UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

mim máster en ingeniería mecatrónica

Control de actitud de un cuadricóptero

MEMORIA PRESENTADA POR:

José Danilo Guerra Quiñones

Máster Universitario en Ingeniería Mecatrónica

DIRECTOR:

[Vicente Fermín Casanova Calvo]

[Julio,2018]

Índice de contenidos

1. Introducción 6
1.1. Objetivos 6
1.1.1 Objetivos específicos 6
1.2. Justificación 7
2. Configuración y Modelado del cuadricóptero 8
2.1. Configuración del cuadricóptero 8
2.2. Movimiento del cuadricóptero 8
2.2.1 Throttle (empuje) 9
2.2.2 *Roll (*alabeo) 9
2.2.3 Pitch (cabeceo) 9
2.2.4 *Yaw* (guiñada) 9
2.3. Modelado Matemático del Cuadricóptero 11
2.3.1 Ecuaciones de Newton-Euler 13
2.3.2 Ecuaciones de Euler-Lagrange 15
2.3.3 Efectos Aerodinámicos 16
3. Diseño y Simulación 17
3.1. Diseño CAD 17
3.1.1 Cuerpo o Cuadro 17
3.1.2 Brazos 18
3.1.3 Motores 18
3.1.4 Hélices 19
3.1.5 Ensamble 19
3.2. Modelado en Simulink 20
3.2.1 Sistema de control 22
3.2.2 Selección del controlador 24
1.1.1. Selección del método de sintonización 27
3.2.3 Sintonización del controlador 27
3.2.4 Desarrollo del sistema de control 28
3.3. Resultados de la simulación 30
3.4. Referencia de escalón de 15° 30
3.5. Referencia senoidal de -20° a 20° 32
3.6. Referencia de escalón variable 34

4. Implementación Real 37
4.1. Selección de componentes 37
4.1.1 Estructura 37
1.1.1. Hélices 37
4.1.2 Motores 37
4.1.3 Control electrónico de velocidad (ESC) 38
4.1.4 Batería 39
4.1.5 Tarjeta de distribución eléctrica 39
4.1.6 Placa controladora o circuito de control 39
4.1.7 Sensores 41
4.2. Programacion/Código Arduino 42
4.2.2 Resultados de las pruebas reales 46
5. Conclusión 47
6. Bibliografía 48
7. Anexos 50
7.1. Presupuesto del Proyecto 50
7.2. Planos y hoja de especificaciones 51
7.2.1 Motor (x4) 51
7.2.2 Hélices (x4) 52
7.2.3 ESC (Variadores de velocidad) (x4) 53
7.2.4 Ensamble del cuadricóptero 58
7.2.5 Arduino DUE 59

Índice de figuras

Figura 1. Configuración en cruz (+) (izquierda). Configuración en cruz (x) (derecha). 8
Figura 2. Ángulo de alabeo 9
Figura 3. Ángulo de cabeceo 9
Figura 4. Ángulo de guiñada 10
Figura 5. Ubicación de los Motores en la configuración equis (x) de un cuadricóptero 10
Figura 6. Sistema de referencia del cuadricóptero. [4] 11
Figura 7. Cuerpo del cuadricóptero. 17
Figura 8. Brazos del cuadricóptero. 18
Figura 9. Motores del cuadricóptero. 18
Figura 10. Hélices con sentido de giro horario. 19
Figura 11. Hélices con sentido de giro antihorario. 19
Figura 12. Ensamble de cuadricóptero. 19
Figura 13. Librería de Simscape Multibody. 20
Figura 14. Modelo 3D en Simulink. 21
Figura 15. Vista en detalle del modelo 3D. 21
Figura 16. Ejemplo de las características de la pieza. 22
Figura 17. Diferentes valores de la ganancia proporcional. [9] 23
Figura 18. Controlador P. 24
Figura 19. Control Proporcional Integral PI. 25
F**igura 20.** Controlador Proporcional Derivativo. 26
Figura 21. Controlador Proporcional Integral Derivativo (PID) 26
Figura 22.Controlador PD para el ángulo de roll (alabeo). 28
Figura 23. Controlador PD para el ángulo de Pitch (cabeceo). 29
Figura 24. Controlador P para el ángulo de yaw (guiñada). 29
Figura 25. Modelo en Simulink con los controladores aplicados. 30
Figura 26. Prueba referencia de escalón de 15°. 30
Figura 27. Referencia vs ángulo de alabeo. **Figura 28.** Referencia vs ángulo de cabeceo. 31
Figura 29. Referencia vs ángulo de guiñada. 31
Figura 30. Seguimiento de la referencia de escalón 15°. 32
Figura 31. Prueba referencia senoidal -20° a 20°. 32
Figura 32. Referencia vs ángulo de alabeo. 33
Figura 33. Referencia vs ángulo de cabeceo. 33
Figura 34. Referencia vs ángulo de guiñada. 33
Figura 35. Seguimiento de la referencia senoidal de -20° a 20°. 34
Figura 36. Referencia de escalón variable vs ángulo de cabeceo. 34

Figura 37. Referencia vs ángulo de guiñada. 35
Figura 38. Referencia vs ángulo de alabeo. 35
Figura 39. Movimiento de los ángulos de cabeceo, alabeo y guiñada. 35
Figura 40. Seguimiento de la referencia variable. 36
Figura 41. Cuerpo y brazos del cuadricóptero. 37
Figura 42. Partes de un ESC. 38
Figura 43. Tarjeta de distribución eléctrica. 39
Figura 44. Placa de desarrollo Arduino Due. 40
Figura 45. Arduino 9 Axis Motion Shield. 41
Figura 46. Arduino WIFI Shield 42
Figura 47. Referencia vs ángulo de alabeo 46

1. Introducción

Un cuadricóptero se define como un helicóptero multi-rotor que consta de cuatro brazos, los cuales tienen en su parte final un motor y una hélice, su principio de funcionamiento es muy similar a los helicópteros en muchos aspectos, aunque a diferencia de estos, la elevación y el empuje en el cuadricóptero se realiza con cuatro hélices en vez de una.

A medida que pasaron los años y la tecnología mejoro, los cuadricópteros se convirtieron en una solución para las operaciones de reconocimiento, tal es el caso de los cuadricópteros utilizados por el escuadrón de infantería de los EE. UU. [1]. Tienen como propósito medir las condiciones en la playa y descubrir y trazas mapas de las posiciones enemigas además de recopilar datos cartográficos para utilizarlos en juegos de decisión táctica, que presentan situaciones hipotéticas a los infantes de marina y los desafían a encontrar una solución rápidamente.

Como la mayoría de la tecnología militar, era cuestión de tiempo para que los cuadricópteros se empezaran a ser aprobados para un uso civil ya sea para el ocio o para la investigación personal. Gracias a proyectos de código abierto como por ejemplo el *ArduPilot* [2] basado en la plataforma electrónica de Arduino, tiene como objetivo el hacer de los cuadricópteros más accesibles a la gente proporcionando el sistema de control y una integración con una gran cantidad de sensores utilizados en los cuadricópteros.

1.1. Objetivos

El objetivo de este proyecto es el desarrollo e implementación de un sistema de control para el control de los ángulos de cabeceo, alabeo y guiñada de un cuadricóptero, además de integrarlo con la parte mecánica, la estructura y con la parte electrónica que corresponde con la tarjeta de control y los sensores.

