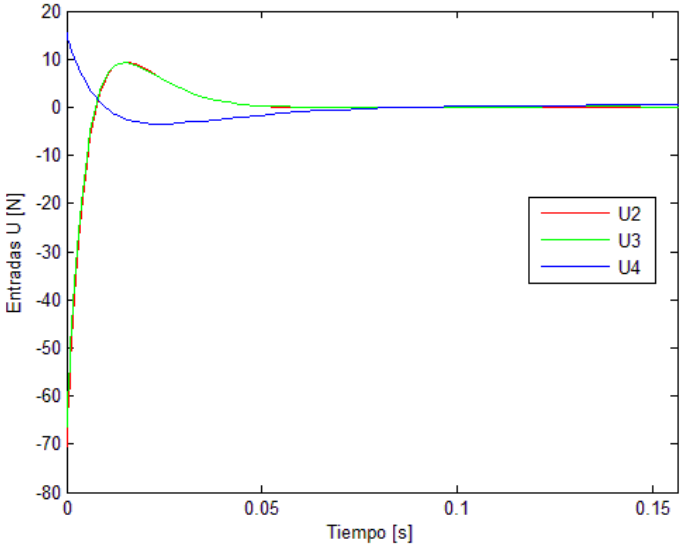


Figura 5.14: Entradas u_2, u_3, u_4 para la estabilidad

En la Figura 5.15, se observa la respuesta de la entrada u_1, la cual está relacionada directamente con la fuerza vertical total requerida para mantener el cuadricóptero en el aire, es decir, para compensar el peso del mismo. Se observa que ante la perturbación

angular, existe un intento por parte de la entrada u_1 de compensar la caída o la pérdida de una componente de la fuerza al estar inclinado el cuadricóptero y después de cierto tiempo, también se estabiliza al valor del peso del cuadricóptero.

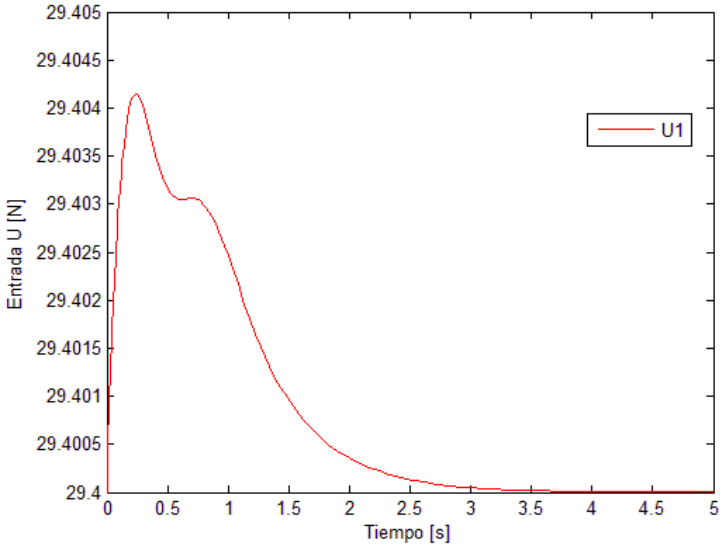


Figura 5.15: Entrada u_1

La Figura 5.16 representa el comportamiento de las variables relacionadas con la posición del cuadricóptero, con lo que se puede observar claramente el desplazamiento en el eje y debido a la variación del ángulo α, así como un ligero desplazamiento en el eje x causado por ligeras variaciones en el ángulo β. Y por último, un desplazamiento casi nulo en el eje z provocado solo por la descompensación de la dirección del vector de empuje vertical que se encarga del peso del cuadricóptero y por ende a la variación de la entrada u_1.

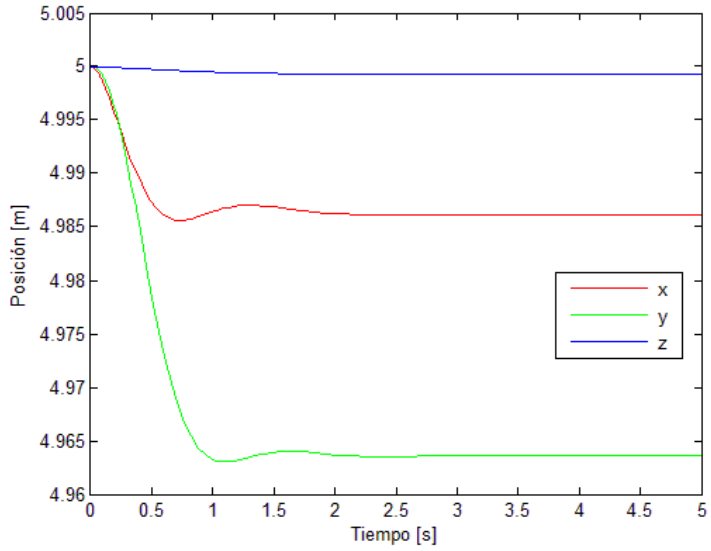


Figura 5.16: Posición para la estabilidad

Aunque el desplazamiento más significativo es de cerca de 0.04 m, es considerable tomando en cuenta el valor de la condición inicial para α y que el efecto producido en el eje y corresponde al comportamiento esperado para dicho ángulo. La condición inicial para x, y, z se utilizó solo para poder observar desplazamientos en el cuadrante positivo.

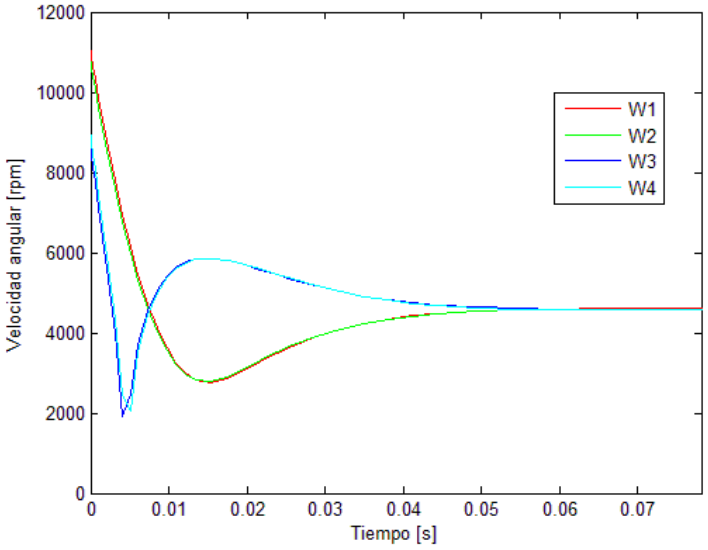


Figura 5.17: Respuesta de las velocidades angulares para estabilidad

Finalmente la gráfica de la Figura 5.17, muestra cómo evolucionan las velocidades angulares de cada rotor de acuerdo a los valores de u_1,u_2,u_3 y u_4. Aquí se puede observar que la variación de dichas velocidades se mantiene dentro del rango de valores de revoluciones proporcionadas por los motores, sin embargo, puede ocurrir que debido a las inercias que se deben vencer, estos valores sobrepasen el permitido por los motores, los que por medio del programa del microcontrolador se pueden limitar a un valor seguro de operación que puede afectar en la respuesta del sistema pero permite disminuir el riesgo de demandar a los motores mayor velocidad de la que pueden proporcionar. En esta figura se puede observar la estabilidad del sistema para las velocidades angulares de los rotores al paso del tiempo.

5.4.5. Resultados de la simulación con control de regulación

De manera similar a la anterior, pero para el objetivo de control de regulación, se realizaron simulaciones con condiciones iniciales angulares iguales a cero y el mismo punto de origen (5, 5, 5) para (x, y, z); pero con entradas al controlador diferentes de cero, es decir que las variables x_d, y_d y z_d toman el lugar de la referencia general del algoritmo de control, estas variables, en la realidad, son equivalentes al valor del *joystick* del control remoto con el que se puede manipular el cuadricóptero como se mencionó previamente. Éstas le indican la rapidez con la que se desea que se mueva en una determinada dirección; dentro

del controlador, lo anterior se traduce en los ángulos deseados antes mencionados. Para la simulación, se eligió una señal escalón unitario.

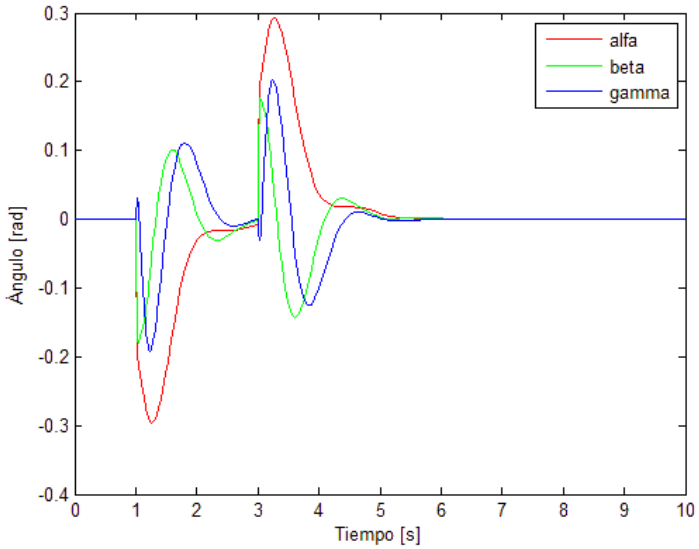


Figura 5.18: Respuesta angular para regulación

Esta señal se introdujo a diferentes tiempos a la variable y_d y z_d, pensando en producir movimiento secuenciales en los ejes correspondientes y al apagar la señal, observar la estabilidad el cuadricóptero en el nuevo punto de operación. La primer señal, para y_d se enciende en el segundo 1 después de iniciada la simulación y regresa a cero en el segundo 3; la segunda señal, para z_d se enciende en el segundo 5 y regresa a cero en el segundo 8.

La Figura 5.18 muestra el comportamiento de las posiciones angulares para la señal descrita, se observa que al momento de apagar la señal para y_d, probablemente por las inercias del modelo, los ángulos intentan contrarrestar el efecto producido al encender la señal, además, para el intervalo entre 5 s y 8 s que corresponde al movimiento en el eje z, no existe un cambio significativo en los ángulos, lo que sugiere que para cambios de altura, los ángulos se mantienen estables y constantes.
En la Figura 5.19 se muestra el comportamiento de las entradas u_2, u_3 y u_4 que de manera similar a la respuesta angular, las entradas relacionadas con los movimientos de alabeo, cabeceo y guiño, solo se ven afectadas en el intervalo en que actúa la primer señal que causa un efecto en y_d, más no en el momento del cambio de altura exclusivamente.

Los valores pico de las entradas, pueden deberse principalmente a que la señal de referencia es una señal escalón, lo que significa un cambio brusco instantáneo en el estado de dicha referencia. En caso de requerirlo, estos valores también se pueden limitar desde la programación para evitar daños a los ESC o a los actuadores.

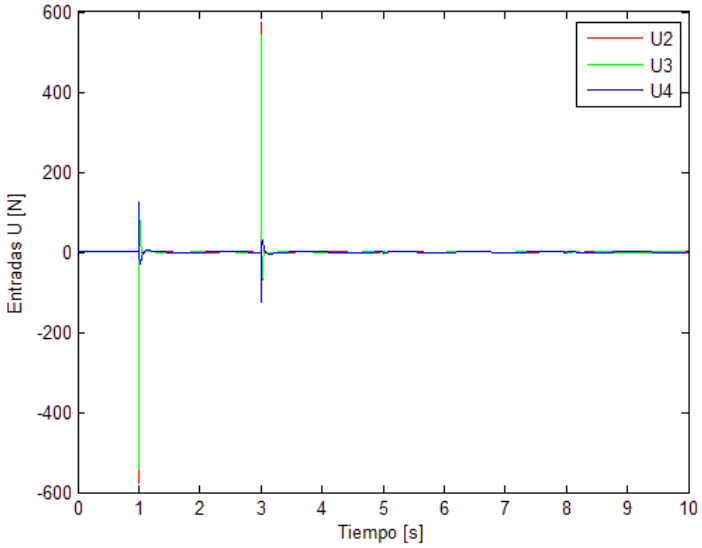


Figura 5.19: Respuesta de las entradas u_2, u_3 y u_4 para regulación

Al igual que en la simulación anterior, u_1 representa la fuerza que compensa el peso del cuadricóptero y su movimiento sobre z, en la gráfica de la Figura 5.20 se muestra su evolución en el tiempo ante las nuevas entradas, con lo que se puede observar con mayor claridad, el papel de la inercia en los cambios de altura significativos.

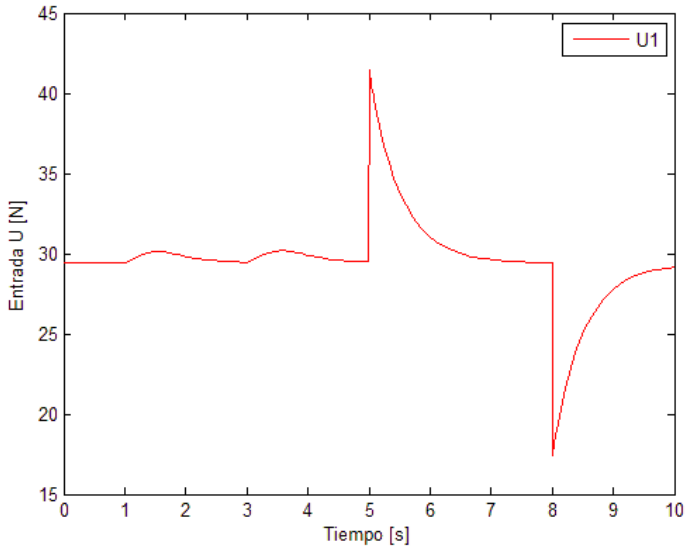


Figura 5.20: Respuesta de u_1 para regulación

En la Figura 5.21 se muestra el comportamiento de la posición del cuadricóptero ante las entradas variables aplicadas en distintos momentos, además, como dato interesante, se encuentra que el cuadricóptero, después de ser movido de su punto original de operación, no intenta regresar a este, sino que estabiliza sus estados en el nuevo punto de operación alcanzado; aunque pueda ser evidente desde el punto de vista de control es conveniente

comentar que ayuda a comprender el comportamiento real de las acciones tomadas por el cuadricóptero.

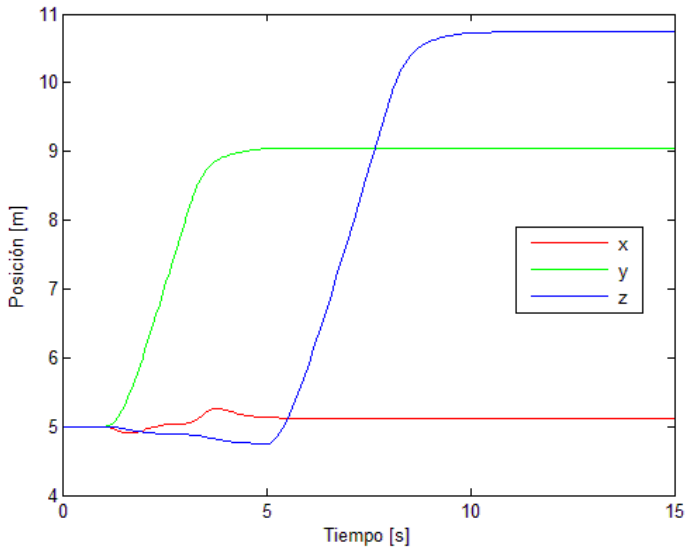


Figura 5.21: Respuesta de la posición para regulación

La Figura 5.22 muestra el comportamiento de las velocidades angulares de cada rotor, que de manera similar a las entradas del sistema, se ven afectadas por el cambio brusco que genera el encendido y el apagado de una referencia escalón, sin embargo, con el paso del tiempo, se observa su estabilidad, así como su ligero incremento y decremento correspondiente para el cambio de altura entre el tiempo 5 s y 8 s, más la compensación inercial.

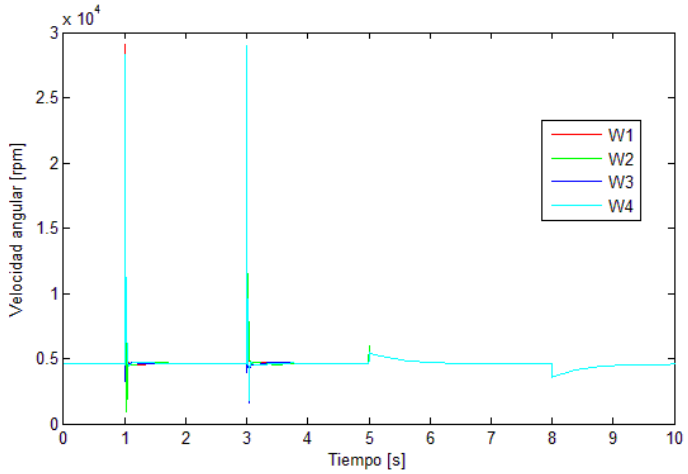


Figura 5.22: Respuesta de las velocidades angulares de cada rotor para regulación

Finalmente, la Figura 5.23 muestra en tres dimensiones, la trayectoria del cuadricóptero para la simulación de regulación con base en los datos de la Figura 5.21. En esta última figura, se observa de manera más clara el comportamiento del cuadricóptero cuando la

señal de entrada de velocidad deseada regresa a cero y el cuadricóptero se estabiliza en ese nuevo punto de operación.