1.1.1 Objetivos específicos

- Modelado y simulación del sistema de control diseñado.
- Implementación real del sistema de control diseñado en un cuadricóptero utilizando una placa de Arduino.

1.2. Justificación

En este proyecto se propone el diseño de un sistema de control capaz de controlar los ángulos de navegación de Euler que son el cabeceo, guiñada y el alabeo, lo cual permitirá el inicio del desarrollo de una controladora de vuelo que hará posible convertir el cuadricóptero en un VANT (vehículo aéreo no tripulado) o un DRON.

Para este desarrollo de este sistema de control, primero se desarrollará el modelado teórico del movimiento de un cuadricóptero donde se definen las ecuaciones que definen el comportamiento de los ángulos de cabeceo, alabeo y guiñada.

La parte fundamental de todo cuadricóptero es el sistema de control que se encargará de administrar, dirigir y regular el funcionamiento del cuadricóptero, este desarrollo estará apoyado de Matlab/Simulink que nos proporciona un entorno de desarrollo y simulación para sistemas de control. Dentro de las limitantes se encuentra el sistema de control para este sistema debido al tiempo destinado para desarrollar dicho sistema y lo complejo que resulta estabilizar sistemas de este tipo. El proyecto finalizara cuando se realice el correcto modelado y simulación del sistema de control además de tratar de realizar una implementación real del sistema de control.

2. Configuración y Modelado del cuadricóptero

2.1. Configuración del cuadricóptero

Los cuadricópteros, los multirrotores más comunes constan de cuatro motores dispuestos simétricamente en una disposición en cruz (+) o equis (x). En una disposición en cruz la parte delantera del cuadricóptero se alinea directamente con un motor. En la disposición en equis la parte delantera del cuadricóptero se sitúa en medio de los dos motores delanteros.

De los cuatro motores en un cuadricóptero, dos giran en sentido horario y dos en sentido contrario (ver figura 1). Esto es para compensar el par motor creado y mantener el cuadricóptero mirando hacia la dirección correcta, básicamente es la misma función que realiza el rotor de la cola de un helicóptero. Para controlar la velocidad del cuadricóptero las velocidades en los motores son variadas, es decir, para hacer que el cuadricóptero vaya hacia adelante los dos motores delanteros disminuyen su velocidad mientras que los motores traseros la aumentan.

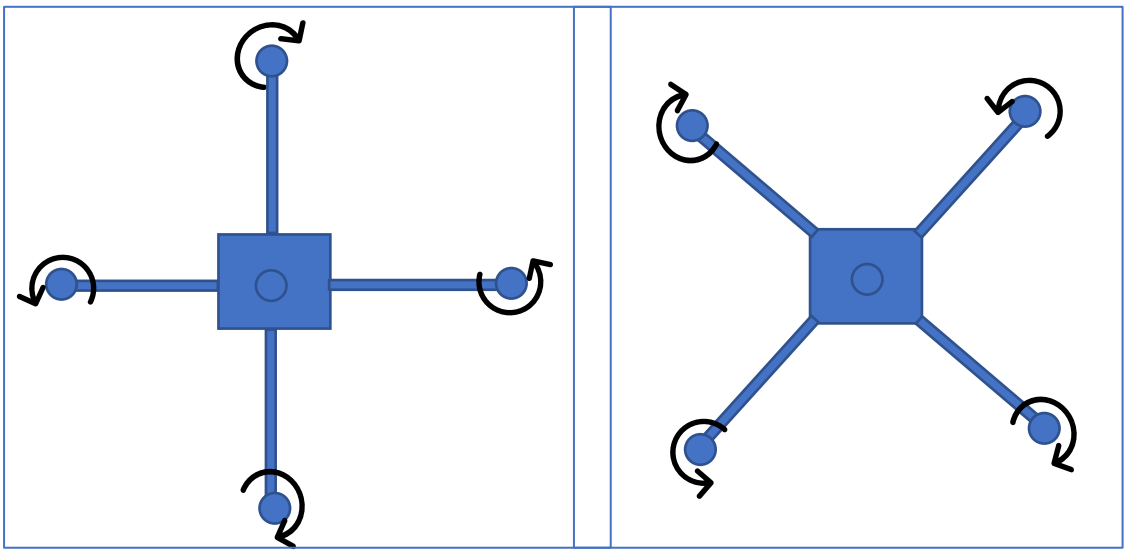


Figura 1. *Configuración en cruz (+) (izquierda). Configuración en cruz (x) (derecha).*

La configuración en equis es la más útil y común, ya que los brazos y las hélices son menos visibles en caso de utilizar una cámara abordo orientada hacia adelantes.

2.2. Movimiento del cuadricóptero

Cualquiera de las dos configuraciones permite cuatro movimientos básicos, los cuales habilitan al cuadricóptero para alcanzar una posición exacta.

El cuadricóptero puede ser considerado como un cuerpo rígido en el espacio, sujeto a una fuerza principal que produce el movimiento “throttle” (empuje) y tres momentos producidos por desequilibrios en las fuerzas de empuje de los rotores. Estos momentos producen el movimiento de “roll”, “pitch” y “yaw” (alabeo, cabeceo y guiñada).

2.2.1 Throttle (empuje)

Este movimiento es proporcionado por el aumento (o disminución) de velocidad de todos los motores en la misma cantidad generando una fuerza vertical que sube, baja o mantiene suspendido el cuadricóptero respecto a su propio sistema inercial. Sin embargo, si el cuadricóptero no se encuentra en posición horizontal respecto al sistema de referencia inercial (fijado en la tierra), el empuje proporcionado por lo rotores generará aceleraciones verticales y horizontales.

2.2.2 *Roll (*alabeo)

Este movimiento es proporcionado por el aumento (o disminución) de velocidad de los motores de la izquierda 1 y 3 y por disminución (o aumento) de la velocidad en los motores de la derecha 2 y 4 (ver figura 3). Esto produce un par de torsión con respecto al eje x que hace girar el cuadricóptero. Este desequilibro de fuerzas sólo conduce a una aceleración en el ángulo de alabeo ϕ.

2.2.3 Pitch (cabeceo)

Este movimiento es muy similar al roll y se genera mediante el aumento (o disminución) la velocidad de los motores del frente 1 y 2, y por la disminución (o aumento) de los motores de atrás 3 y 4 (ver figura 2). El par de torsión con respecto al eje y hace girar el cuadricóptero. Este desequilibrio de fuerzas sólo conduce a una aceleración en el ángulo de cabeceo θ.

2.2.4 *Yaw* (guiñada)

Este movimiento es proporcionado por el aumento (o disminución) velocidad de los motores 1 o 2 y por la disminución (o aumento) de los motores 3 o 4 (ver figura 4). Esto conduce a un par de torsión con respecto al eje z que hace girar el cuadricóptero. El movimiento de guiñada se genera gracias a que las hélices de izquierda y derecha giran en sentido horario mientras que las hélices delantero-trasero giran en sentido antihorario. Este movimiento sólo conduce a una aceleración de ángulo de guiñada ψ.

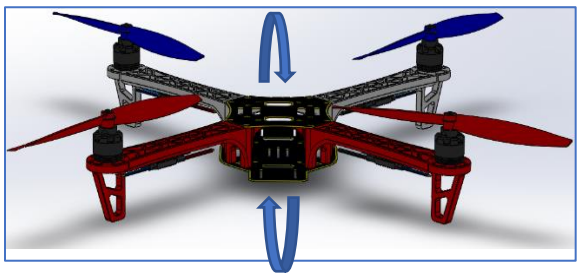


***Figura 2**. Ángulo de cabeceo*

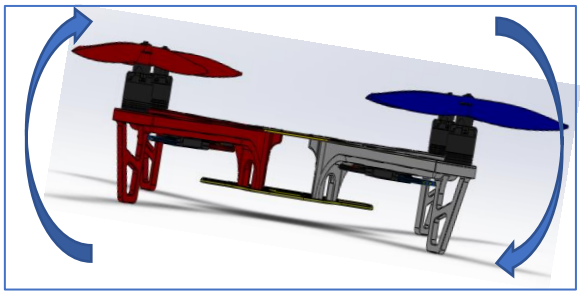


***Figura 3.** Ángulo de alabeo*

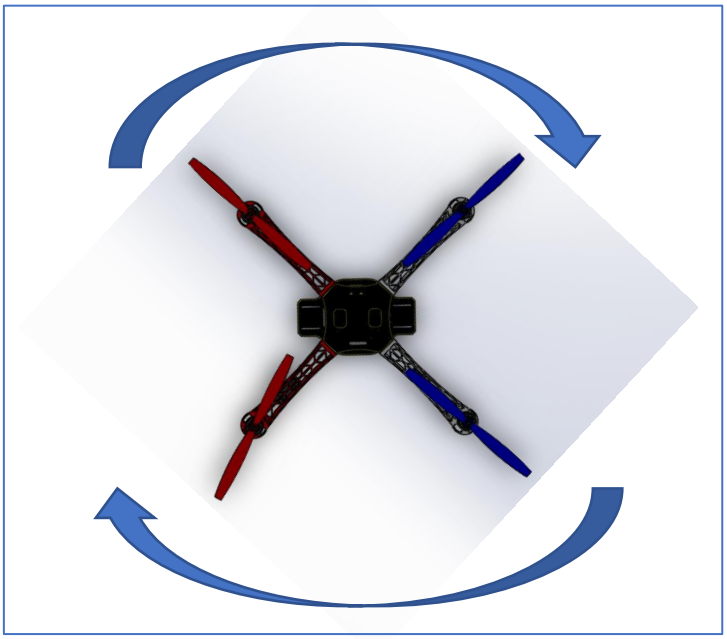


***Figura 2**. Ángulo de guiñada*

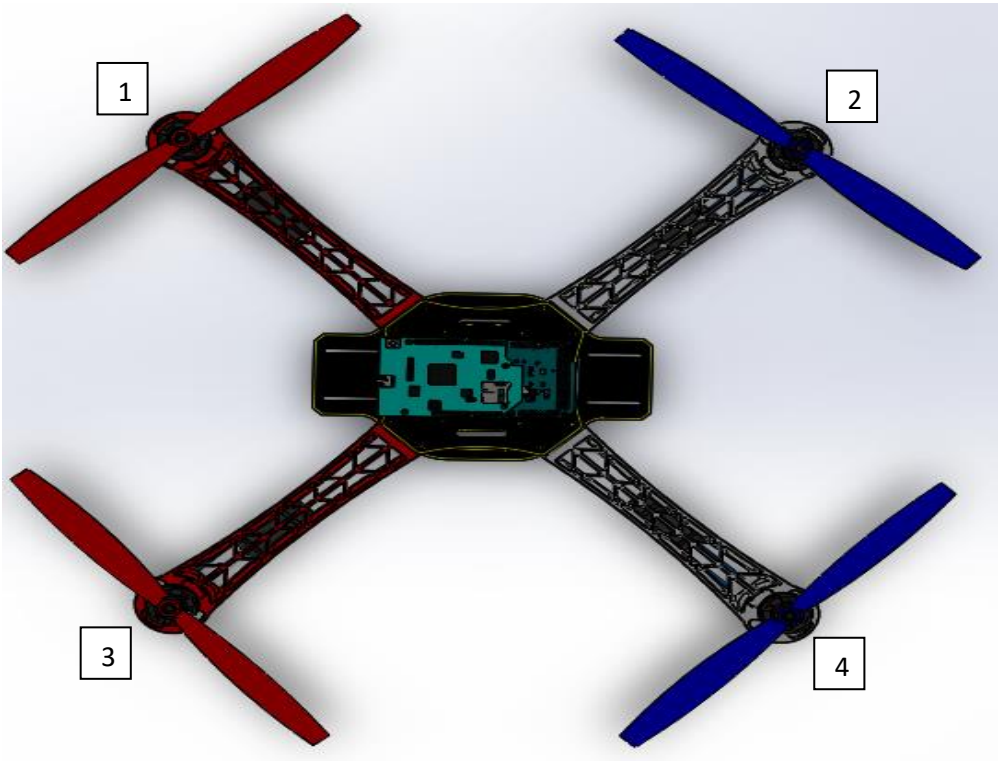


***Figura 3.** Ubicación de los Motores en la configuración equis (x) de un cuadricóptero*

2.3. Modelado Matemático del Cuadricóptero

Una de las herramientas más interesantes que actualmente disponemos para analizar y predecir el comportamiento de un sistema es la construcción y posterior simulación de un modelo matemático.

Para el ingeniero y el científico un modelo es todo lo que se emplea para describir la estructura o el comportamiento de una contraparte de la vida real. Con modelos se logra esto mediante palabras, números, símbolos especiales, diagramas, gráficas o semejanza en cuento a apariencia o en cuanto a comportamiento con las contrapartes de la vida real que representan. Esto quiere decir que un Modelo es sinónimo de una representación.

Para describir y representar el modelo matemático del cuadricóptero se usará como punto de partida el modelo presentado en el artículo de investigación de Teppo Luukkonen [3].

A continuación, se presenta el modelado matemático de cuadricóptero en configuración equis, donde se analizará la estructura del cuadricóptero, incluyendo las velocidades angulares, pares y fuerzas creadas por los cuatro rotores.