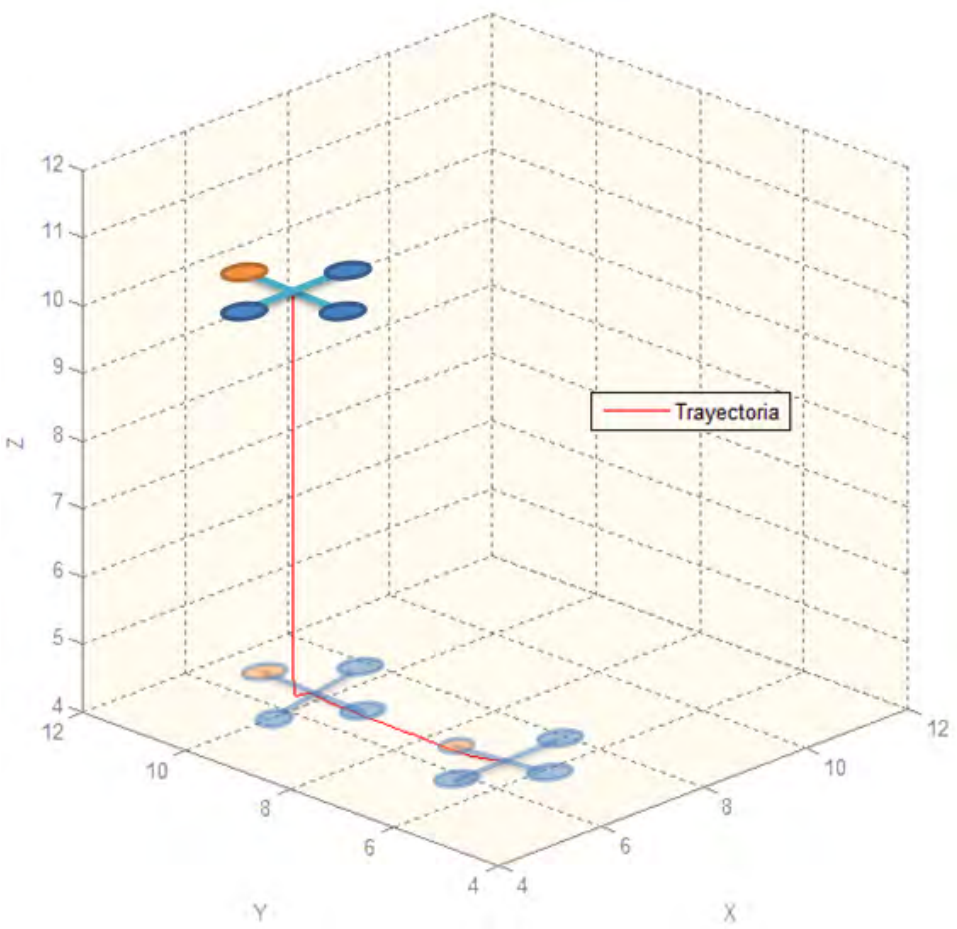


Figura 5.23: Trayectoria en 3D del cuadricóptero

Capítulo 6

Programación del control PID

El programa completo programado en el ambiente Arduino®, que se utilizó para realizar algunas pruebas con el cuadricóptero integra la inicialización de las bibliotecas, la configuración de la comunicación, el arranque de motores, la lectura del sensor y la ley de control PID entre otros. El algoritmo está basado en el diagrama de flujo de la Figura 6.1 ubicada al final del presente capítulo.

6.1. Bibliotecas

Como se mencionó anteriormente, los sensores que sirven en este proyecto, dentro del cumplimiento del objetivo de estabilidad, para determinar el valor de los ángulos, son los que se encuentran integrados en la *IMU*. La lectura de estos sensores se hace por medio del microcontrolador en el Arduino Mega vía I^2C en una rutina programada que solicita información a los registros de cada sensor de la *IMU*. Esta rutina es una aportación de [36] utilizada para obtener los datos de la *IMU* presentada en ese trabajo, pero que es compatible con el hardware con el que se trabaja en este proyecto.

Las bibliotecas utilizadas para leer e integrar los datos se obtuvieron de [37], y forman parte del mismo proyecto que [36]. Las encargadas de leer información de cada sensor son: *ADXL345.h*, *HMC58X3.h* e *ITG3200.h*. La biblioteca utilizada para fusionar los datos de la IMU tiene por nombre *FreeIMU.h*, la cual contiene un algoritmo especial de integración de información conocido como filtro DCM (*Direction Cosine Matrix*) con el que se calculan los ángulos de navegación a partir de los datos del girómetro y del acelerómetro como se explica en [36]. Dicho algoritmo junto con otros programas relevantes, de estas bibliotecas, se encuentra en el *Apéndice D: Programas*. La biblioteca *FreeIMU.h* contiene además, una función llamada "*getYawPitchRoll*" que almacena en un vector de tres elementos el valor de cada ángulo de navegación a partir del cálculo de la función arcotangente y arcotangente2 cuya base está explicada en [12]. La función *getYawPitchRoll*

se muestra en el siguiente algoritmo[1]:

```
void FreeIMU::getYawPitchRoll(float * ypr) {
  float q[4];
  float gx, gy, gz;
  getQ(q);
  gx = 2 * (q[1]*q[3] - q[0]*q[2]);
  gy = 2 * (q[0]*q[1] + q[2]*q[3]);
  gz = q[0]*q[0] - q[1]*q[1] - q[2]*q[2] + q[3]*q[3];
  ypr[0] = atan2(2 * q[1] * q[2] - 2 * q[0] * q[3],
     2 * q[0]*q[0] + 2 * q[1] * q[1] - 1) * 180/M_PI;
  ypr[1] = atan(gx / sqrt(gy*gy + gz*gz))  * 180/M_PI;
  ypr[2] = atan(gy / sqrt(gx*gx + gz*gz))  * 180/M_PI;
}
```

Para complementar la funcionalidad del programa, se hizo uso de otras bibliotecas provenientes también de [37] como: *DebugUtils.h* que se encarga de algunos parámetros de corrección y funcionamiento de la biblioteca *FreeIMU.h* y la biblioteca *CommunicationUtils.h* que se encarga del protocolo de comunicación entre las bibliotecas de los sensores y las que integran y almacenan la información en variables útiles. Por parte de la comunidad de Arduino® [25], se hizo uso de bibliotecas como: *Wire.h*, encargada del protocolo y configuración de la comunicación I^2C, la biblioteca *Servo.h*, encargada de escribir en el puerto de salida digital un PWM originalmente destinado a controlar servomotores, los cuales utilizan la misma señal de control que los ESC para variar la velocidad de los motores del cuadricóptero, dicha señal puede tomar valores entre los 0 y los 2500 μs.

6.2. Variables involucradas

Se definieron las variables globales necesarias para almacenar datos en el programa como las contstantes PID, las entradas u_1, u_2, u_3 y u_4, los ángulos α y β, entre otras. Dichas variables, al ser globales se definieron al inicio del programa para poder ser usadas en todas las funciones del mismo. También se definieron variables necesarias para crear los objetos utilizados por bibliotecas como *Servo.h* y *FreeIMU.h*.

6.3. Configuración inicial

A continuación se muestra la sección del programa completo en donde se inicializan los protocolos de comunicación serial, I^2C, la IMU y la comunicación con los ESC variando el tiempo de la amplitud de la señal desde cero hasta un valor en donde los ESC reconocen una entrada inalámbrica válida pero sin encender los motores, que para este caso es de 0.9 ms.

[1]El texto de los códigos de programa está escrito intencionalmente sin acentos ni 'ñ' por funcionalidad del compilador.

```
void setup(){
  Serial2.begin(9600);    //Inicializar comunicacion con XBee
  Wire.begin();      //Inicializar comunicacion con IMU
  M1.attach(2);      //Asignacion de puertos para cada motor
  M2.attach(3);
  M3.attach(4);
  M4.attach(5);
  delay(5);
  sensorIMU.init();      //Inicializar IMU
  delay(5);

  //***********Inicio de comunicacion con los ESC************//

  for(i=1; i<=((limite-150)/10); i++){
    vel_ini=vel_ini+10;
    M1.writeMicroseconds(vel_ini);
    M2.writeMicroseconds(vel_ini);
    M3.writeMicroseconds(vel_ini);
    M4.writeMicroseconds(vel_ini);
    delay(15);
  }
delay(1000);
  for(i=1; i<=15; i++){
    vel_ini=vel_ini+10;
    M1.writeMicroseconds(vel_ini);
    M2.writeMicroseconds(vel_ini);
    M3.writeMicroseconds(vel_ini);
    M4.writeMicroseconds(vel_ini);
    delay(15);
  }
}
```

6.4. Ciclo inicial de espera

Terminando la rutina de configuración, se procede a entrar al ciclo principal del programa, el cual se repetirá mientras esté encendido el microcontrolador. Este ciclo comienza con otro ciclo interno que no ejecuta ninguna acción más, que leer del puerto serial comunicado con el XBee® a la computadora el caracter de arranque, el cual, de ser igual a la letra 'A', sería la única condición para terminar con dicho ciclo, dando paso al arranque del cuadricóptero y su posterior ciclo de control.

6.5. Arranque e incremento de velocidad en los motores

En el momento en el que el cuadricóptero recibe el caracter 'A' desde la computadora, sale del ciclo de espera y comienza una rutina de incremento de velocidad en los motores similar a la que se encuentra en la configuración inicial. Este incremento es lento, de manera que el usuario lo pueda detener enviando el caracter 'S', en cuanto el cuadricóptero se eleve y esté listo para comenzar el ciclo de control principal, 'P' si el usuario quiere detener la ejecución del programa y regresar al ciclo inicial de espera o, si el usuario deja pasar la rutina completa de incremento de velocidad, esta eleva la velocidad de los motores a un valor cercano o dentro del rango de operación del control para evitar sobresaltos en la acción de empuje vertical.

6.6. Ciclo principal

Al concluir la rutina de incremento de velocidad, se prosige con las siguientes rutinas de programación, en las que se encuentran las acciones principales ejecutadas para obtener la información de los sensores y estabilizar los ángulos del cuadricóptero. Es el lazo de control principal que contiene el algoritmo PID y el cálculo de las velocidades angulares de cada motor. Comienza con una primera medición de los ángulos y el ciclo sólo es detenido al enviar el caracter 'P', que es una señal de paro enviada desde la computadora.

6.6.1. Cálculo del tiempo entre cada ciclo principal

En el siguiente algoritmo, se calcula el tiempo que tarda en pasar cada ciclo de control para poder realizar la derivada y la integral del error en el PID. Durante las últimas pruebas realizadas, este tiempo oscilaba entre los 180 ms y 210 ms.

```
ahora=millis();
    tiempo=(ahora-transcurridos)*0.001;   //Tiempo entre cada ciclo
    delay(25);   //Retardo necesario previo a la lectura
                 //de la IMU sin bloquear el ciclo
```

6.6.2. Lectura del sensor

Recordando el funcionamiento de la IMU, se sabe que del girómetro, se obtiene información de la velocidad angular que se traduce, por medio de algunos cálculos, a la posición angular en sus tres ejes de medición; del acelerómetro, se obtiene información de aceleraciones estáticas y dinámicas también en sus tres ejes de medición, con lo que se puede detectar cuando el dispositivo está en reposo, caída libre o movimiento acelerado. Del magnetómetro se obtiene información de referencia con respecto a los ejes magnéticos de la tierra y esto permite afinar el valor de la orientación del dispositivo. Sin embargo

para obtener los ángulos que requiere el algoritmo de control (α y β), se utiliza solo la combinación de la información del acelerómetro y del girómetro.

Como se mencionó anteriormente, la biblioteca que integra estos sensores, lo hace por medio de un algoritmo denominado *filtro DCM*, sin embargo, otra manera de integrar estos sensores para obtener los ángulos requeridos, se encuentra en [18] por medio de un *filtro Kalman*. La función que devuelve el valor de los ángulos calculados se llama *getYawPitchRoll*, la cual tiene como argumento el vector de tres elementos: *ypr*, en el que se almacenan dichos valores, de los cuales, sólo se utilizan las últimas dos localidades que contienen el valor del ángulo β y α respectivamente.

6.6.3. Filtro atenuador

Posterior a la obtención de información del sensor y al cálculo de los ángulos de navegación, debido al ruido mecánico, eléctrico, entre otros, que pudiera afectar las mediciones y los datos que se ingresan a la ley de control, se programó la función *smooth* que contiene un filtro tipo paso bajas discreto que funciona con base en la teoría de alisado exponencial explicada en [18] y cuyo algoritmo se muestra a continuación:

```
double smooth(double data,
     double filterVal, double smoothedVal){
  if (filterVal > 1){
    filterVal = .99;
  }
  else if (filterVal <= 0){
    filterVal = 0;
  }
  smoothedVal = (data*(1-filterVal))+
     (smoothedVal*filterVal);
  return smoothedVal;
}
```

El funcionamiento básico de esta función es dar un porcentaje o peso a la medición actual y el complemento a la medición anterior, con esto se puede dar cierta importancia a la nueva medición dependiendo del ruido o las constantes variaciones que ésta pueda tener y que se deseen atenuar.

La anterior función se programó fuera del "*loop*" principal de arduino, sin embargo, se manda llamar desde el interior del ciclo principal de control con una rutina en la que se asigna en una nueva variable la salida de la función de alisamiento para cada ángulo medido.

6.6.4. Control: Estabilización de ángulos

Como se mencionó en el Capítulo 5, se programó una sola función de control PID que sirve para estabilizar los ángulos de navegación α y β, recordando que para γ, en esta

serie de pruebas, tiene el valor de cero. Sin embargo, existen dos funciones diferentes para cada ángulo (*estabiliza_roll* y *estabiliza_pitch*), que tienen como objetivo, calcular el error y almacenar en una variable el resultado de la ley de control. El algoritmo del control PID calcula de manera discreta una integral y una derivada del error a partir de su valor anterior, el valor actual y el tiempo entre cada medición. Se realizan las respectivas multiplicaciones por las constantes PID y se suman las respuestas de cada etapa en una sola que es la que se devuelve como el resultado de la ley de control.
De manera similar a la función de atenuación o alisamiento, las funciones de *estabiliza_roll* y *estabiliza_pitch* se encuentran fuera del ciclo de control principal, sin embargo, se accede a estas funciones dentro del ciclo de control al momento de calcular las entradas del sistema.

La función que contiene el algoritmo del control PID a la que manda llamar las funciones *estabiliza_roll* y *estabiliza_pitch*, también se encuentra fuera del ciclo de control principal. La acción del control derivativo también es atenuada por la función *smooth* para casos en los que hay cambios bruscos en el error y la acción del control integral, está limitada para valores del error menores a 10°.

6.6.5. Cálculo y escritura de velocidades angulares para cada motor

Para el cálculo de las salidas en microsegundos, se hizo uso de las Ecuaciones 5.6 y 5.2 para determinar el valor en rpm que debe tener cada motor de acuerdo al valor en un instante dado de las entradas u_1, u_2, u_3 y u_4, y se programó su respectiva ecuación.

Para la escritura de las velocidades angulares de cada motor, se utilizó la función *map* de arduino para escalar la acción de control a valores válidos para la escritura en microsegundos del tiempo del pulso para los ESC. Además, se agregó un "*offset*" para cada motor, dependiendo de su comportamiento basado en las pruebas realizadas.

Con esta rutina de escritura de velocidades en motores, se cierra el ciclo de control principal y regresa a una nueva medición del ángulo mientras no se le ordene que pare el ciclo por medio de la condición antes descritas.

6.7. Rutina de paro

Cuando el ciclo de control principal recibe la señal de paro, se procede a disminuir las velocidades de los motores hasta pararlos, sin salir del rango de señal válida para los ESC, para este caso, basta con escribir en los ESC un ancho de pulso de tiempo igual a 0.9 ms para detener los motores de manera segura.

Al terminar esta rutina, el "*loop*" principal de arduino, regresa a la función de espera, lista para recibir una nueva orden de arranque.

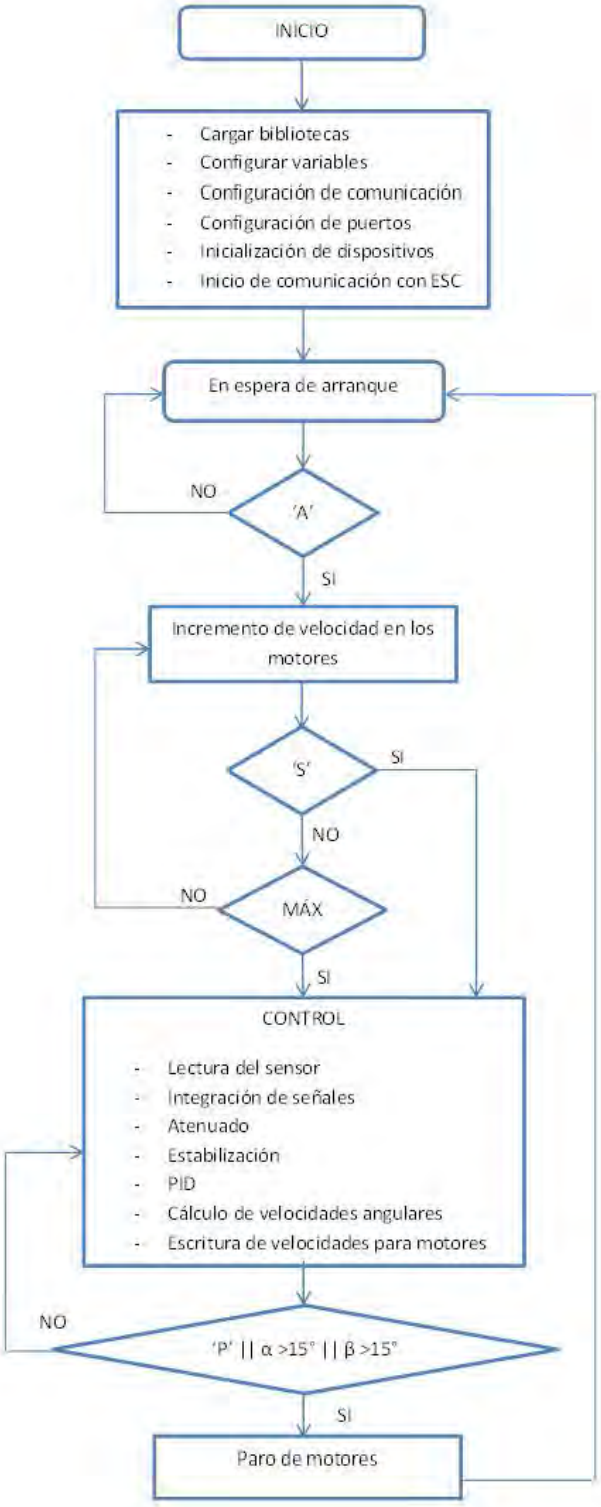


Figura 6.1: Diagrama de flujo del programa principal

Capítulo 7

Pruebas y resultados

El presente capítulo muestra las pruebas realizadas con el prototipo de cuadricóptero, incluyendo la implementación del algoritmo de control y los algoritmos de arranque y paro antes mencionados. Primero, se desarrolla una prueba de caracterización de motores en la que se espera obtener como resultado alguno a algunos datos característicos de cada motor que sirven como entrada para algunos parámetros utilizados en el desarrollo del sistema de control. Segundo, se desarrolla una serie de pruebas con los componentes del prototipo ubicados en un solo eje horizontal, incluyendo el sistema de control de estabilidad diseñado, con lo que se espera obtener una primera aproximación al comportamiento de algunas series de constantes PID.