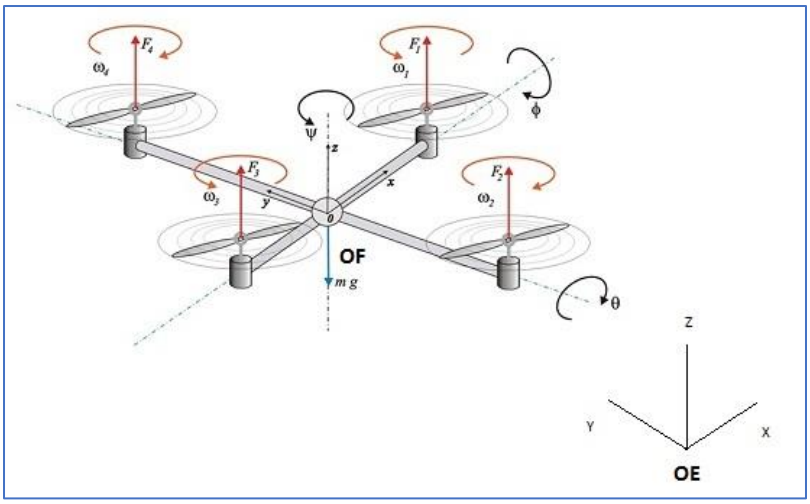


Figura 4. *Sistema de referencia del cuadricóptero. [4]*

La posición lineal absoluta del cuadricóptero se define en el sistema de referencia inercial, ejes x, y, z con ξ. La posición angular, se define en el sistema de referencia inercial con tres ángulos de Euler η. El ángulo de **cabeceo (pitch)** "θ" determina la rotación del cuadricóptero sobre el eje Y. El ángulo de **alabeo (roll)** "ϕ" determina la rotación sobre el eje X y el ángulo de **guiñada (yaw)** "ψ" sobre el eje Z. El vector q contiene los vectores de la posición lineal y angular.

$$\xi = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \qquad \xi = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}, \quad q = \begin{bmatrix} \xi \\ \eta \end{bmatrix}. \tag{1}$$

El origen del sistema de referencia del cuadricóptero está situado el centro de masas de este. En el sistema de referencia, las velocidades lineales son determinadas por VB y las velocidades angulares por υ.

$$V_B = \begin{bmatrix} v_x, B \\ v_y, B \\ v_z, B \end{bmatrix}, \quad \xi = \begin{bmatrix} p \\ q \\ r \end{bmatrix}. \tag{2}$$

La matriz de rotación del sistema de referencia respecto al sistema de referencia inercial es

$$R = \begin{bmatrix} C_\psi C_\theta & C_\psi C_\theta S_\phi - S_\psi C_\phi & C_\psi S_\theta C_\phi + S_\psi C_\phi \\ S_\psi C_\theta & S_\psi S_\theta S_\phi + C_\psi C_\phi & S_\psi S_\theta C_\phi - C_\psi S_\phi \\ -S_\theta & C_\theta S_\phi & C_\theta C_\phi \end{bmatrix}, \tag{3}$$

En donde, Sx=sin (x) y Cx=cos (x). La matriz de rotación R es ortogonal por lo que $R^{-1} = R^T$, donde es matriz de rotación desde el sistema de referencia inercial al sistema de referencia del cuadricóptero.

"La matriz de transformación para las velocidades angulares desde el sistema de referencia inercial al sistema de referencia del cuadricóptero es W_η , y para el sistema de referencia del cuadricóptero al sistema de referencia inercia es W_η^{-1}." [5].

$$\begin{aligned} R\eta = \dot{W}_\eta^{-1} v, \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} &= \begin{bmatrix} 1 & S_\phi T_\theta & C_\phi T_\theta \\ 0 & C_\phi & -S_\phi \\ 0 & S_\phi / C_\theta & C_\phi / C_\theta \end{bmatrix} \begin{bmatrix} \mathrm{p} \\ q \\ \mathrm{r} \end{bmatrix}, \\ v = \mathrm{W}_\eta \dot{\eta}, \qquad \begin{bmatrix} \mathrm{p} \\ q \\ \mathrm{r} \end{bmatrix} &= \begin{bmatrix} 1 & 0 & -S_\theta \\ 0 & C_\phi & C_\theta S_\phi \\ 0 & -S_\phi & C_\theta C_\phi \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}, \end{aligned} \tag{4}$$

En donde $T_x = tan\,(x)$. La matriz W_η es invertible si $\theta \neq (2k-1)\phi 2\,/, (k \epsilon Z)$.

Se asume que el cuadricóptero tiene una estructura simétrica con los cuatro brazos alineados respecto a los ejes X e Y. Por lo que, la matriz de inercia es una matriz diagonal I en la que $I_{xx} = I_{yy}$.

$$R = I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}. \tag{5}$$

La velocidad angular del rotor i, se denotada con w_i, crea una fuerza f_i, en dirección del eje del rotor. La velocidad angular y la aceleración del rotor crean una fuerza de par T_{Mi}, alrededor del eje del rotor.

$$f_i = kw_i^2, T_{Mi} = bw_i^2 + I_M\dot{w}_\iota, \tag{6}$$

en la que la constante de elevación es k, la constante de arrastre es b y el momento de inercia del rotor es I_M. Usualmente el efecto de $\dot{w}_\iota$ se considera pequeño y por lo tanto se omite.

La combinación de fuerzas del rotor crea una fuerza de empuje T en la dirección del eje Z del cuerpo. El Torque T_B está formada por los torques τ_ϕ, τ_θ y τ_ψ en la dirección correspondiente a los ángulos del sistema de referencia del cuadricóptero.

$$T = \sum_{i=1}^{4} f_i = k\sum_{i=1}^{4} w_i^2, T^B = \begin{bmatrix} 0 \\ 0 \\ \mathrm{T} \end{bmatrix}, \tag{7}$$

$$T_B = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} l\,k\,(-w_2^2 + w_4^2) \\ l\,k\,(-w_1^2 + w_3^2) \\ \sum_{i=1}^{4} T_{Mi} \end{bmatrix} \tag{8}$$

donde l es la distancia entre el rotor y el centro de masas del cuadricóptero. Por lo tanto, el movimiento de ***roll (alabeo)*** se consigue disminuyendo la velocidad del segundo rotor y aumentando la velocidad del cuarto rotor. Del mismo modo, el movimiento de ***pitch (cabeceo)*** se consigue disminuyendo la velocidad del primer rotor y aumentando la velocidad del cuarto rotor. Por último, el movimiento ***yaw (guiñada)*** se consigue aumentando la velocidad angular de los dos rotores opuestas o disminuyendo la de los otros dos opuestos.

2.3.1 Ecuaciones de Newton-Euler

El cuadricóptero se considera un cuerpo rígido y por lo tanto las ecuaciones de Newton-Euler se pueden utilizar para describir su dinámica. En el sistema de referencia del cuadricóptero las fuerzas requeridas para la aceleración de la masa $m\dot{V}_B$ y la fuerza centrifuga $v \times (mV_B)$ son iguales a la gravedad $R^T G$ y la fuerza de empuje total de los cuatro rotores T_B

$$m\dot{V}_B + v = (mV_B)bw_i^2 = R^T G + T_B, \tag{9}$$

En el sistema de referencia inercial, la fuerza centrífuga es anulada. Por lo tanto, sólo la fuerza de gravedad, la magnitud y la dirección de empuje contribuyen a la aceleración del cuadricóptero.

$$m\ddot{\xi} = G + R\,T_B,$$

$$\begin{bmatrix}\ddot{x}\\ \ddot{y}\\ \ddot{z}\end{bmatrix} = -g\begin{bmatrix}0\\0\\1\end{bmatrix} + \frac{T}{m}\begin{bmatrix}C_\psi S_\theta C_\phi + S_\psi C_\phi\\ S_\psi S_\theta C_\phi - C_\psi S_\phi\\ C_\psi S_\phi\end{bmatrix}. \tag{10}$$

En el sistema de referencia del cuadricóptero, la aceleración angular de la inercia $I\dot{v}$, las fuerzas centrípetas $v \times (Iv)$ y las fuerzas giroscópicas Γ son iguales al torque externo τ

$$I\dot{v} + v\;x\;(Iv) + \Gamma = \tau$$

$$\dot{v} = I^{-1}\left(-\begin{bmatrix}p\\q\\r\end{bmatrix} \times \begin{bmatrix}I_{xx}p\\I_{yy}q\\I_{zz}r\end{bmatrix} - I_r\begin{bmatrix}p\\q\\r\end{bmatrix} \times \begin{bmatrix}0\\0\\0\end{bmatrix}w_r + \tau\right),$$

$$\begin{bmatrix}\dot{p}\\\dot{q}\\\dot{r}\end{bmatrix} = \begin{bmatrix}(I_{yy}-I_{zz})qr/I_{xx}\\(I_{zz}-I_{xx})qr/I_{yy}\\(I_{xx}-I_{yy})qr/I_{zz}\end{bmatrix} - I_r\begin{bmatrix}\frac{q}{I_{xx}}\\-\frac{p}{I_{yy}}\\0\end{bmatrix}w_r + \begin{bmatrix}\frac{\tau_\phi}{I_{xx}}\\\frac{\tau_\theta}{I_{yy}}\\\frac{\tau_\psi}{I_{zz}}\end{bmatrix}, \tag{11}$$

donde $w_r = w_1 - w_2 + w_3 - w_4$. Las aceleraciones angulares en el sistema de referencia inercial son entonces atraídas a partir de las del sistema de referencia del cuadricóptero con la matriz de transformación W_η^{-1} y su derivada temporal.

$$\ddot{\eta} = \frac{d}{dt}\left(W_\eta^{-1}\,v\right) = \frac{d}{dt}\left(W_\eta^{-1}\right)v = W_\eta^{-1}\,\dot{v}$$

$$= \begin{bmatrix}0 & \dot{\phi}C_\phi T_\phi + \dot{\theta}S_\phi/C_\theta^2 & \dot{\phi}S_\phi C_\theta + \dot{\theta}C_\phi/C_\theta^2\\ 0 & -\dot{\phi}S_\phi & -\dot{\phi}C_\phi\\ 0 & \dot{\phi}C_\phi/C_\theta + \dot{\phi}S_\phi T_\theta/C_\theta & \dot{\phi}S_\phi/C_\theta + \dot{\theta}C_\phi T_\theta/C_\theta\end{bmatrix}v + W_\eta^{-1}\,\dot{v} \tag{12}$$

2.3.2 Ecuaciones de Euler-Lagrange

La función de Lagrange $\mathcal{L}$ es la suma de las energías de traslación E_{trans}y rotación E_{rot} menos la energía potencial E_{pot}

$$\mathcal{L}(\mathrm{q}, \dot{\mathrm{q}}) = E_{trans} + E_{rot} - E_{pot}$$

$$= \left(\frac{m}{2}\right)\dot{\xi}^T\,\dot{\xi} + \left(\frac{1}{2}\right)v^T I v - mgz. \tag{13}$$

Tal como se muestra en [6]. Las ecuaciones de Euler-Lagrange con fuerzas externas y torques son:

$$\begin{bmatrix} f \\ \tau \end{bmatrix} = \frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\mathrm{q}}}\right) - \left(\frac{\partial \mathcal{L}}{\partial \mathrm{q}}\right) \tag{14}$$

Los componentes lineales y angulares no dependen entre sí, por lo tanto, pueden estudiarse por separado. La fuerza lineal externa es el empuje total de los rotores. La ecuación lineal de Euler-Lagrange es

$$f = RT_B = m\ddot{\xi} + mg\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \tag{15}$$

que es equivalente con la ecuación (10).

La matriz Jacobiana $\mathcal{J}\,(\eta)$ desde v hasta $\dot{\eta}$ es

$$\mathcal{J}\,(\eta) = \mathcal{J} = W_\eta^T\,\mathrm{W}_\eta$$

$$= \begin{bmatrix} I_{xx} & 0 & -I_{xx}S_\theta \\ 0 & I_{yy}C_\phi^2 + I_{zz}S_\phi^2 & (I_{yy} - I_{zz}C_\phi S_\phi C_\theta) \\ -I_{xx}S_\theta & (I_{yy} - I_{zz}C_\phi S_\phi C_\theta) & (I_{xx}C_\theta^2 + I_{yy}S_\phi^2 C_\theta^2 + I_{zz}C_\phi^2 C_\theta^2) \end{bmatrix} \tag{16}$$

Por lo tanto, la energía de rotación E_{rot} puede expresarse en el sistema de referencia como:

$$E_{rot} = \left(\frac{1}{2}\right)v^T I v = \left(\frac{1}{2}\right)\ddot{\eta}^T J \ddot{\eta}. \tag{17}$$

La fuerza angular externa es la de los pares de los rotores. Las ecuaciones angulares de Euler-Lagrange son:

$$\tau = \tau_B = J\ddot{\eta} + \frac{d}{dt}(J)\dot{\eta} - \frac{1}{2}\frac{\partial}{\partial \eta}(\dot{\eta}^T J \dot{\eta}) = J\ddot{\eta} + C(\eta, \dot{\eta})\dot{\eta} \tag{18}$$

donde la matriz $C\ (\eta,\ \dot{\eta})$ es el término de Coriolis, que contiene los términos giroscópicos y centrípetas.

De acuerdo con [7], la matriz $C\ (\eta, \dot{\eta})$ tiene la forma:

$$C(\eta,\dot{\eta}) = \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix},$$

$$\begin{aligned}
C_{11} &= 0 \\
C_{12} &= (I_{xx} - I_{zz})\left(\dot{\theta}C_{\phi}S_{\phi} + \dot{\psi}S_{\phi}^2C_{\theta}\right) + (I_{zz} - I_{yy})\dot{\psi}C_{\phi}^2C_{\theta} - I_{xx}\dot{\psi}C_{\theta} \\
C_{13} &= (I_{zz} - I_{yy})\dot{\psi}C_{\phi}S_{\phi}C_{\theta}^2 \\
C_{21} &= (I_{zz} - I_{yy})\left(\dot{\theta}C_{\phi}S_{\phi} + \dot{\psi}S_{\phi}C_{\theta}\right) + (I_{yy} - I_{zz})\dot{\psi}C_{\phi}^2C_{\theta} + I_{xx}\dot{\psi}C_{\theta} \\
C_{22} &= (I_{zz} - I_{yy})\dot{\theta}C_{\phi}S_{\phi} \\
C_{23} &= -I_{xx}\dot{\psi}S_{\theta}C_{\theta} + I_{yy}\dot{\psi}S_{\phi}^2S_{\theta}C_{\theta} + I_{zz}\dot{\psi}C_{\phi}^2S_{\theta}C_{\theta} \\
C_{31} &= (I_{yy} - I_{zz})\dot{\psi}C_{\theta}^2S_{\phi}C_{\phi} - I_{xx}\dot{\theta}C_{\theta} \\
C_{32} &= (I_{zz} - I_{yy})\left(\dot{\theta}C_{\phi}S_{\phi}S_{\theta} + \dot{\phi}S_{\phi}^2C_{\theta}\right) + (I_{yy} - I_{zz})\dot{\phi}C_{\phi}^2C_{\theta} \\
&\quad + I_{xx}\dot{\psi}S_{\theta}C_{\theta} - I_{yy}\dot{\psi}S_{\phi}^2S_{\theta}C_{\theta} - I_{zz}\dot{\psi}C_{\phi}^2S_{\theta}C_{\theta} \\
C_{33} &= (I_{yy} - I_{zz})\dot{\phi}C_{\phi}S_{\phi}C_{\theta}^2 - I_{yy}\dot{\theta}S_{\phi}^2C_{\theta}S_{\theta} - I_{zz})\dot{\theta}C_{\phi}^2C_{\theta}S_{\theta} + I_{xx}\dot{\theta}C_{\theta}S_{\theta}
\end{aligned} \tag{19}$$