Finalmente, se desarrollan varias pruebas con el sistema completo, es decir, con el prototipo de cuadricóptero completo y con el sistema de control PID para la estabilidad de los ángulos relacionados con los dos ejes horizontales del cuadricóptero, con lo que se espera obtener resultados que permitan concluir sobre el cumplimiento del objetivo de estabilidad planteado en un inicio.

7.1. Caracterización de motores

Procedimiento para caracterizar los motores

Las constantes relacionadas con los motores necesarias para desarrollar el sistema de control son, como se ha mencionado, b y d, que relacionan fuerza con velocidad angular de cada rotor.

Para encontrar el valor de la constante b, se diseñó un banco de pruebas con la configuración de la Figura 7.1, que se compone de un brazo de aluminio de 0.60 m de longitud, dos láminas también de aluminio que sujetan el motor en un extremo y el contrapeso de 2 kg en el otro, además de una base de madera a la que se une el brazo por medio de una barra circular de aluminio de 0.011 m de diámetro y 0.15 m de longitud colocada al centro, dándole un grado de libertad a dicho brazo.

Figura 7.1: Banco de pruebas

Con este banco de pruebas se intenta relacionar la fuerza de empuje generada por el motor y medida por medio del contrapeso al intentar eqilibrarlo con la velocidad angular del motor controlada y conocida por medio de los microsegundos del ancho de pulso enviados al ESC. Para el encendido del motor y para poder variar su velocidad con ayuda del ESC, se programó un algoritmo también en el ambiente Arduino®, usando la biblioteca *Servo.h* con la función *writeMicroseconds(),* un algoritmo en el que se hace uso de un potenciómetro conectado al convertidor analógico-digital de la tarjeta Arduino Mega, con el que se controla el valor de microsegundos que se desea escribir en el ESC, tal como aparece en el siguiente programa:

```
#include <Servo.h>

Servo ESC;   //Crea un objeto de tipo Servo
int potpin = 0;   //Entrada de un potenciometro
int val;   //Variable que almacena el valor deseado

void setup()  {
  Serial.begin(9600); //Inicio de comunicacion serial
  ESC.attach(9);   //El objeto ESC se modifica en el pin 9
}

void loop()  {
  val = analogRead(potpin); //Lectura del ADC
  val = map(val, 0, 1023, 500, 1800); //Valor deseado
  ESC.writeMicroseconds(val); //microsegundos en el ESC
  Serial.println(val); //Impresion del valor para monitoreo
}
```

Se conectó la señal de entrada del ESC al Arduino Mega® y la salida de potencia al motor. Se instalaron las hélices 12 x 3.8 y se colocó el motor de manera que al hacerlo girar, el empuje resultante estuviera en dirección hacia el piso para poder elevarel contrapeso en

el otro extremo. Se conectó la entrada de potencia del ESC a una de las baterías Rhino y se encendió el sistema, previa verificación de que el potenciómetro estuviera en su valor mínimo que equivaldría, según el algoritmo programado a una salida de 500 μs, que corresponde a un motor con velocidad inicial de 0 rpm.

Se monitoreó el valor de microsegundos que se estaba escribiendo en el ESC, así como la respuesta del rotor con respecto a poder levantar y equilibrar el peso de la cubeta con piedras. Posteriormente se asignó un número de motor y de ESC para que a partir de ese momento trabajaran juntos. Luego se reemplazaron tanto el motor como el ESC por otro par, para realizar una nueva medición siguiendo el mismo procedimiento y con el mismo programa.

Para la variable d, no se encontró forma sencilla de poderla obtener, por lo que su valor se tomó de algunos artículos que han trabajado previamente con cuadricópteros como en [14].

Resultados de la caracterización

Cuadro 7.1: Resultados de caracterizar motores

Motor	Microsegundos para equilibrar 2 kg	Velocidad angular aproximada en rpm	Peso del rotor kg	Peso real equilibrado en kg	Cte. b
1	1621	7003	0.21	1.8	$3.6x10^{-7}$
2	1636	7068	0.21	1.8	$3.5x10^{-7}$
3	1656	7154	0.21	1.8	$3.4x10^{-7}$
4	1638	7076	0.21	1.8	$3.5x10^{-7}$

Algunas observaciones importantes, realizadas a partir de las mediciones anteriores, fueron:

- El ESC requiere que la señal de entrada (ancho de pulso en microsegundos) se incremente de manera contínua entre 0 μs y 600 μs para reconocer que existe una señal de comunicación y control válida.
- El ESC emite sonidos intermitentes con una frecuencia aproximada de 5 Hz cuando no reconocen una señal de entrada válida, lo que ocurre cuando al momento de encenderlos, el ancho de pulso tiene un valor inferior a 600 μs o superior a 900 μs.
- EL ESC deja de emitir sonidos intermitentes cuando tiene una señal de entrada válida, lo que ocurre para los cuatro, al llegar al valor de entre 600 μs y 800 μs.

- El ESC emite cuatro "*bips*" cuando recibe una señal válida y útil para controlar el motor, lo que ocurre para los cuatro, al llegar al valor de 800 μs.

Las variaciones en los microsegundos necesarios para equilibrar el peso de 2 kg se compensan en la programación del microcontrolador por medio de un *offset*, al momento de probar todo el sistema completo para que el comportamiento de éste sea uniforme cercano al punto de equilibrio, este valor se modifica también dependiendo de la respuesta del cuadricóptero en las pruebas mencionadas.

7.2. Pruebas en un solo eje

Procedimiento para las pruebas en un solo eje horizontal

Posterior a la caracterización anterior, se procedió a evaluar la ley de control en el modelo real pero limitándolo a un solo grado de liberad, lo que a su vez se traduce en la intención de estabilizar un solo ángulo de navegación. Para esta prueba, se quitó el contrapeso de la prueba anterior, se colocó el segundo motor en el extremo correspondiente y se le colocaron las hélices 12 x 3.8 *Pusher*, sin embargo, ahora ambos motores están colocados de manera que al hacerlos girar, el empuje generado esté en dirección al techo para compensar el empuje de uno con el contrario y buscar la estabilidad como se muestra en la Figura 7.2.

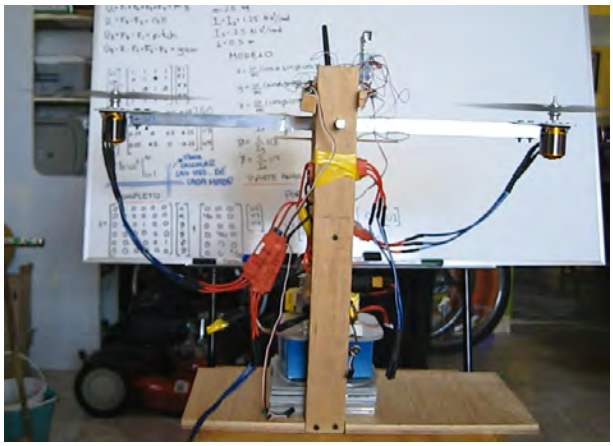


Figura 7.2: Disposición de componentes para las pruebas en un eje

Se programó el algoritmo de control mencionado en el Capítulo 6 pero controlando sólo el ángulo β. Se conectó el segundo ESC al Arduino Mega®, al motor correspondiente y a la misma alimentación de energía que el motor anterior. Se encendió el sistema y se monitoreó el valor del ángulo y el valor de los microsegundos que se escriben en cada uno de los dos ESC. Este último valor de microsegundos, estuvo limitado por programación, dependiendo del comportamiento del sistema.

Al término de una serie de pruebas para un par de motores, se desmontaron y colocaron en su lugar el otro par de motores y ESC y se realizaron las mismas pruebas. Las variables más importantes en esta serie de pruebas, fueron las constantes del PID.

Resultados para las pruebas en un solo eje horizontal

A continuación se presenta una serie de gráficas correspondientes al comportamiento de los ángulos medidos con respecto al tiempo, durante cada prueba con diferentes valores para las constantes K_p, K_i y K_d.

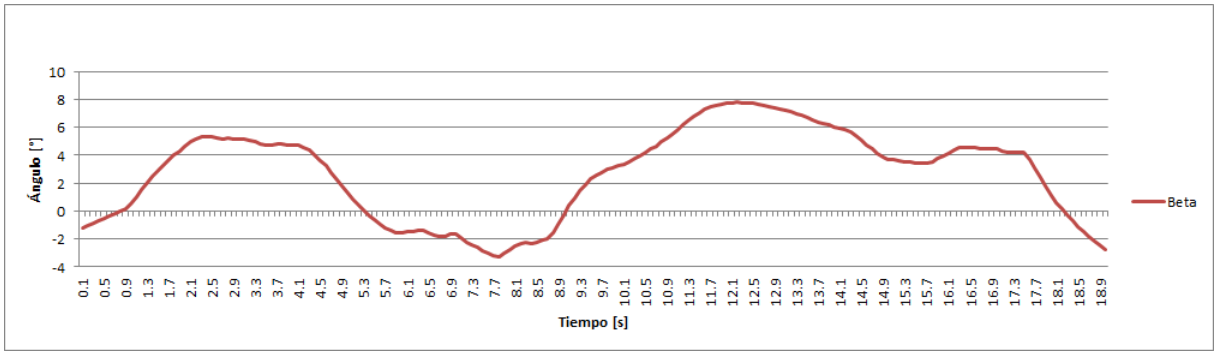


Figura 7.3: Prueba 1 de estabilidad en un eje

La Figura 7.3 muestra el ángulo β para la serie de constantes $K_p = 1323.3$, $K_i = 25147$ y $K_d = 35.93$, las cuales se obtuvieron a partir de las siguientes características del desempeño deseado: $ts = 0.2\ s$ y $sp = 5\,\%$. En esta Figura se observan cuatro instantes en los que el ángulo β es cero, pero no se observa convergencia alguna hacia este valor o algún otro ángulo. El rango de valores de β oscila entre 8 grados y -4 grados y existen variaciones máximas de cerca de 10 grados por segundo, por lo que no es un control adecuado de la estabilidad.

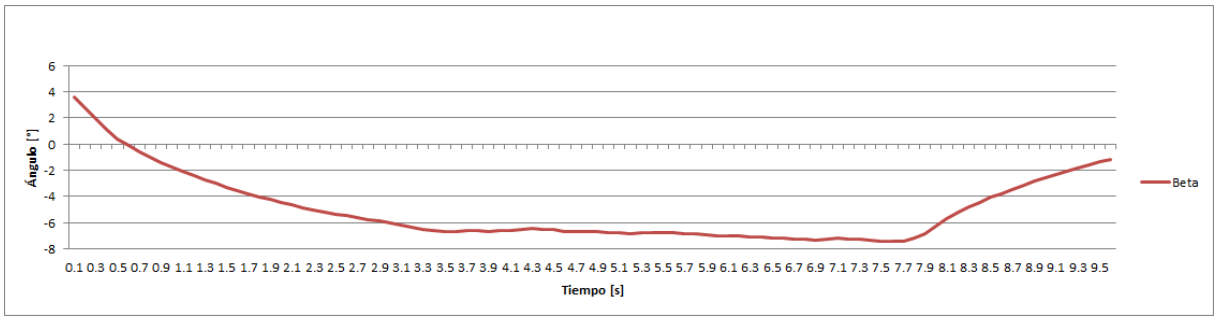


Figura 7.4: Prueba 2 de estabilidad en un eje

La Figura 7.4 muestra los valores registrados para el ángulo β y la serie de constantes $K_p = 205.66$, $K_i = 1123.2$ y $K_d = 14.37$, obtenidas para un $ts = 0.5\ s$ y $sp = 1\,\%$. En esta Figura se observa solamente un instante en el que el ángulo β toma el valor de cero, posteriormente se decrementa hasta alcanzar un valor sostenido de aproximadamente -7 grados durante 4 s antes de incrementarse en los últimos 2 s de la prueba. La causa por la que β no decrementa más allá de los -8 grados, es un tope mecánico ubicado en la parte superior de la base de pruebas, para evitar que se alcancen valores de β que ocasionen que las hélices se dañen al tener contacto con la parte inferior de dicha base. El incremento

del ángulo β en los últimos dos segundos de la prueba, se debió a una baja carga en las baterías que ocasionó que se perdiera control sobre la velocidad de giro de los motores. Al haberse mantenido un valor casi constante pero diferente de cero en el ángulo β, se puede considerar que el algoritmo de control no estabilizaba dicho ángulo y se mantenía también una inlcinación constante, por lo tanto, este control de estabilidad tampoco fue adecuado.

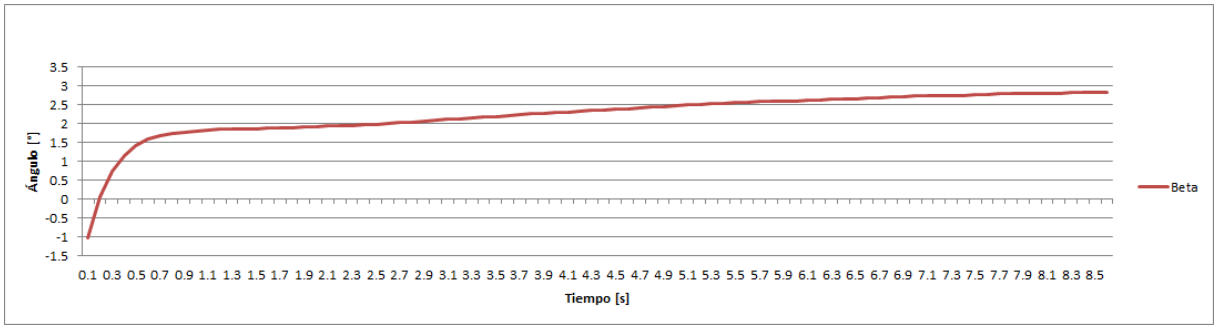


Figura 7.5: Prueba 3 de estabilidad en un eje

El comportamiento de β para la serie de constantes $K_p = 575.91$, $K_i = 5835.5$ y $K_d = 23.95$, obtenidas para un $ts = 0.3\ s$ y $sp = 2\,\%$, se muestra en la Figura 7.5, en la que el ángulo β varia casi 6 grados por segundo durante los primeros 0.5 s, después de los cuales, se incrementa lentamente hasta alcanzar el valor de 2.8 grados. Este último valor, es aún cercano a cero y β se mantiene inferior a 2.8 grados durante casi 8 s, sin variaciones repentinas, por lo que se puede considerar que el control de estabilidad elegido es adecuado para el sistema, aunque hay que tomar en cuenta que el valor de β aparenta seguirse incrementando al paso del tiempo por lo que la consideración anterior, es válida para los nueve segundos de la prueba.

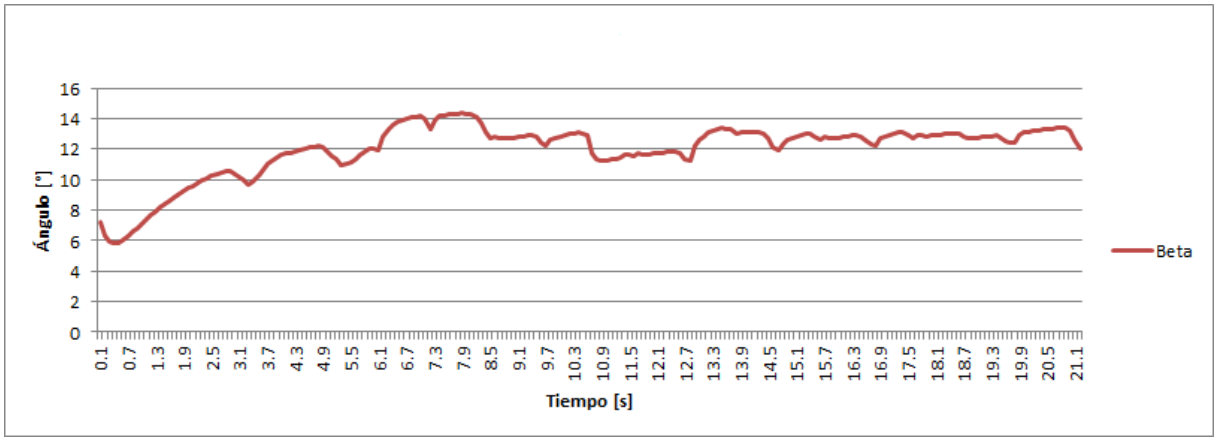


Figura 7.6: Prueba 4 de estabilidad en un eje

El comportamiento de β para la serie de constantes $K_p = 2433.7$, $K_i = 81234$ y $K_d = 47.9042$, obtenidas para un $ts = 0.15\ s$ y $sp = 10\,\%$, se muestra en la Figura 7.6 en la

que se observa una convergencia entre los 12 grados y 14 grados, aunque con variaciones constantes dentro de dicho rango. El tope mecánico para el rango positivo de β, limita su valor cerca de los 14 grados, es por esto que el valor de β en esta prueba no sobrepasa este límita; así pues las variaciones representan ligeros golpes del brazo de aluminio contra este tope, además de mantener una inclinación constante. Por lo anterior se puede considerar que el control de estabilidad diseñado, no es el adecuado para el sistema.