La ecuación (18) conduce a las ecuaciones diferenciales para las aceleraciones angulares que son equivalentes con las ecuaciones (11) y (12).

$$\ddot{\eta} = J^{-1}(\tau_B - C(\eta,\dot{\eta})\dot{\eta}). \tag{20}$$

2.3.3 Efectos Aerodinámicos

El modelo presentado es una simplificación de interacciones dinámicas complejas. Para que el comportamiento sea más realista, la fuerza de arrastre generada por la resistencia del aire es incluida. Esto es debido a las ecuaciones (10) y (15) con la matriz de coeficientes diagonal asociando las velocidades lineales con la fuerza de desaceleración del movimiento, tal como se presenta en [8].

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = -g\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + \frac{T}{m}\begin{bmatrix} C_{\psi}S_{\theta}C_{\phi} + S_{\psi}C_{\phi} \\ S_{\psi}S_{\theta}C_{\phi} - C_{\psi}S_{\phi} \\ C_{\psi}S_{\phi} \end{bmatrix} - \frac{1}{m}\begin{bmatrix} A_x & 0 & 0 \\ 0 & A_y & 0 \\ 0 & 0 & A_z \end{bmatrix}\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} \tag{21}$$

En donde $A_x, A_y\ y\ A_z$ son los coeficientes de fuerza de arrastre para las velocidades en las direcciones correspondientes del sistema de referencia inercial.

Varios otros efectos aerodinámicos podrían incluirse en el modelo. Por ejemplo, la dependencia del empuje en el ángulo de ataque y las distribuciones del flujo de aire sido estudiado en (1) y (2). La influencia de los efectos aerodinámicos es complicada y los efectos son difíciles de modelar. Además, algunos de los efectos tienen un efecto significativo solo en altas velocidades. Por lo tanto, estos efectos se excluyen del modelo ya que solo se presentó el modelo simple.

3. Diseño y Simulación

Para definir lo que es la simulación, citaremos a Banks (2009) ... “La simulación es la imitación de la operación de un proceso o sistema del mundo real a lo largo del tiempo.” (p. 6).

Una vez realizado el correcto modelo matemático del cuadricóptero se procede a realizar la simulación para analizar su comportamiento y estudiar los efectos de cambios internos y externos del sistema como por ejemplo realizar alguna alteración en el modelo y de esta forma observando las distintas alteraciones en el comportamiento de este.

Una buena simulación nos conduce hacia a un mejor entendimiento del sistema y por consiguiente a sugerir estrategias que mejoren la operación y eficiencia del sistema.

3.1. Diseño CAD

Las partes del cuadricóptero en 3D fueron obtenidas desde el repositorio de librerías de piezas GRABCAD, para tener un punto de partida para realizar el diseño, ya que estas piezas serán modificadas para adaptarlas a nuestros requerimientos. Se utilizó el software de diseño asistido por computador, SolidWorks para realizar estas modificaciones, ya que nos ofrece un conjunto de herramientas que nos facilitan la creación de piezas además que nos ofrece la posibilidad del desarrollo de los planos, gracias al diseño 3D intuitivo podemos ver nuestro diseño en 3D en tiempo real para ver cómo vamos construyendo nuestro ensamble y poder realizar cualquier modificación o ajuste si fuese necesario, por ultimo pero nos menos importante ofrece una interfaz fácil de utilizar para la realización de piezas y ensambles.

Los elementos realizados son las siguientes:

3.1.1 Cuerpo o Cuadro

Es la estructura central que da cuerpo y soporte a cada una de las demás partes de nuestro cuadricóptero, eh aquí donde se encuentra el sistema de control, la batería y demás sensores necesarios para el funcionamiento.

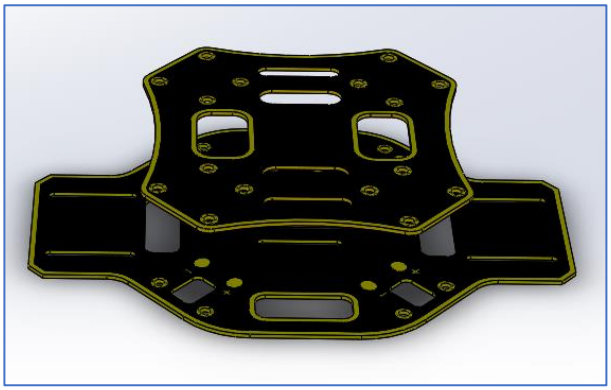


Figura 5. *Cuerpo del cuadricóptero.*

3.1.2 Brazos

Estos son los elementos en donde se ubican los motores, ESC (variadores) y las hélices, y van unidos al soporte o cuerpo del cuadricóptero.

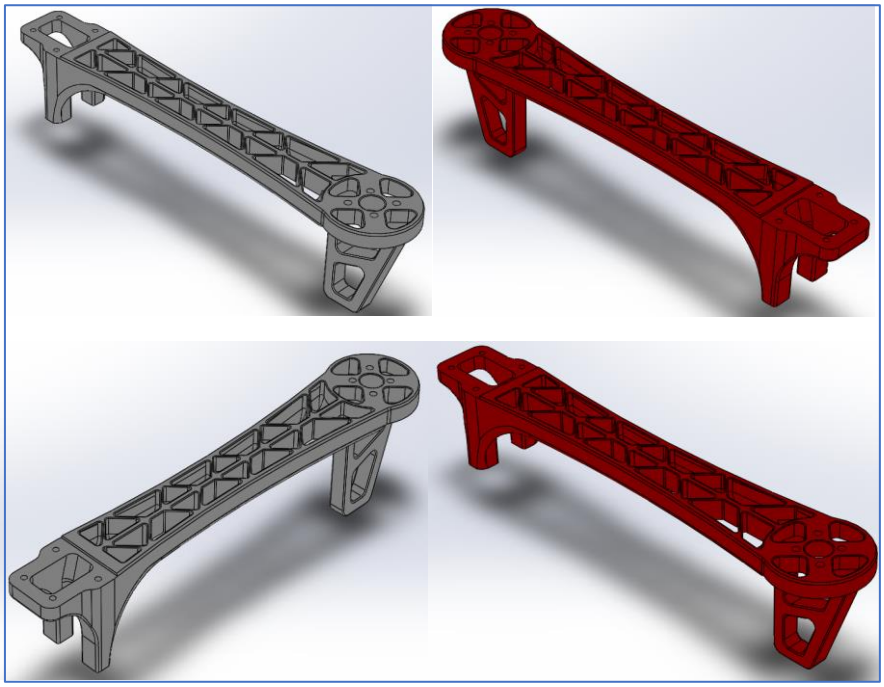


Figura 6. *Brazos del cuadricóptero.*

3.1.3 Motores

Los motores son los encargados de hacer que giren nuestras hélices para subir o bajar el cuadricóptero.

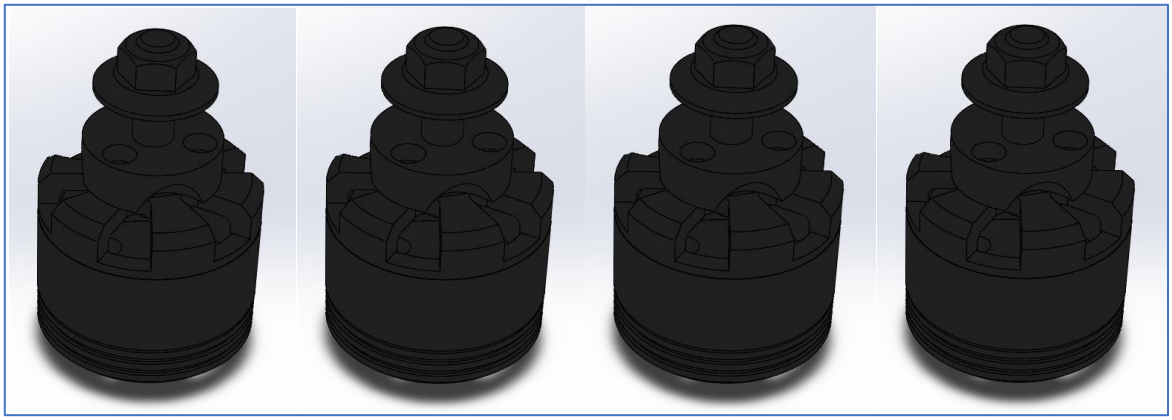


Figura 7. *Motores del cuadricóptero.*

3.1.4 Hélices

Las hélices cuentan con dos arpas que son las encargadas de elevar nuestro cuadricóptero con mayor potencia o mayor estabilidad dependiendo el control que se realiza. Al igual que los motores las hay de ambos sentidos de giro, el giro horario "CW" y antihorario "CCW".

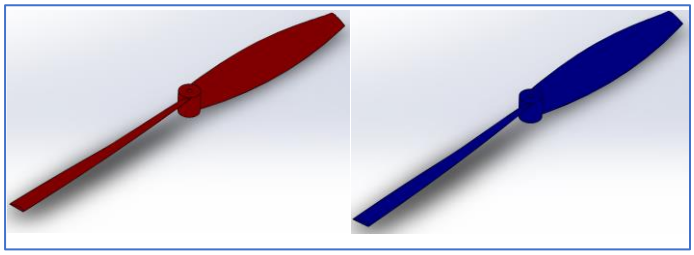


Figura 8. *Hélices con sentido de giro horario.*

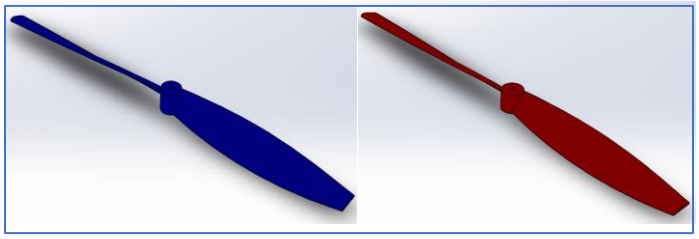


Figura 9. *Hélices con sentido de giro antihorario.*

3.1.5 Ensamble

Una vez diseñados todos los elementos necesarios para el cuadricóptero, se procede al ensamble de todas las partes, cabe destacar que más adelante se profundizará a fondo sobre los elementos del cuadricóptero y el papel que tiene cada elemento y porque es importante su elección.

Figura 10. *Ensamble de cuadricóptero.*

3.2. Modelado en Simulink

Utilizaremos la librería de "*Simscape Multibody*" que nos proporciona un entorno de simulación para sistemas mecánicos en 3D, con este paquete podemos modelar el sistema utilizando bloques que representan los elementos, las uniones, restricciones, elementos de fuerza y sensores. Con el uso de esta librería nos permite formular y resolver todas las ecuaciones de movimiento del sistema tal como fueron descritas en el capítulo anterior en el modelo matemático. Obtenido el diseño 3D de todos los elementos necesarios con SolidWorks, podemos utilizar todas las piezas diseñadas utilizando el complemento de Simscape Multibody, en donde se importan todas las piezas y luego le agregamos todas las masas, inercias, uniones, restricciones y las geometrías del modelo.

Ya una vez tengamos implementado el modelo con el paquete de Simscape Multibody lo incluiremos en Simulink y desarrollaremos el sistema de control que se encargara de controlar y regular el comportamiento del sistema.

Los resultados de la simulación lo visualizaremos a través de graficas que nos muestran la respuesta en función del tiempo de las variables de interés además de una animación 3D que es generada automáticamente para visualizar la dinámica del sistema.