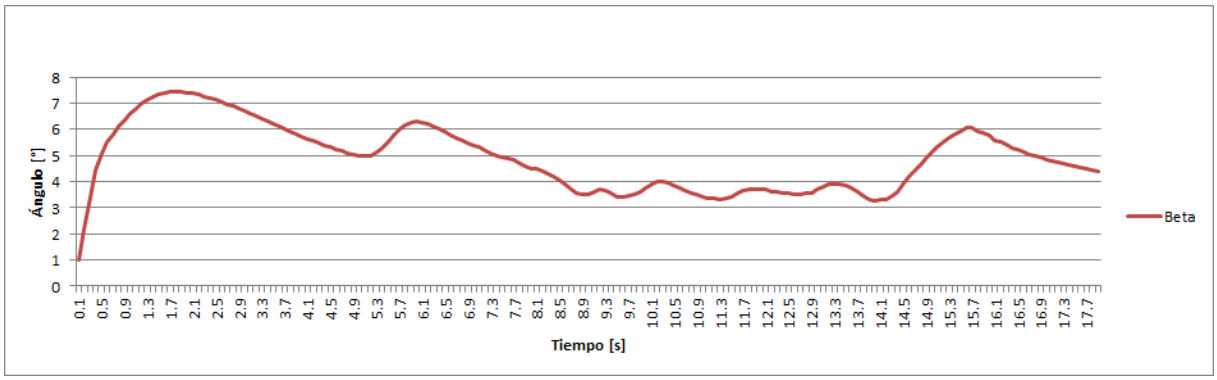


Figura 7.7: Prueba 5 de estabilidad en un eje

Finalmente, el los valores resgistrados de β para la serie de constantes $K_p = 21.8$, $K_i = 70.42$ y $K_d = 3.24$, obtenidas para un $ts = 1\ s$ y $sp = 5\,\%$, que son los valores utilizados para la simulación del control PID mostrada anteriormente, se muestra en la Figura 7.7 en la que se observa que el valor de β oscila entre 3 grados y 8 grados, con un periodo de oscilación corta entre 3 grados y 4 grados durante 6 segundos. Aunque algunas variaciones son lentas (máxima de 7 grados por segundo y mínimas de 1 grado por segundo) y el valor de β para la oscilación corta es de 3.5 grados, no se observa una convergencia estable ni acercamientos a cero durante la prueba, por lo que el control diseñado puede considerarse no adecuado para el sistema.

Tomando en cuenta que la configuración en un solo eje, no corresponde completamente con el modelo matemático utilizado para el diseño del controlador debido a la inclusión de constantes inerciales, al no contemplar vínculos mecánicos entre el eje de giro y el brazo, el peso, entre otros, las pruebas anteriores, pueden variar su resultado al probar el sistema completo, es decir, al intentar estabilizar los dos ejes horizontales y sin un vínculo mecánico de apoyo como el requerido para las pruebas en un solo eje.

7.3. Pruebas con el sistema completo

Procedimiento para realizar las pruebas con el sistema completo

Como medida de seguridad, para probar el cuadricóptero, se construyó una plataforma de pruebas de la que el cuadricóptero fue amarrado, permitiéndole un espacio de trabajo de no más de 0.2 m sobre el nivel de la plataforma. Lo anterior fue para proteger las hélices

de tener contacto con algún elemento en la plataforma o fuera de ella, para restringir los movimientos del cuadricóptero y para proteger a los usuarios de algún contacto con éste, ya que sus rotores, a las velocidades angulares en que trabajan, son peligrosos, sobre todo en etapa de prueba y verificiación del controlador. La plataforma se muestra en la Figura 7.8 con el cuadricóptero montado.

Figura 7.8: Plataforma de pruebas

De manera similar al procedimiento realizado para las pruebas en un solo eje, se eligieron diferentes constantes para el control PID y se observó el comportamiento del cuadricóptero registrando el valor de los ángulos de cada eje horizontal al paso del tiempo. Para el desarrollo de estas pruebas, se agregó el programa completo descrito en el capítulo anterior que incluye los algoritmos de arranque y de paro remoto.

Resultados de las pruebas del sistema completo.

La Figura 7.9 muestra los valores registrados de α y β para la serie de constantes $K_p = 1323.3$, $K_i = 25147$ y $K_d = 35.93$, con el desempeño deseado: $ts = 0.2\,s$ y $sp = 5\,\%$. Esta prueba duró siete segundos y se muestran valores para β de entre 0 grados y 35 grados con variaciones de entre 5 y 6 grados por segundo además de que es evidente que no existe convergencia a cero.

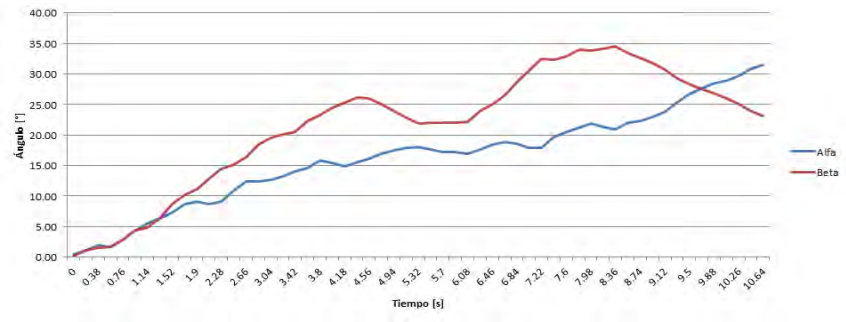


Figura 7.9: Prueba 1 de estabilidad con el sistema completo

La Figura 7.10 muestra los valores de PWM en microsegundos para cada motor. Se puede observar que dichos valores se mantienen constantes en 1351 μs y 1330 μs, lo que podría explicar la falta de control sobre los ángulos del cuadricóptero.

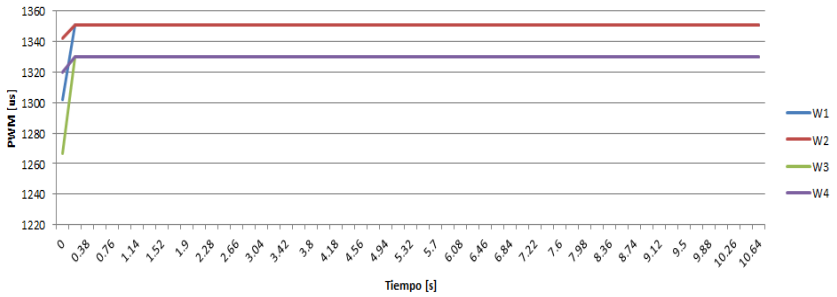


Figura 7.10: Evolución 1 de las velocidades angulares en PWM

La Figura 7.11 muestra los valores del ángulo β para la serie de constantes $K_p = 205.66$, $K_i = 1123.2$ y $K_d = 14.37$, con el desempeño deseado: $ts = 0.5\ s$ y $sp = 1\,\%$. En esta figura se observa que los valores de α se mantienen cercanos a cero durante los primeros cuatro segundos de la prueba, antes de que incrementen su valor. Para β, los valores se incrementan a razon de 10 grados por segundo durante los primeros dos segundos de la prueba para después decrementar lentamente hasta el final de la misma. Tampoco se observa convergencia a cero por un tiempo razonable que haga suponer un control adecuado de la estabilidad.

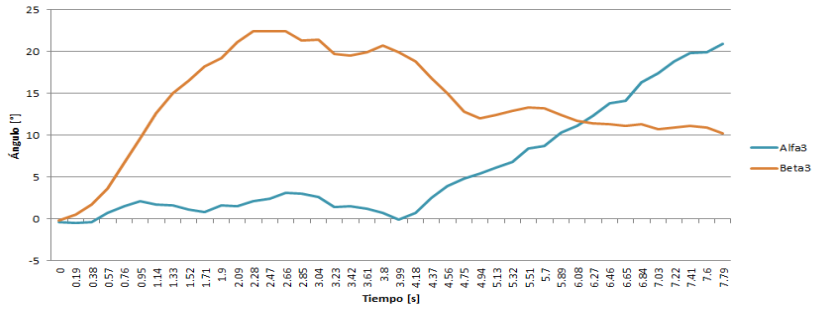


Figura 7.11: Prueba 2 de estabilidad con el sistema completo

En la Figura 7.12 se muestran los valores de PWM de cada motor, en los que a diferencia de la Prueba 1, se pueden identificar variaciones importantes a los cuatro segundos de iniciada la prueba, aunque tampoco representan un control adecuado de las velocidades angulares de los motores para estabilizar los ángulos.

El comportamiento de α y β para la serie de constantes $K_p = 575.91$, $K_i = 5835.5$ y $K_d = 23.95$, obtenidas para un $ts = 0.3\ s$ y $sp = 2\,\%$, se muestra en la Figura 7.13, en la que ambos ángulos se mantienen relativamente cerca de cero durante los primeros siete segundos de la prueba, antes de inrementar su valor.

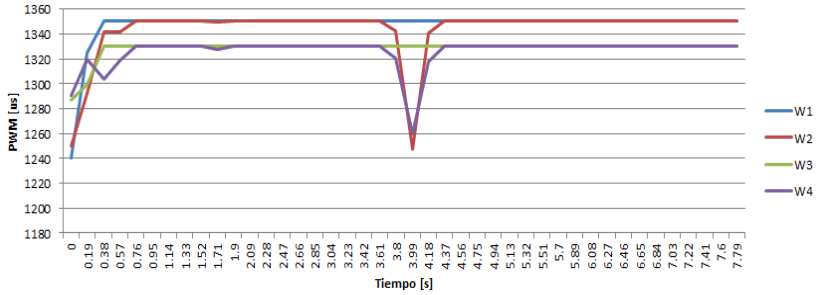


Figura 7.12: Evolución 2 de las velocidades angulares en PWM

Tomando en cuenta que entre el segundo 2.5 y 7, los ángulos se mantienen entre -5 grados y 7 grados, se puede pensar que el control funciona adecuadamente para este intervalo e incluso antes, cuando al ser un ángulo con valores mayores que cero, posteriormente disminuyen su valor y viceversa.

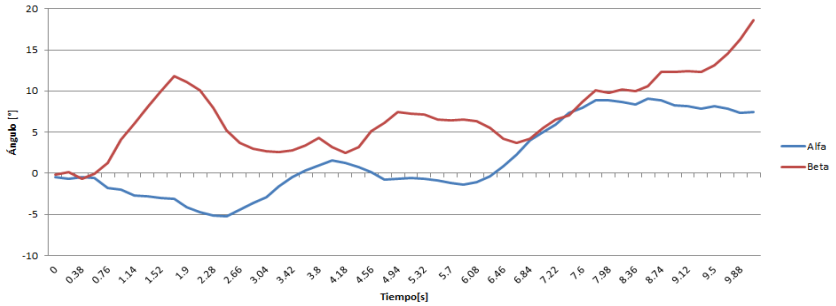


Figura 7.13: Prueba 3 de estabilidad con el sistema completo

La evolución de las velocidades angulares en PWM de los cuatro motores para la prueba 3 se muestra en la Figura 7.14 en la que se pueden observar aun más, variaciones importantes para cada velocidad, pero sobre todo, cuando en los ángulos existe un valor cercano a cero.

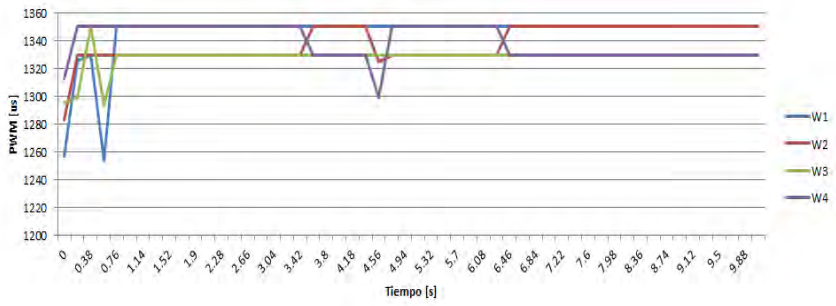


Figura 7.14: Evolución 3 de las velocidades angulares en PWM

El comportamiento de α y β para la serie de constantes $K_p = 2433.7$, $K_i = 81234$ y $K_d = 47.9042$, obtenidas para un $ts = 0.15\,s$ y $sp = 10\,\%$, se muestra en la Figura 7.15. La prueba 4 con el sistema completo tuvo una duración de casi 15 segundos, lo que permite

observar durante más tiempo los valores que toman los ángulos medidos. En esta prueba, tampoco se observa una convergencia adecuada a cero, sin embargo, en los últimos cuatro segundos de la prueba, los ángulos se mantienen con valores entre -7 grados y 5 grados.

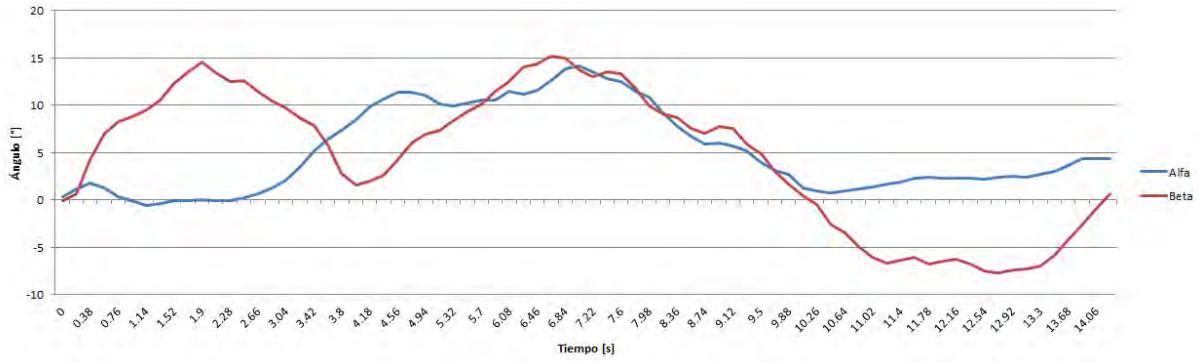


Figura 7.15: Prueba 4 de estabilidad con el sistema completo

En la Figura 7.16 se muestra los valores calculados para la señal de PWM de cada motor y en donde nuevamente se observa que los dichos valores cambian cuando el ángulo correspondiente se acerca a cero.

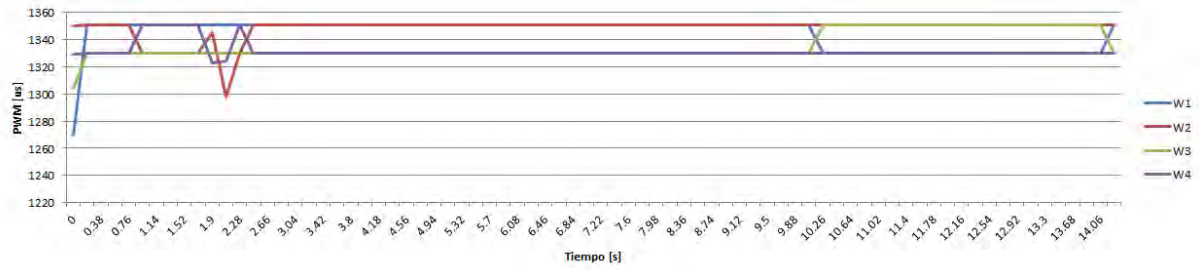


Figura 7.16: Evolución 4 de las velocidades angulares en PWM

Finalmente, el comportamiento de α y β para la serie de constantes $K_p = 21.8$, $K_i = 70.42$ y $K_d = 3.24$, obtenidas para un $ts = 1\ s$ y $sp = 5\,\%$, que son los valores utilizados para la simulación del control PID mostrada en el Capítulo 5, se muestra en la Figura7.17 en la que se observan grandes variaciones en β y un incremento lento para α durante los primeros 10 segundos de la prueba para posteriormente disminuir hasta acercarse a cero por unos instantes.

Y el comportamiento de las señales de PWM para cada motor durante la prueba 5 se muestran en la Figura 7.18 en donde también se observa que conforme un ángulo se acerca a cero, las señales correspondientes de PWM varían más, y cuando el valor del ángulo está más alejado de cero, el valor de las señales de PWM correspondientes se satura en $1351\mu s$ y $1330\mu s$.

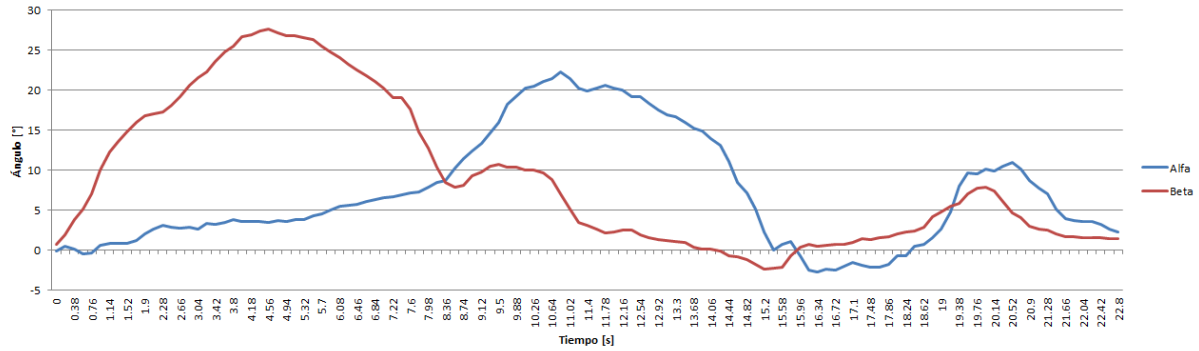


Figura 7.17: Prueba 5 de estabilidad en un eje

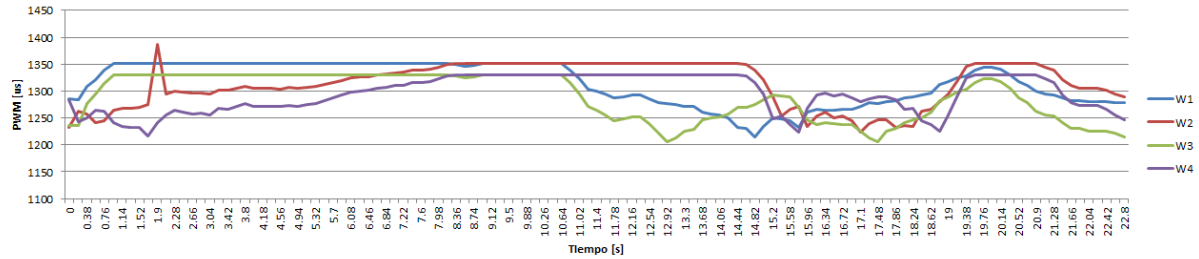


Figura 7.18: Evolución 5 de las velocidades angulares en PWM

Con lo anterior, analizando sobre todo las gráficas de las señales de PWM y tomando en cuenta las constantes del control PID de cada prueba, se puede observar, primero, que las señales PWM varían más cuando el valor del ángulo correspondiente se acerca a cero y segundo, que esta variación es más notable, o permite un error en el ángulo con respecto a cero más grande conforme las constantes PID son más pequeñas.

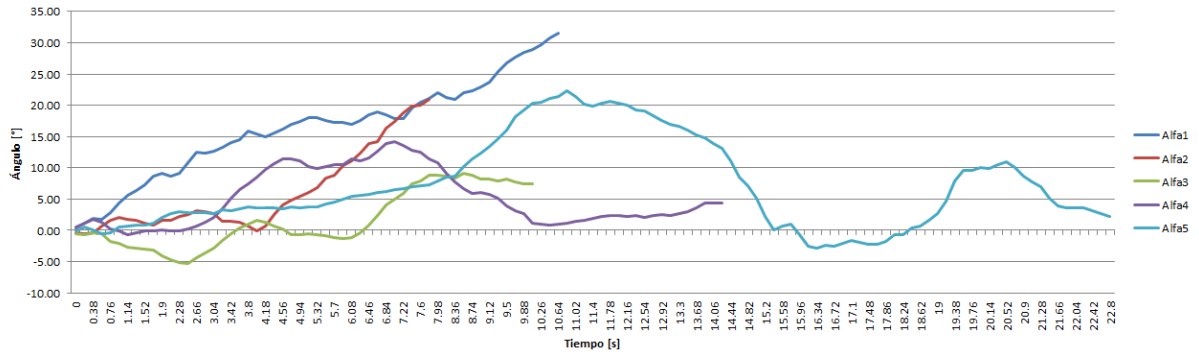


Figura 7.19: Comparativa de α

Por lo tanto, se puede pensar que en algún proceso interno del microcontrolador, alguna de las variables involucradas en el calculo de dichas señales PWM alcanza un valor límite según el tipo de variable, lo que provoca que el cálculo se detenga y la señal PWM se mantenga constante en un límite superior, lo cual afecta directamente al desempeño del controlador. Sin embargo, a continuación se presenta una gráfica en la Figura 7.19 en la que se registran los valores de α para cada una de las pruebas, para poder comparar el

comportamiento del cuadricóptero a fin de analizar cuál configuración de constantes PID, obtiene un control más adecuado de la estabilidad del prototipo en las pruebas realizadas. De manera similar, la Figura 7.20 presenta una comparativa del ángulo β para cada una de las pruebas anteriores.

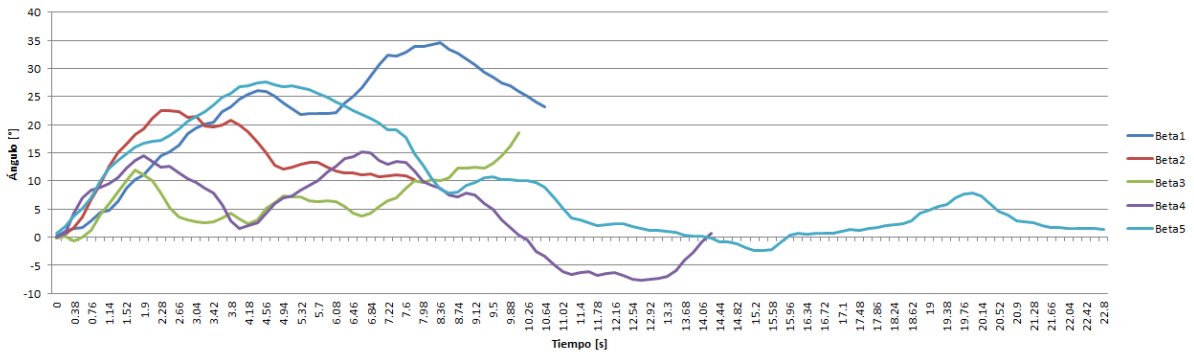


Figura 7.20: Comparativa de β

De las gráficas comparativas de α y β, se puede decir que las pruebas que más se acercaron al control adecuado de la estabilidad son, la prueba 3 y la prueba 5, sin embargo, esta selección se hace con base en la comparativa y en el rango de valores para cada ángulo durante el periodo de tiempo en el que tenían valores cercanos a cero. Sin embargo, lo anterior no significa que estas dos pruebas representan realmente un control adecuado de la estabilidad, sino que solo se acercan más que las demás.

7.4. Pruebas a futuro

Con base en la experiencia adquirida y los resultados obtenidos en las pruebas anteriores con el sistema completo, se proponen diversos cambios tanto en la programación del algoritmo de control, como en el algoritmo de obtención de datos de la IMU, así como en la declaración del tipo de variables, entre otros. Como trabajos a futuro, se planea además, agregar diferentes funciones y soluciones al cuadricóptero como:

- El uso de un control remoto o una interfaz de computadora más completa y versátil para el control del cuadricóptero que pueda mostrar de manera más simple y gráfica los valores de las variables del sistema como los ángulos de navegación, la orientación, la existencia de comunicación inalámbrica, datos adicionales como velocidad del viento, presión atmosférica, entre muchos otros. Además que se cuente con elementos que permitan dirigir el prototipo de manera más ergonómica como un *joystcik* o botones, incluso para el encendido y apagado de la unidad o el depegue y aterrizaje.

- Nuevas pruebas con diferentes constantes PID con valores de diseño diferentes como en el caso del tiempo de asentamiento dependiendo del comportamiento del sistema completo y buscando una convergencia hacia el control adecuado de la estabilidad.

Además de una programación adecuada que permita sensar en menor tiempo las variables del cuadricóptero, incluídas las velocidades y aceleraciones del mismo; y que el microcontrolador puede manejar los valores de dichas variables sin generar errores por saturación de registros por tipo de variables entre otros.

- Diseñar un armazón de seguridad para protección en vuelos en interiores, así como un rediseño de la estructura del cuadricóptero y posiblemente una nueva selección de componentes electronicos para hacerlo más ligero y por consiguiente mejorar el desempeño de los motores y prolongar el tiempo de vuelo relacionado con la energía requerida a las baterías.

- Ampliar el panorama de control, incluyendo nuevos algoritmos como el control difuso, el control robusto, entre otras de manera que su pueda obtener un control más adecuado de la estabilidad y también de regulación y de seguimiento para ampliar a su vez, el panorama de aplicaciones y usos del cuadricóptero.

- Después de haber experimentado con diversos algoritmos de control y de haber obtenido un resultado satisfactorio con respecto a un control adecuado, realizar diversas pruebas con el sistema completo sin la necesidad de ligarlo a una plataforma.

- La búsqueda, el uso y la implementación del algoritmo de control en otros microcontroladores con más capacidades de procesamiento, menor tiempo de ejecución, mayor memoria, entre otros factores.

Figura 7.21: El cuadricóptero en acción

Conclusiones

El objetivo planteado de este trabajo trata sobre buscar el control de la estabilidad de un cuadricóptero con una referencia constante igual a cero, a lo que se dió cumplimiento por las diversas simulaciones y pruebas realizadas. Además se planteó la necesidad de diseñar y construir un prototipo que permita definir los parámetros y constantes requeridas para el desarrollo del controlador, a lo que también se dió cumplimiento con el diseño y construcción del prototipo utilizado para realizar las diferentes pruebas.

Con respecto a las características deseadas del cuadricóptero, se construyó un prototipo que cumple dichas características como la de volar, envio y recepción de información, despegue y aterrizaje autónomo, suministro de energía por baterías recargables, entre otras. Con respecto al control de perturbaciones, como se mencionó anteriormente, es cuestión de realizar más experimentos para afinar el desempeño del controlador, sin embargo, con base en el diseño del control PID, del control por realimentación de estados, sus respectivas simulaciones y la implementación y pruebas realizadas, el cuadricóptero también cumple con estas características deseadas.

El modelo matemático de un cuadricóptero, es un sistema complicado ya que como se mencionó en el Capítulo 5, el modelo presentado en este trabajo es una de las representaciones más sencillas de un cuadricóptero al despreciar efectos y no linealidades inherentes en el sistema, así como el asignar valores casi aleatorios en los parámetros desconocidos del sistema y para los cuales se necesitaría pruebas complejas para determinar su valor exacto.

El control PID funciona bien para el objetivo de estabilidad, tal como lo muestra la simulación correspondiente, sin embargo, con base en las pruebas realizadas, es necesario rediseñar el algoritmo de control programado para conocer con certeza si el control PID funciona adecuadamente en la experimentación.

El control PID, así como el control por realimentación de estados, según las simulaciones correspondientes, son suficientes para lograr la estabilidad del cuadricóptero, e incluso para realizar vuelos y tareas más complejas como ir de un punto a otro con ayuda, por ejemplo, de un sistema GPS, sin embargo, para maniobras a grandes velocidades, "acrobacias" o control de perturbaciones severas como ráfagas de aire que lo hagan operar

muy alejado de su punto de equilibrio, es indispensable contar con una técnica de control más robusta, de respuesta rápida y que incluso no dependa del modelo matemático y sus consideraciones y simplificaciones o que por lo menos, pueda adaptar los parámetros desconocidos o despreciados, como lo haría un control robusto-adaptable [38] o un control difuso.

Así como en la técnica de control, se requiere mayor versatilidad y velocidad en las operaciones, así también se requiere un tiempo de sensado mucho más veloz y/o una capacidad de procesamiento más amplia como la del microcontrolador ARM Cortex M3 utilizado en la tarjeta de desarrollo Maple de LeafLabs que trabaja a 90 DMIPS, la del microcontrolador LPC2148 integrado en la *Ultimate IMU* [39] para integrar las señales de los sensores con 60 MIPS o incluso el uso de un DSP (*Digital Signal Processor*) en caso de ser requerido.

Se requiere tener cuidado con el ensamble de los motores y las hélices, el balanceo y la presión de apriete de tuercas ya que alguna deficiencia en las hélices como el desgaste o el no fijarlas adecuadamente al eje del motor, puede ocasionar ruido mecánico, daños en el eje del motor, sobrecalentamiento del motor e incluso que, por las velocidades de operación, las hélices salgan disparadas hacia algún espectador y lo lastimen y/o que el cuadricóptero caiga al piso, dañando los demás componentes. Además es necesario, como protocolo de mantenimiento previo a cada puesta en marcha y operación del cuadricóptero, verificar que las tuercas, los tornillos, los cinturones sujetadores, los conectores, los cables, la comunicación inalámbrica, los componentes electronicos de procesamiento y sensado, las baterías, las conexiones, la polaridad y el sentido de giro de los motores se encuentren en buen estado y en su correcta posición para evitar algún percance.

El ruido mecánico es un factor complicado de manejar pero que es inherente a un sistema que tiene cuatro rotores funcionando a velocidades superiores a las 2000 rpm, por lo que es indispensable contar con un filtro mecánico y digital para obtener una mejor lectura de los sensores y evitar respuestas no deseadas del controlador. Adicionalmente, no se registraron problemas con la comunicación inalámbrica debido a las vibraciones mecánicas.

Las baterías, el peso y el tiempo de vuelo, son factores determinantes del rendimiento del cuadricóptero y la viabilidad de su uso. Estos factores son parte de una función de "costo-beneficio" que es importante analizar para brindar el tiempo más prolongado de vuelo con el menor peso posible. Así mismo, se puede buscar que la estructura sea más ligera sin afectar la robustez, probando con materiales novedosos como la fibra de carbono.

En esto último es importante recalcar que debido a la disponibilidad actual de baterías ligeras y de gran capacidad, es recomendable utilizar cuadricóptero ligeros, posiblemente menores a 2 kg ya que se cuentan con componentes electrónicos de gran calidad y funcionalidad como microcontroladores muy compactos, mini o micro cámaras, trans-

misión de video de alta calidad por medio de circuitos muy pequeños y ligeros y GPS cada vez más potentes con mejor resolución y menor tamaño, entre otros. Así que puede ser mejor aprovechar la tecnología de microelectrónica para este tipo de vehículos, que diseñar vehículos grandes para poder llevar abordo componentes electronicos grandes.

Para las pruebas con el sistema completo, es muy importante tener correcta la configuración de los motores y el sentido de giro de los ángulos en el modelo real, de modo que el programa corresponda tanto al modelo físico como al modelado matemático.

Los parámetros despreciados desde el modelado, son muy importantes cuando se desea que el ambiente de operación del cuadricóptero sea en exteriores, o en su defecto, utilizar un control que no dependa de estos parámetros.

Como parte de las modificaciones posibles a los componentes electrónicos elegidos, debido a la gran capacidad de empuje de los motores, se puede contemplar incrementar la capacidad de las baterías para aumentar el tiempo de vuelo, tomando en cuenta el "costo-beneficio" mencionado.

El tamaño de las hélices también se puede incrementar de 12 x 3.8 a 14 x 4.7 para aprovechar de una mejor manera el par mecánico de los motores *brushless outrunner* seleccionados.

Los alcances y posibilidades de un cuadricóptero son muy amplias, como parte de trabajos posteriores, se puede agregar al sistema, localización GPS, sensores de distancia, sonares, sistemas de visión, sistemas de seguridad, instrumentos de medición de uso específico, entre muchos otros. Lo anterior, encaminado a darle al cuadricóptero una aplicación específica como puede ser el reconocimiento de terrenos, exploración, adquisición de datos atmosféricos, visión remota, usos topográficos, rescate, vigilancia, apoyo a instituciones de seguridad, militar, etc.

Referencias

[1] W Johnson. *Helicopter Theory.* Dover Publications, New York, 1994.

[2] *Real Academia Española: Definición de Helicóptero.*
URL: http://www.rae.es, Agosto 2011.

[3] *Draganfly: Aplicaciones del Draganfly X8.*
URL: http://www.draganfly.com/uav-helicopter/draganflyer-X8/applications/, Marzo 2011.

[4] *Sparkfun Electronics: Equipo de cuadricópteros para construcción.*
URL: http://www.sparkfun.com/news/532, Agosto 2011.

[5] H. Wu, J. Peng and Q. Chen. (rbf-arx) model based modeling and control of quadrotor. *IEEE International Conference on Control Applications*, 2010. Yokohama, Japan.

[6] H. Waslander L. Hoffmann, M. Huang and J. Tomlin. Quadrotor helicopter flight dynamics and control: Theory and experiment. *American Institute of Aeronautics and Astronautics.*

[7] *Draganfly: Página principal.*
URL: http://www.draganfly.com, Marzo 2011.

[8] A. Dickmen, C. Arisoy and H. Temeltas. Attitude control of a quadrotor. *IEEE*, 2009.

[9] T. Madani and A. Benallegue. Backstepping control for a quadrotor helicopter. *IEEE International Conference on Intelligent Robots and Systems*, 2006. Beijing, China.

[10] G. Raffo, V. Ortega and R. Rubio. An integral predictive/nonlinear h∞ control structure for a quadrotor helicopter. *Departamento de Ingeniería de Sistemas y Automática*, 2009. Universidad de Sevilla, España.

[11] V. Burg T. Dawson P. Lee, P. Chitrakaran and B. Xitan. Control of a remotely operated quadrotor aerial vehicle and camera unit using a fly-the-camera perspective.

[12] M. Mohamed H. Salih, A. Moghavvemi and K. Sallom. Modelling and pid controller design for a quadrotor unmanned air vehicle. *Centre for Research in Applied Electronics*. Malay.

[13] T. Dierks and S. Jagannathan. Output feedback control of a quadrotor uav using neural networks. *IEEE trans. on Neural Networks*, Vol. 21(1), January 2010.

[14] A. Astrov, I. Pedai and E. Rusten. Desired trajectory generation of a quadrotor helicopter using hybrid control for enhanced situational awareness. *IEEE International Conference on Information and Automation*, June 2010. Harbin, China.

[15] R. Pounds, P. Mahony and P. Corke. Modelling and control of a quadrotor robot. *Proceedings of the Australian Conference on Robotics and Automation*, 2006.

[16] *Wikipedia: Artículo sobre motores tipo brushless.*
URL: http://en.wikipedia.org/wiki/Brushless_motor, Julio 2011.

[17] *Sparkfun Electronics: Catálogo de productos sensores.*
URL: http://www.sparkfun.com/categories/23, Agosto 2011.

[18] Aldo Retana, David y Vargas. Estabilización de un helicóptero a escala media mediante sistemas neurodifusos, 2010.

[19] *Sparkfun Electronics: Producto 9 DOF Sensor Stick.*
URL: http://www.sparkfun.com/products/10724, Septiembre 2011.

[20] *Sparkfun Electronics: Hoja de datos del acelerómetro ADXL345.*
URL: http://www.sparkfun.com/datasheets/Sensors/Accelerometer/ADXL345.pdf, Octubre 2011.

[21] *Sparkfun Electronics: Hoja de datos del girómetro ITG3200.*
URL: http://www.sparkfun.com/datasheets/Sensors/Gyro/PS-ITG-3200-00-01.4.pdf, Octubre 2011.

[22] *Sparkfun Electronics: Hoja de datos del magnetómetro HMC5883L.*
URL: http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Sensors/Magneto/HMC5883L-FDS.pdf, Octubre 2011.

[23] *Wiring: Página principal y descripción general.*
URL: http://wiring.org.co/, Noviembre 2011.

[24] *Processing: Página principal y descripción general.*
URL: http://www.processing.org/, Noviembre 2011.

[25] *Arduino: Página principal.*
URL: http://www.arduino.cc, Agosto 2011.

[26] *Digi: Página principal y catálogo.*
URL: http://www.digi.com, Julio 2011.

[27] *Sparkfun Electronics: Producto XBee Pro 900 XSC RPSMA.*
URL: http://www.sparkfun.com/products/9087, Agosto 2011.

[28] *Siemens: Página principal de Solid Edge.*
URL: http://www.plm.automation.siemens.com/en_sg/products/velocity/solidedge/, Agosto 2011.

[29] *Draganfly: Características del Draganflyer X8.*
URL: http://www.draganfly.com/uav-helicopter/draganflyer-X8/, Abril 2011.

[30] *Dassault Systemes: Página principal de Catia.*
URL: http://www.3ds.com/products/catia, Agosto 2011.

[31] *Metales Díaz: Página principal y catálogo.*
URL: http://www.metalesdiaz.com.mx, Julio 2011.

[32] Y. Engr and V. Abbass. Modeling of quadrotor helicopter dynamics. *International Conference on Smart Manufacturing Application*, April 2008. Gyeonggi-do, Korea.

[33] Katsuhiko Ogata. *Ingeniería de Control Moderna.* Pearson Educación, Madrid, 2003.

[34] *MathWorks Matlab: Página principal.*
URL: http://www.mathworks.com, Julio 2011.

[35] H. Voos. Nonlinear state-dependent riccati equation control of a quadrotor uav. *IEEE International Conference on Control Applications*, October 2006. Munich.

[36] Fabio Varesano. *Using Arduino for Tangible Human Computer Interaction.* PhD thesis, Universita degh Studi di Torino, 2011.

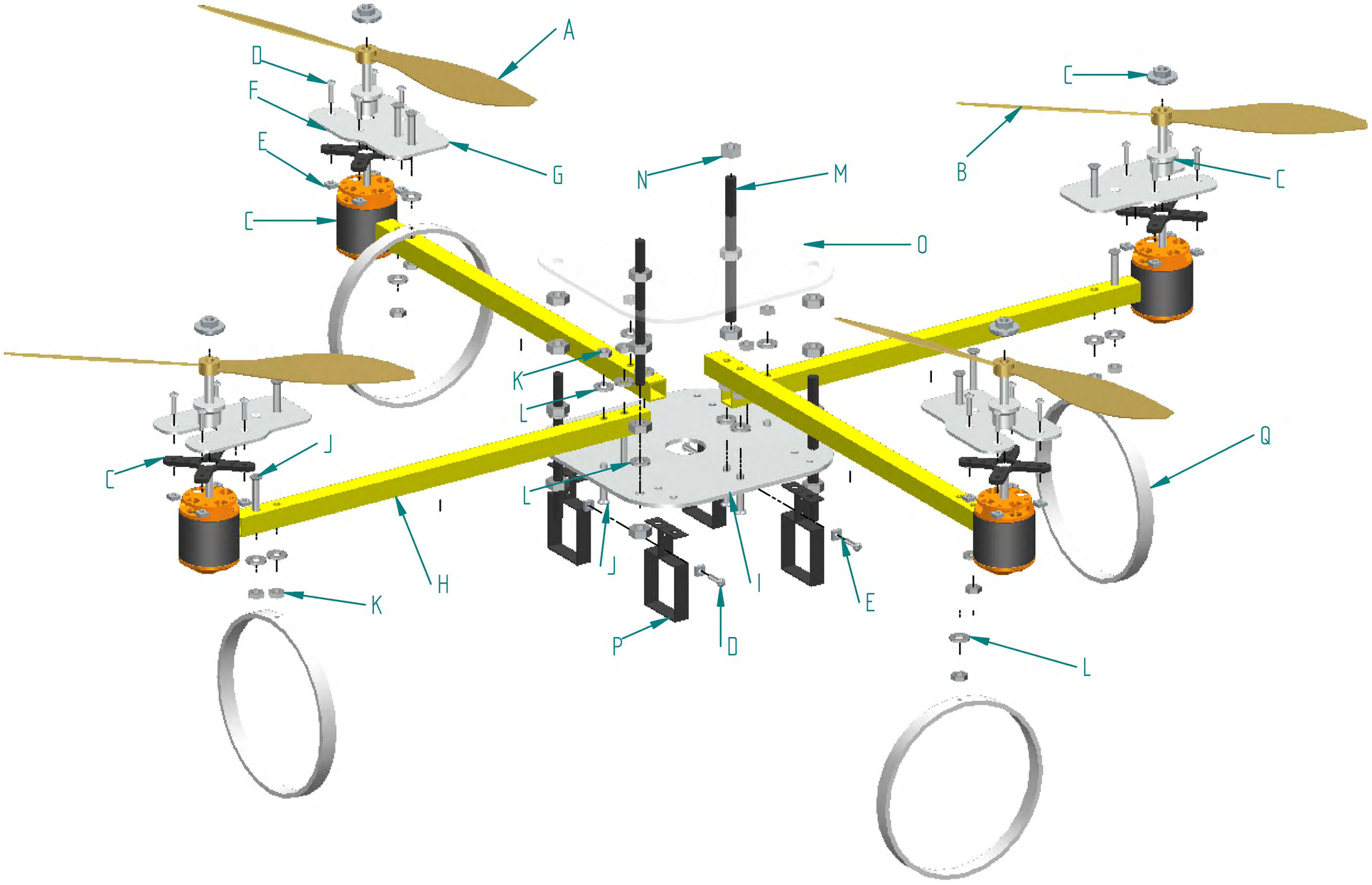
[37] *Fabio Varesano: Página principal y ligas de descarga de bibliotecas.*
URL: http://www.varesano.net/, Agosto 2011.

[38] P. N. Paraskevopoulos. *Modern Control Engineering.* Dekker Publications, United States, 2002.

[39] *Sparkfun Electronics: Producto Ultimate IMU.*
URL: http://www.sparkfun.com/products/10082, Agosto 2011.

Apéndices

A. Planos, materiales y diagrama de conexiones



	NOMBRE	FECHA
DIBUJÓ	LUIS A. J.	26/03/11
REVISÓ	SERAFIN C.	29/03/11

SI NO SE ESPECIFICA LO CONTRARIO, TODAS LAS DIMENSIONES SON EN MM

CUADRICÓPTERO

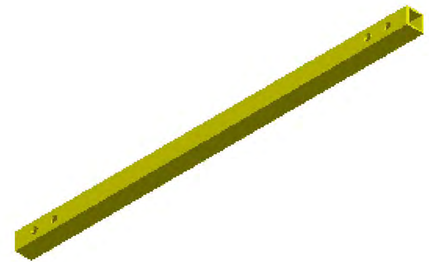
TÍTULO
ENSAMBLE GENERAL

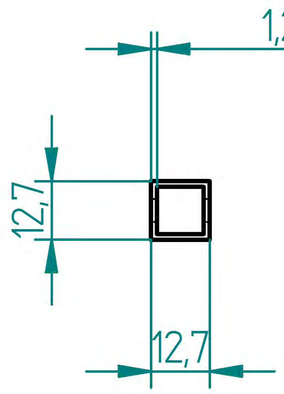
ESCALA: S/E

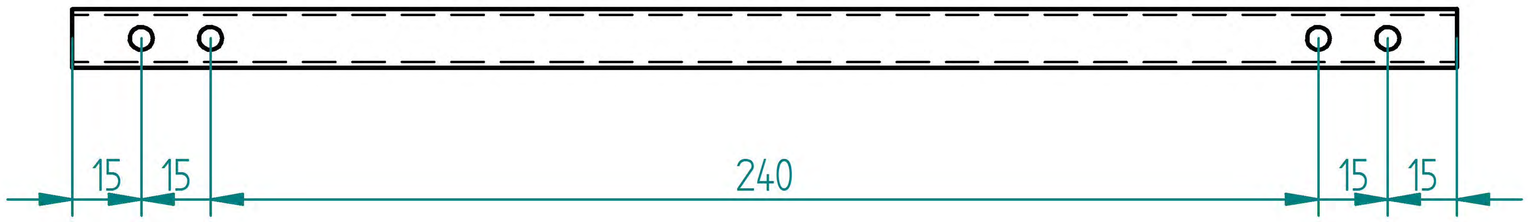
HOJA 6 DE 6

Cuadro A.1: Elementos del Ensamble General

Id	Elemento	Cant.	Plano o Descripción
A	Hélice Pusher	2	APC PusHer 12 pulg x 3.8 pulg
B	Hélice Flyer	2	APC Flyer 12 pulg x 3.8 pulg
C	Motor y Accesorios	4	KD-36 10-XL 900 kv
D	Tornillo cab. gota	16	1/8 pulg Ø x 1/2 pulg
E	Tuerca Cuadrada	16	1/8 pulg Ø
F	Rondana de Presión	16	1/8 pulg Ø
G	Lámina de Aluminio	4	CUAD-002
H	Perfil Cuadrado	4	CUAD-001
I	Lámina de Aluminio	1	CUAD-003
J	Tornillo cab. plana	16	3/16 pulg Ø x 1pulg
K	Tuerca Hexagonal	16	3/16 pulg Ø
L	Rondana de Presión	20	1/4 pulg Ø
M	Varilla Roscada	4	1/4 pulg Ø x 0.1 m
N	Tuerca Hexagonal	16	1/4 pulg Ø
O	Placa Acrílico	1	CUAD-004
P	Soporte Baterías	4	Ajustables de aluminio
Q	Aro de P.V.C.	4	CUAD-005
-	Ligas	16	Comerciales
-	Cinturones de Plástico	20	Ajustables, comerciales
-	Interruptor de corriente	1	Tipo cuchilla para alta corriente







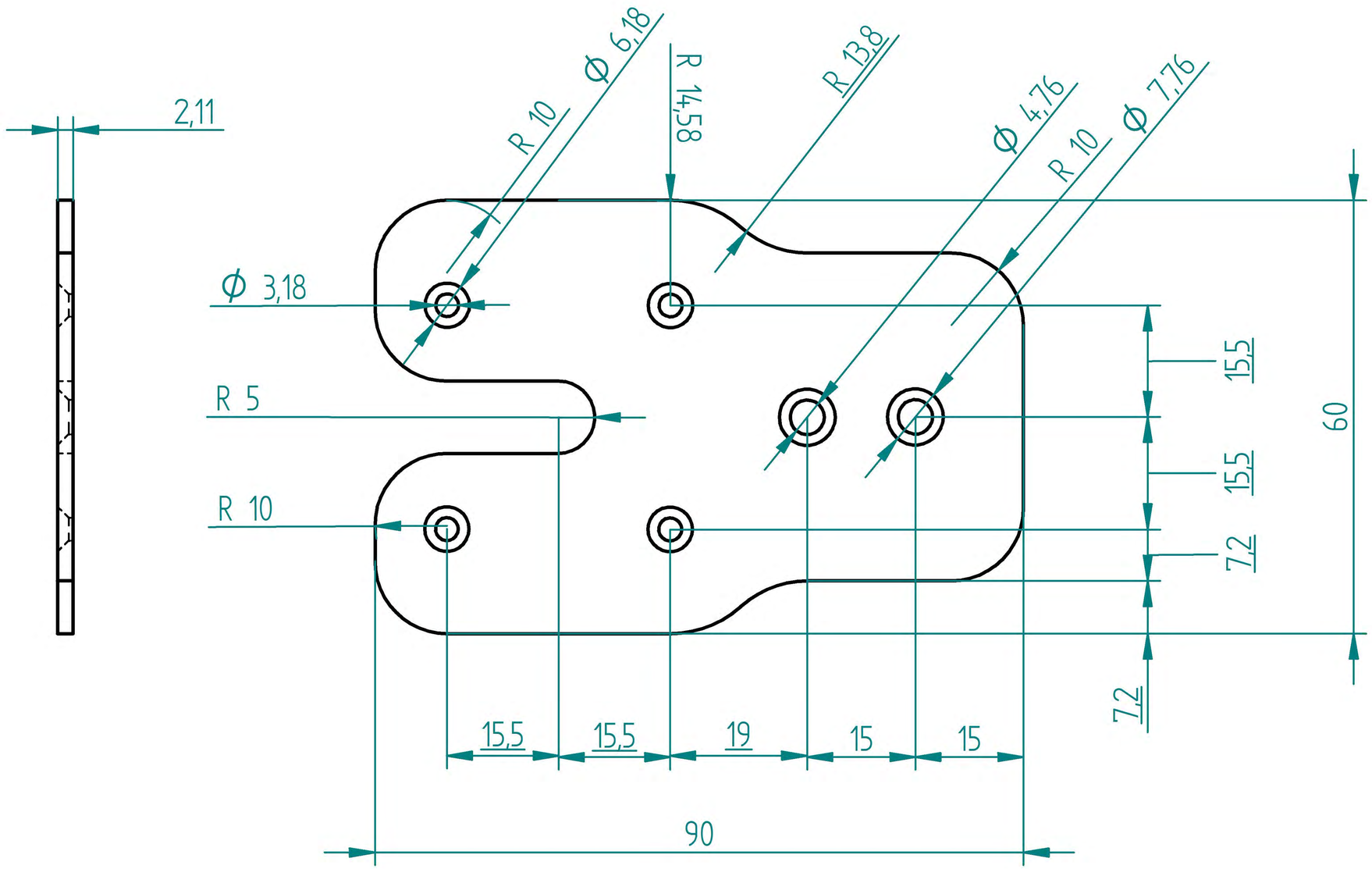
	NOMBRE	FECHA
DIBUJÓ	LUIS A. J.	26/03/11
REVISÓ	SERAFIN C.	29/03/11

ALUMINIO, PERFIL CUADRADO 12.7X12.7

CUADRICÓPTERO

TÍTULO
CUAD-001

ESCALA: S/E | Unidades: mm | HOJA 1 DE 6



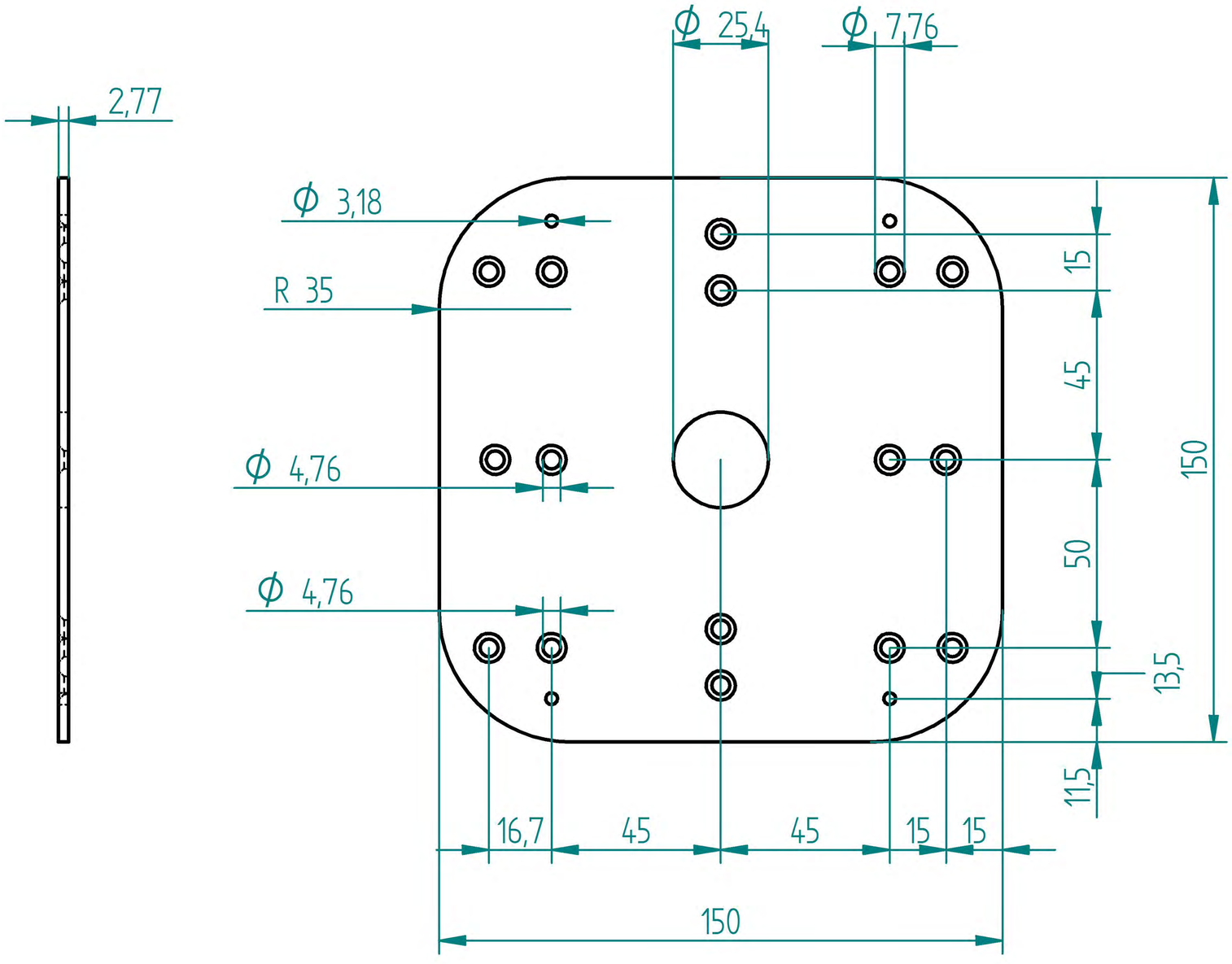
	NOMBRE	FECHA
DIBUJÓ	LUIS A. J.	26/03/11
REVISÓ	SERAFIN C.	29/03/11

ALUMINIO, LÁMINA CALIBRE 14

CUADRICÓPTERO

TÍTULO
CUAD-002

ESCALA: S/E | Unidades: mm | HOJA 2 DE 6



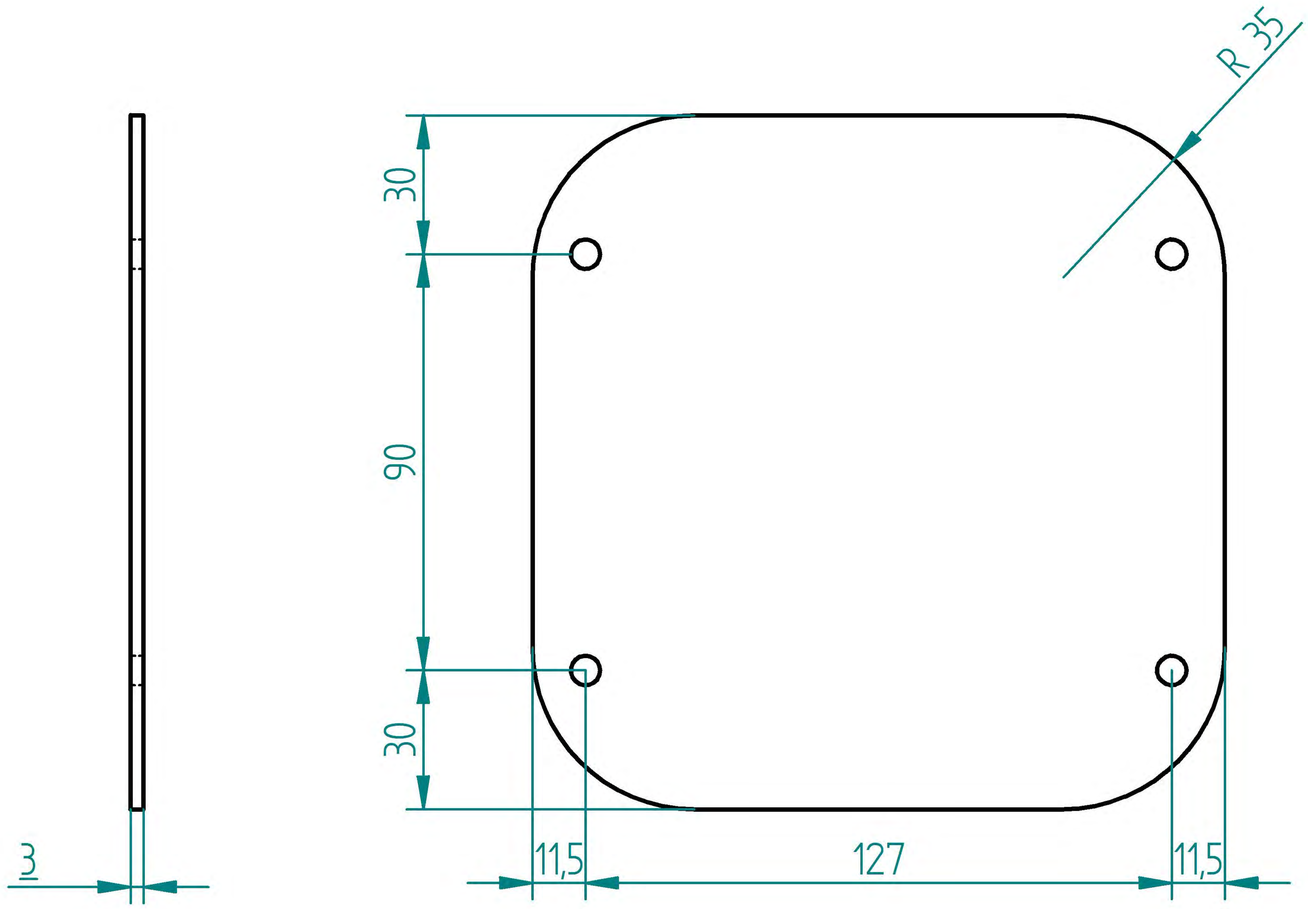
	NOMBRE	FECHA
DIBUJÓ	LUIS A. J.	26/03/11
REVISÓ	SERAFIN C.	29/03/11

ALUMINIO, LÁMINA CALIBRE 12

CUADRICÓPTERO

TÍTULO
CUAD-003

ESCALA: S/E | Unidades: mm | HOJA 3 DE 6



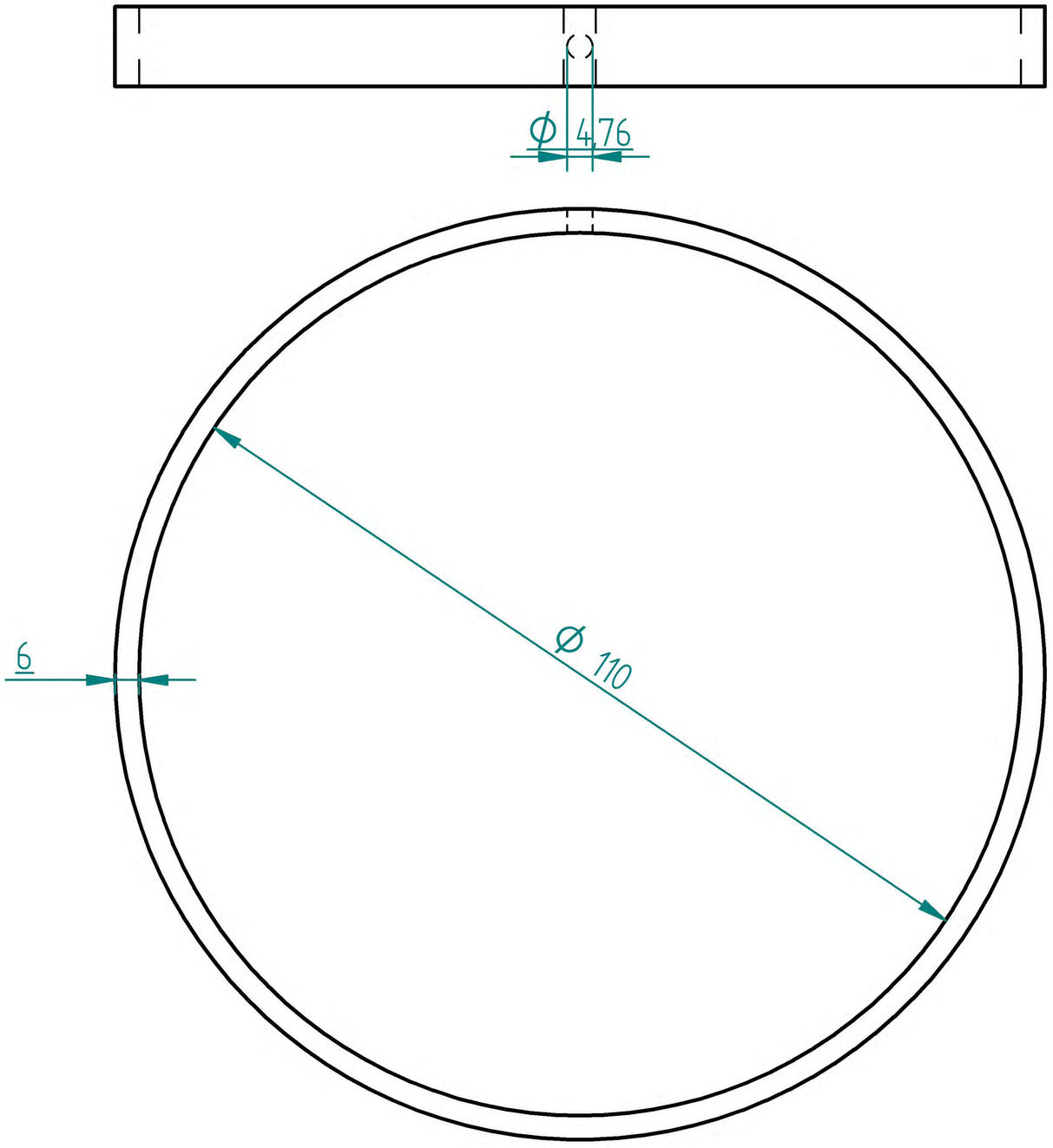
	NOMBRE	FECHA
DIBUJÓ	LUIS A. J.	26/03/11
REVISÓ	SERAFIN C.	29/03/11

ACRÍLICO, ESPESOR DE 3 MM

CUADRICÓPTERO

TÍTULO
CUAD-004

ESCALA: S/E | Unidades: mm | HOJA 4 DE 6



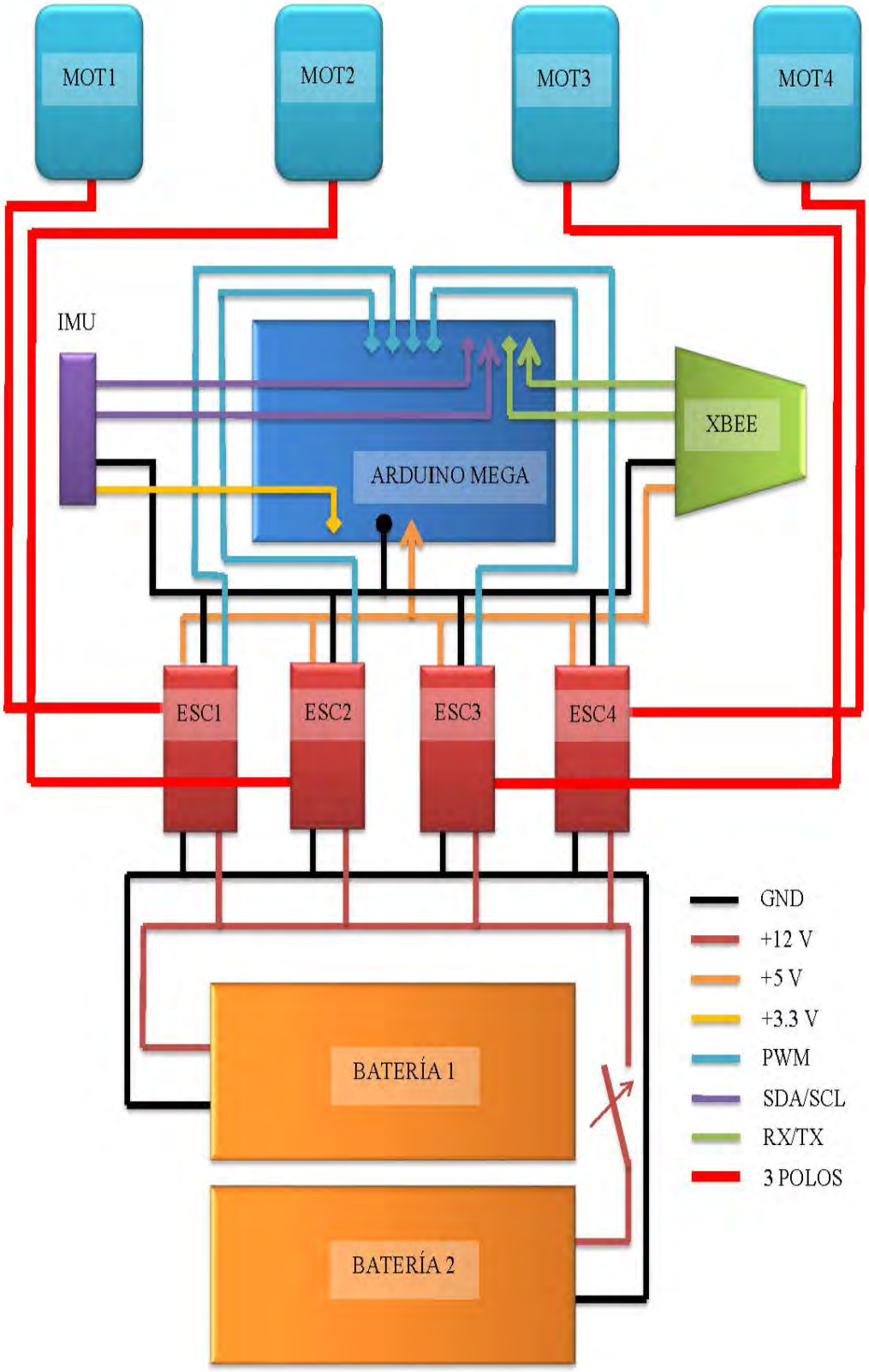
	NOMBRE	FECHA
DIBUJÓ	LUIS A. J.	26/03/11
REVISÓ	SERAFIN C.	29/03/11

P.V.C. DIÁMETRO INTERIOR DE 4", ESPESOR DE 6 MM

CUADRICÓPTERO

TÍTULO CUAD-005

ESCALA: S/E | Unidades: mm | HOJA 5 DE 6


MOT1
MOT2
MOT3
MOT4
IMU
ARDUINO MEGA
XBEE
ESC1
ESC2
ESC3
ESC4
BATERÍA 1
BATERÍA 2
GND
+12 V
+5 V
+3.3 V
PWM
SDA/SCL
RX/TX
3 POLOS

B. Programas

Programa principal completo

```
#include <ITG3200.h>     //Biblioteca del girometro
#include <HMC58X3.h>     //Biblioteca del magnetometro
#include <ADXL345.h>     //Biblioteca del acelerometro
#include <Servo.h>       //Biblioteca de control de servomotores
#include <Wire.h>        //Biblioteca del protocolo I2C
#include <DebugUtils.h> //Biblioteca de correccion para IMU
#include <FreeIMU.h>     //Biblioteca de integracion de datos de IMU
#include <CommunicationUtils.h>     //Biblioteca de comunicacion

int i=0;     //Contador
double vel_ini=0;     //Velocidad inicial para ESC de los motores
Servo M1;     //Creacion de objetos Servo para controlar motores
Servo M2;
Servo M3;
Servo M4;
double limite=900;     //Maximo PWM antes de encender motores
 char id;     //Identificador de ciclo
unsigned long ahora=millis(), transcurridos=ahora;
double tiempo; float ypr[3];     //Arreglo yaw, pitch y roll
FreeIMU sensorIMU = FreeIMU();     //Objeto del tipo FreeIMU
double b=0.0000035, m=3, g=9.8, L=0.3; //Datos del cuadricoptero
double kp=21.8, ki=70.42, kd=3.24;     //Constantes del PID
double Pitch, Roll;     //Variables de los angulos estabilizados
double U1=0, U2=0, U3=0, U4=0;     //Entradas de control
double e=0, rate=0, salida=0;     //Variables del control
double W1, W2, W3, W4;     //Velocidad de los motores

void setup(){
```

```
  Serial2.begin(9600);     //Inicializar comunicacion con XBee
  Wire.begin();     //Inicializar comunicacion con IMU

  M1.attach(2);     //Asignacion de puertos para cada motor
  M2.attach(3);
  M3.attach(4);
  M4.attach(5);

  delay(5);
  sensorIMU.init();     //Inicializar IMU
  delay(5);

  //***********Inicio de comunicacion con los ESC***********//

  for(i=1; i<=((limite-150)/10); i++){
    vel_ini=vel_ini+10;
    M1.writeMicroseconds(vel_ini);     //Contemplar compensacion
    M2.writeMicroseconds(vel_ini);
    M3.writeMicroseconds(vel_ini);
    M4.writeMicroseconds(vel_ini);
    delay(15);
  }
  delay(1000);
  for(i=1; i<=15; i++){
    vel_ini=vel_ini+10;
    M1.writeMicroseconds(vel_ini);     //Con compensacion
    M2.writeMicroseconds(vel_ini);
    M3.writeMicroseconds(vel_ini);
    M4.writeMicroseconds(vel_ini);
    delay(15);
  }
}

void loop(){

  //*****************Ciclo de inicio*****************//

  while(id != 'A'){
    if(Serial2.available()>0){     //Lectura de senal de arranque
      id=Serial2.read();
    }
```

```
}

//**********Aumentando velocidad de motores**********//

for(i=1; i<=60; i++){
  vel_ini=vel_ini+10;
  M1.writeMicroseconds(vel_ini);     //Escribiendo en el puerto
  M2.writeMicroseconds(vel_ini);     //el valor correspondiente
  M3.writeMicroseconds(vel_ini);
  M4.writeMicroseconds(vel_ini);
  Serial2.print("Escribiendo a motores: ");
  Serial2.println(vel_ini, DEC);
  if(Serial2.available()>0){     //Paro en caso de
    id=Serial2.read();        //que la velocidad de los motores
    if(id=='S'){                    //alcance un nivel deseado
      Serial2.print("Ultimo valor: ");
      Serial2.println(vel_ini, DEC);
      break;
    }
  }
  delay(500);
}
   sensorIMU.getYawPitchRoll(ypr); //Primera lectura IMU

   //**********************Control*******************//

 while(id != 'P'){     //Condicion de funcionamiento
  ahora=millis();     //del control antes del paro total
  tiempo=(ahora-transcurridos)*0.001;  //Tiempo entre cada ciclo
  delay(25);  //Retardo mínimo necesario para poder entrar a la
                  //funcion de lectura
              //de la IMU sin bloquear el ciclo
  sensorIMU.getYawPitchRoll(ypr);  //Informacion del sensor

  /////////Filtro de atenuacion/////////
  Roll = smooth(ypr[2], 0.95, Roll);
  Pitch = smooth(ypr[1], 0.95, Pitch);
  Serial2.print("alfa: ");
  Serial2.println(Roll);
  Serial2.print("beta: ");
  Serial2.println(Pitch);
```

```
///////Calculo de entradas U///////
U1=m*g;     //Ya que se asume que no hay cambios de altura
U2=estabiliza_roll()*0.1;
U3=estabiliza_pitch()*0.1;
U4=0;     //Ya que se asume que no hay cambios en gamma

///////Calculo y escritura de las W////////

W1=sqrt(abs((0.25*U1)-(0.5*U3))/b);
W1=map(W1, 0, 8000, 1500, 1900);  //Contemplando un offset
 Serial2.print("W1: ");              //dependiendo de cada motor
Serial2.print(W1);
Serial2.print(", ");
M1.writeMicroseconds(W1-250);

W2=sqrt(abs((0.25*U1)-(0.5*U2))/b);
W2=map(W2, 0, 8000, 1500, 1900);
Serial2.print("W2: ");
Serial2.print(W2);
Serial2.print(", ");
M2.writeMicroseconds(W2);

W3=sqrt(abs((0.25*U1)+(0.5*U3))/b);
W3=map(W3, 0, 8000, 1500, 1900);
Serial2.print("W3: ");
Serial2.print(W3);
Serial2.print(", ");
M3.writeMicroseconds(W3+70);

W4=sqrt(abs((0.25*U1)+(0.5*U2))/b);
W4=map(W4, 0, 8000, 1500, 1900);
Serial2.print("W4: ");
Serial2.println(W4);
M4.writeMicroseconds(W4+30);

delay(80);

if(Serial2.available()>0){     //Lectura de la senal de paro
  id=Serial2.read();
}
```

```
    transcurridos=ahora;
  }

  //************Paro de motores************//
  W1=900;
  W2=900;
  W3=900;
  W4=900;

  M1.writeMicroseconds(W1);
      M2.writeMicroseconds(W2);
      M3.writeMicroseconds(W3);
       M4.writeMicroseconds(W4);

  Serial2.print(W1, DEC);
  Serial2.print(", ");
  Serial2.print(W2, DEC);
  Serial2.print(", ");
  Serial2.print(W3, DEC);
  Serial2.print(", ");
  Serial2.println(W4, DEC);

  delay(100);
  vel_ini=900;
}

//////////*******Estabiliza angulos********//////////
double estabiliza_roll() {
  e = (0 - Roll);     //Angulo deseado - medido
  e = constrain(e, -180, 180);
  rate = pid(e, tiempo, 0.1);     //Entra al algoritmo PID
  return constrain(rate, -1000, 1000);     //Limita control
}

double estabiliza_pitch() {
  e = (0 - Pitch);
  e = constrain(e, -180, 180);
  rate = pid(e, tiempo, 0.1);
  return constrain(rate, -1000, 1000);
}
```

```
//////////*******Atenuador o filtro********//////////
double smooth(double data, double filterVal, double smoothedVal) {
  if (filterVal > 1){       //Revision de rango
    filterVal = .99;     //Maximo filtrado
  }   else if (filterVal <= 0){
    filterVal = 0;    //Minimo filtrado
  }

  smoothedVal = (data * (1-filterVal)) + (smoothedVal*filterVal);

  return smoothedVal;
}

//////////////********Control PID*********//////////

double pid(double error, double dt, double escala) {
  double derivado, derivada, ultimoerror, integrador, imax=200;

 //////Control Proporcional//////

  salida = error*kp;

  ///////Control Derivativo///////

  derivado = (error - ultimoerror) / dt;
  derivada = smooth(derivado, 0.3, derivada);
  ultimoerror = error;
  salida += derivada*kp;

  /////////Control Integral/////////
  double errorabs = abs(error);
     if(errorabs < 10){
    integrador += (error * ki * dt);
    if(integrador < -imax){
      integrador = -imax;
    }
    else if(integrador > imax){
      integrador = imax;
    }
    salida += integrador;
  }
```

```
  return salida;
}
```

FreeIMU.h

```
#include <Wire.h>
#include "WProgram.h"
#include <ADXL345.h>
#include <HMC58X3.h>
#include <ITG3200.h>
#ifndef FreeIMU_h
#define FreeIMU_h

// default I2C 7-bit addresses of the sensors
#define FIMU_ADXL345_DEF_ADDR ADXL345_ADDR_ALT_LOW
// SDO connected to GND
//#define FIMU_ADXL345_DEF_ADDR ADXL345_ADDR_ALT_HIGH
// SDO connected to GND
#define FIMU_ITG3200_DEF_ADDR ITG3200_ADDR_AD0_LOW
// AD0 connected to GND
// HMC5843 address is fixed so don't bother to define it

// ITG3200 constants
#define FIMU_ITG3200_SMPLRT_DIV 0x15
#define FIMU_ITG3200_DLPF_FS 0x16
#define FIMU_ITG3200_INT_CFG 0x17
#define FIMU_ITG3200_PWR_MGM 0x3E

#ifndef cbi #define cbi(sfr, bit) (_SFR_BYTE(sfr) &= ~_BV(bit))
#endif

class FreeIMU {
  public:
    FreeIMU();
   void init();
    void init(bool fastmode);
    void init(bool fastmode, int period);
    void init(int acc_addr, int gyro_addr, bool fastmode,
                                                int period);
    void getRawValues(int * raw_values);
    void getValues(float * values);
    void getQ(float * q);
```

```
    void getEuler(float * angles);
    void getYawPitchRoll(float * ypr);
    // we make them public so that users can
     //interact directly with device classes
    HMC58X3 magn;
    ADXL345 acc;
    ITG3200 gyro;
       private:
    int* raw_acc, raw_gyro, raw_magn;
    void AHRSupdate(float gx, float gy, float gz,
     float ax, float ay, float az, float mx, float my, float mz);
    float q0, q1, q2, q3;
// quaternion elements representing the estimated orientation
    float exInt, eyInt, ezInt; // scaled integral error
    int period, lastUpdate, now;
// sample period expressed in milliseconds
    float halfT; // half the sample period expressed in seconds
   int startLoopTime;
};

float invSqrt(float number);

#endif
// FreeIMU_h
```

FreeIMU.cpp

```
#include <inttypes.h>
//#define DEBUG
#include "WProgram.h"
#include "FreeIMU.h"
// #include "WireUtils.h"
#include "DebugUtils.h"

//--------------------------------------
// Definitions

#define Kp 2.0f
  // proportional gain governs rate of convergence
  // to accelerometer/magnetometer
#define Ki 0.005f
  // integral gain governs rate of convergence of
```

```
  // gyroscope biases
//#define halfT 0.02f
  // half the sample period

FreeIMU::FreeIMU() {
  acc = ADXL345();
  gyro = ITG3200();
  magn = HMC58X3();
     // initialize quaternion
  q0 = 1;
  q1 = 0;
  q2 = 0;
  q3 = 0;
  exInt = 0;
  eyInt = 0;
  ezInt = 0;
  lastUpdate = 0;
  now = 0;
}

void FreeIMU::init() {
  init(FIMU_ADXL345_DEF_ADDR, FIMU_ITG3200_DEF_ADDR,
          false, 20);
}

void FreeIMU::init(bool fastmode) {
  init(FIMU_ADXL345_DEF_ADDR, FIMU_ITG3200_DEF_ADDR,
          fastmode, 20);
}

void FreeIMU::init(bool fastmode, int newperiod) {
  // use default addresses
  init(FIMU_ADXL345_DEF_ADDR, FIMU_ITG3200_DEF_ADDR,
          fastmode, newperiod);
}

void FreeIMU::init(int acc_addr, int gyro_addr, bool
          fastmode, int newperiod) {
  delay(5);
        // disable internal pullups of the ATMEGA
          // which Wire enable by default
```

```
  #if defined(__AVR_ATmega168__) || defined(__AVR_ATmega8__)
          || defined(__AVR_ATmega328P__)
    // deactivate internal pull-ups for twi
    // as per note from atmega8 manual pg167
    cbi(PORTC, 4);
    cbi(PORTC, 5);
  #else
    // deactivate internal pull-ups for twi
    // as per note from atmega128 manual pg204
    cbi(PORTD, 0);
    cbi(PORTD, 1);
  #endif
       if(fastmode) {
// switch to 400KHz I2C - eheheh
    TWBR = ((16000000L / 400000L) - 16) / 2;
// see twi_init in Wire/utility/twi.c
    // TODO: make the above usable also for 8MHz arduinos..
  }
     period = newperiod;
  halfT = newperiod / (1000.0 * 2);
// store the half of the period expressed in seconds
     // init ADXL345   acc.init(acc_addr);
  // init ITG3200   gyro.init(gyro_addr);
  // calibrate the ITG3200   gyro.zeroCalibrate(64,5);
     // init HMC5843   magn.init(false);
// Don't set mode yet, we'll do that later on.
  // Calibrate HMC using self test,
  //not recommended to change the gain after calibration.
  magn.calibrate(1);
// Use gain 1=default, valid 0-7, 7 not recommended.
  // Single mode conversion was used in
  //calibration, now set continuous mode
  magn.setMode(0);
  delay(10);
  magn.setDOR(B110);
}

void FreeIMU::getRawValues(int * raw_values) {
  acc.readAccel(&raw_values[0], &raw_values[1],
         &raw_values[2]);
  gyro.readGyroRaw(&raw_values[3], &raw_values[4],
```

```
            &raw_values[5]);
  magn.getValues(&raw_values[6], &raw_values[7],
            &raw_values[8]);
}

void FreeIMU::getValues(float * values) {
    int accval[3];
  acc.readAccel(&accval[0], &accval[1], &accval[2]);
  values[0] = ((float) accval[0]);
  values[1] = ((float) accval[1]);
  values[2] = ((float) accval[2]);
    gyro.readGyro(&values[3]);
    magn.getValues(&values[6]);
}

//=====================
// AHRS.c
// S.O.H. Madgwick
// 25th August 2010
//=====================
//Description: //
// Quaternion implementation of the 'DCM filter'
// [Mayhony et al].
// Incorporates the magnetic distortion
// compensation algorithms from my filter [Madgwick]
// which eliminates the need for a reference
// direction of flux (bx bz) to be predefined
// and limits the effect of magnetic distortions to yaw
// axis only.
//
// User must define 'halfT' as the (sample period / 2),
// and the filter gains 'Kp' and 'Ki'.
//
// Global variables 'q0', 'q1', 'q2', 'q3' are the
// quaternion elements representing the estimated
// orientation.  See my report for an overview of the
// use of quaternions in this application.
//
// User must call 'AHRSupdate()' every sample period
// and parse calibrated gyroscope ('gx', 'gy', 'gz'),
// accelerometer ('ax', 'ay', 'ay') and magnetometer
```

```
// ('mx', 'my', 'mz') data.  Gyroscope units are
// radians/second, accelerometer and magnetometer
// units are irrelevant as the vector is normalised.
//
//=====================================

//**********FILTRO DCM***********//

void FreeIMU::AHRSupdate(float gx, float gy, float gz,
     float ax, float ay, float az, float mx, float my,
          float mz) {
  float norm;
  float hx, hy, hz, bx, bz;
  float vx, vy, vz, wx, wy, wz;
  float ex, ey, ez;

  // auxiliary variables to reduce number of repeated
     // operations
  float q0q0 = q0*q0;
  float q0q1 = q0*q1;
  float q0q2 = q0*q2;
  float q0q3 = q0*q3;
  float q1q1 = q1*q1;
  float q1q2 = q1*q2;
  float q1q3 = q1*q3;
  float q2q2 = q2*q2;
  float q2q3 = q2*q3;
  float q3q3 = q3*q3;

                // normalise the measurements
    norm = sqrt(ax*ax + ay*ay + az*az);
         ax = ax / norm;    ay = ay / norm;
  az = az / norm;
       now = millis();
  halfT = (now - lastUpdate) / 2000.0;
  lastUpdate = now;
    /*    norm = invSqrt(ax*ax + ay*ay + az*az);
  ax = ax * norm;
  ay = ay * norm;
  az = az * norm;    */
    norm = sqrt(mx*mx + my*my + mz*mz);
```

```
            mx = mx / norm;
  my = my / norm;
  mz = mz / norm;
     /*    norm = invSqrt(mx*mx + my*my + mz*mz);
  mx = mx * norm;
  my = my * norm;
  mz = mz * norm;   */

    // compute reference direction of flux
  hx = 2*mx*(0.5 - q2q2 - q3q3) + 2*my*(q1q2 - q0q3)
          + 2*mz*(q1q3 + q0q2);
  hy = 2*mx*(q1q2 + q0q3) + 2*my*(0.5 - q1q1 - q3q3)
          + 2*mz*(q2q3 - q0q1);
  hz = 2*mx*(q1q3 - q0q2) + 2*my*(q2q3 + q0q1)
          + 2*mz*(0.5 - q1q1 - q2q2);
           bx = sqrt((hx*hx) + (hy*hy));
  bz = hz;
          // estimated direction of gravity and flux (v and w)
  vx = 2*(q1q3 - q0q2);
  vy = 2*(q0q1 + q2q3);
  vz = q0q0 - q1q1 - q2q2 + q3q3;
  wx = 2*bx*(0.5 - q2q2 - q3q3) + 2*bz*(q1q3 - q0q2);
  wy = 2*bx*(q1q2 - q0q3) + 2*bz*(q0q1 + q2q3);
  wz = 2*bx*(q0q2 + q1q3) + 2*bz*(0.5 - q1q1 - q2q2);

      // error is sum of cross product between reference
     //direction of fields and direction measured by sensors

 ex = (ay*vz - az*vy) + (my*wz - mz*wy);
  ey = (az*vx - ax*vz) + (mz*wx - mx*wz);
  ez = (ax*vy - ay*vx) + (mx*wy - my*wx);
     // integral error scaled integral gain
  exInt = exInt + ex*Ki;
  eyInt = eyInt + ey*Ki;
  ezInt = ezInt + ez*Ki;
// adjusted gyroscope measurements
  gx = gx + Kp*ex + exInt;
  gy = gy + Kp*ey + eyInt;
  gz = gz + Kp*ez + ezInt;
    // integrate quaternion rate and normalise
  q0 = q0 + (-q1*gx - q2*gy - q3*gz)*halfT;
```

```
  q1 = q1 + (q0*gx + q2*gz - q3*gy)*halfT;
  q2 = q2 + (q0*gy - q1*gz + q3*gx)*halfT;
  q3 = q3 + (q0*gz + q1*gy - q2*gx)*halfT;
      // normalise quaternion
      norm = sqrt(q0*q0 + q1*q1 + q2*q2 + q3*q3);
  q0 = q0 / norm;
  q1 = q1 / norm;
  q2 = q2 / norm;
  q3 = q3 / norm;
   /*   norm = invSqrt(q0*q0 + q1*q1 + q2*q2 + q3*q3);
  q0 = q0 * norm;
  q1 = q1 * norm;
  q2 = q2 * norm;
  q3 = q3 * norm;
  */
}
void FreeIMU::getQ(float * q) {
  float val[9];
  getValues(val);
     DEBUG_PRINT(val[3] * M_PI/180);
  DEBUG_PRINT(val[4] * M_PI/180);
  DEBUG_PRINT(val[5] * M_PI/180);
  DEBUG_PRINT(val[0]);
  DEBUG_PRINT(val[1]);
  DEBUG_PRINT(val[2]);
  DEBUG_PRINT(val[6]);
  DEBUG_PRINT(val[7]);
  DEBUG_PRINT(val[8]);
     // gyro values are expressed in deg/sec,
     //the * M_PI/180 will convert it to radians/sec
  AHRSupdate(val[3] * M_PI/180, val[4] * M_PI/180,
     val[5] * M_PI/180, val[0], val[1], val[2], val[6],
          val[7], val[8]);
  q[0] = q0;
  q[1] = q1;
  q[2] = q2;
  q[3] = q3;
}

// Returns the Euler angles in radians defined with
// the Aerospace sequence.
```

```
// See Sebastian O.H. Madwick report
// "An efficient orientation filter for inertial and
//intertial/magnetic sensor arrays" Chapter 2
//Quaternion representation
void FreeIMU::getEuler(float * angles) {
  float q[4];
// quaternion
  getQ(q);
  angles[0] = atan2(2 * q[1] * q[2] - 2 * q[0] * q[3], 2 *
          q[0]*q[0] + 2 * q[1] * q[1] - 1) * 180/M_PI;
// psi    angles[1] = -asin(2 * q[1] * q[3] + 2 *
          q[0] * q[2]) * 180/M_PI;
// theta    angles[2] = atan2(2 * q[2] * q[3] - 2 *
          q[0] * q[1], 2 *
          q[0] * q[0] + 2 * q[3] * q[3] - 1) * 180/M_PI;
// phi
}

void FreeIMU::getYawPitchRoll(float * ypr) {
  float q[4];
// quaternion
  float gx, gy, gz;
// estimated gravity direction
  getQ(q);
     gx = 2 * (q[1]*q[3] - q[0]*q[2]);
  gy = 2 * (q[0]*q[1] + q[2]*q[3]);
  gz = q[0]*q[0] - q[1]*q[1] - q[2]*q[2] + q[3]*q[3];
     ypr[0] = atan2(2 * q[1] * q[2] - 2 * q[0] * q[3], 2
          * q[0]*q[0] + 2 * q[1] * q[1] - 1) * 180/M_PI;
  ypr[1] = atan(gx / sqrt(gy*gy + gz*gz))  * 180/M_PI;
  ypr[2] = atan(gy / sqrt(gx*gx + gz*gz))  * 180/M_PI;
}

float invSqrt(float number) {
  volatile long i;
  volatile float x, y;
  volatile const float f = 1.5F;

  x = number * 0.5F;
  y = number;
  i = * ( long * ) &y;
```

```
  i = 0x5f375a86 - ( i >> 1 );
  y = * ( float * ) &i;
  y = y * ( f - ( x * y * y ) );
  return y;
}
```