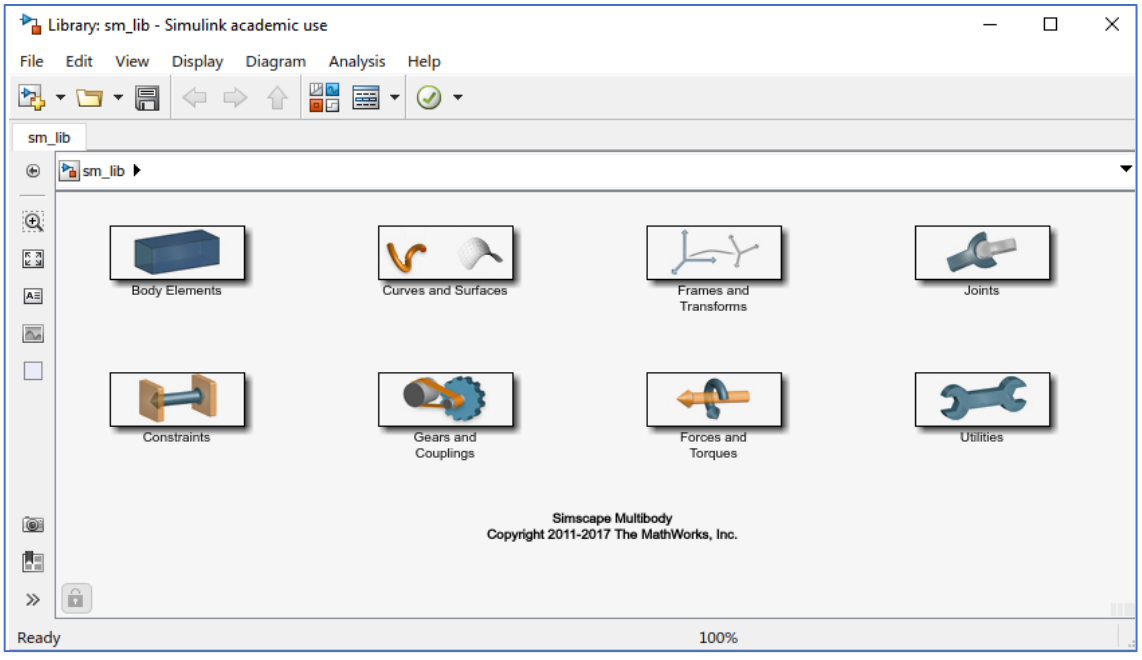


Figura 11. *Librería de Simscape Multibody.*

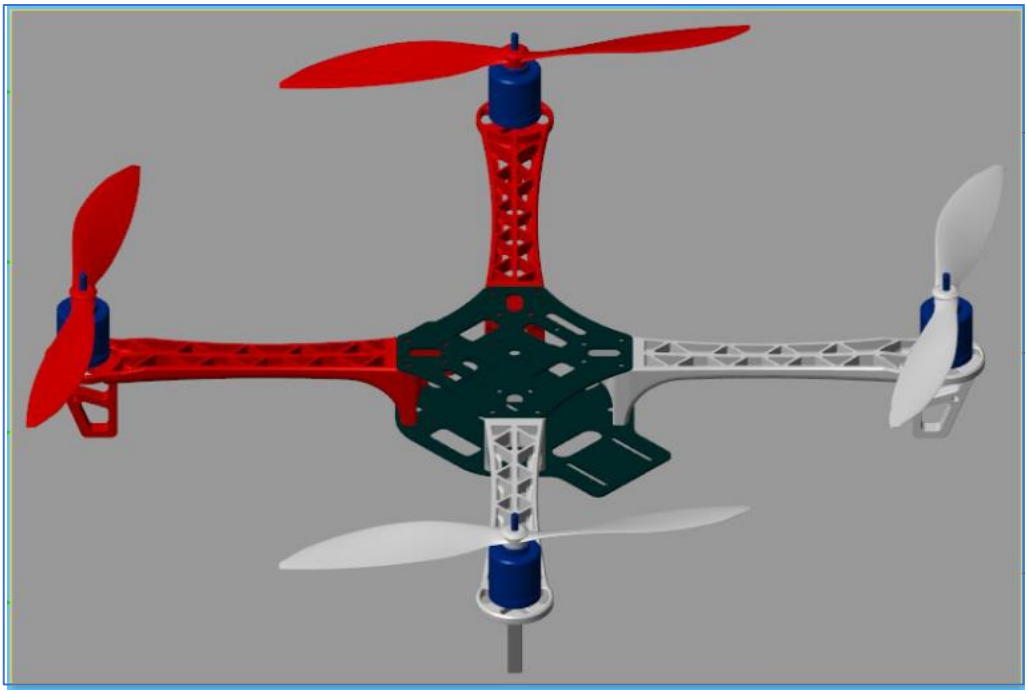


Figura 12. *Modelo 3D en Simulink.*

Una vista más completa del modelado en Simulink donde se muestran todas las piezas sólidas y las uniones entre estas y las piezas importadas además de incluir las restricciones necesarias para el correcto funcionamiento, cabe destacar que cada solido tiene sus correspondientes características como su masa, densidad, geometría, etc.

Entre las variables de interés que nos interesa analizar y ver su evolución en el tiempo, tenemos los ángulos de pitch, roll y yaw (cabeceo, alabeo y guiñada), que con estas variables realizamos el lazo cerrado de control.

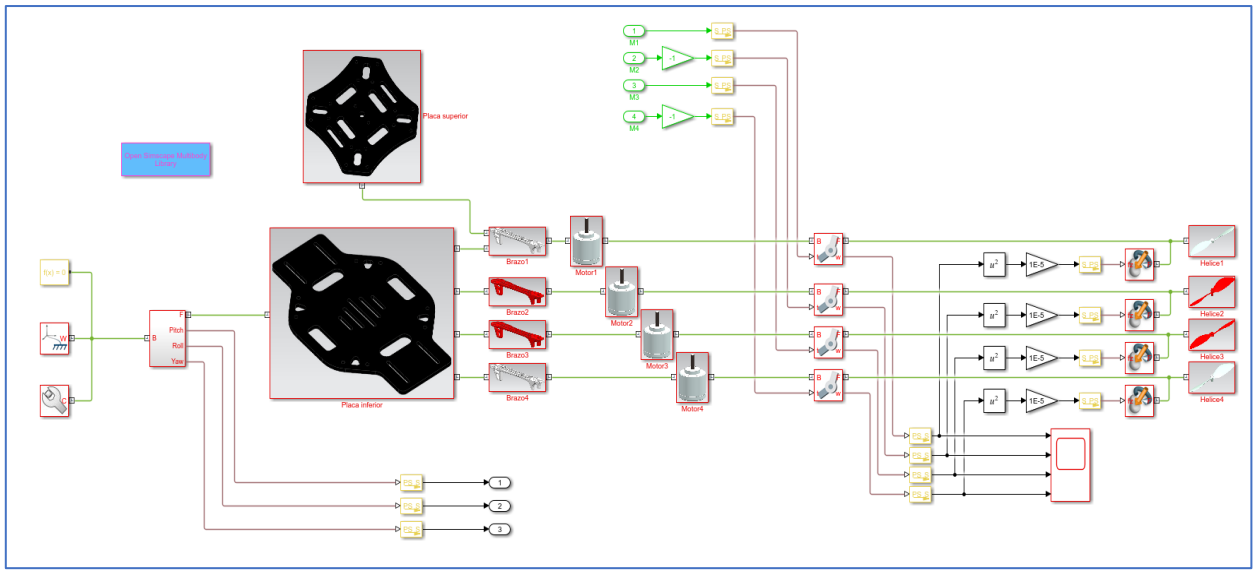


Figura 13. *Vista en detalle del modelo 3D.*

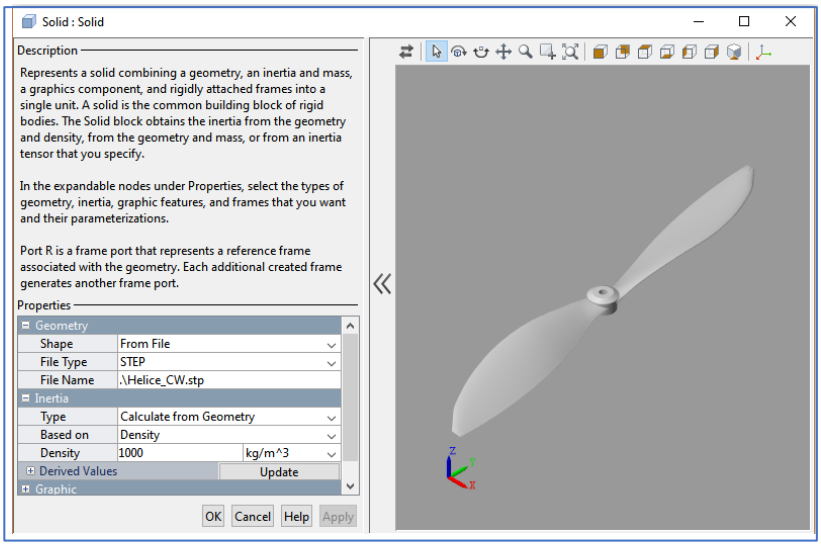


Figura 14. *Ejemplo de las características de la pieza.*

3.2.1 Sistema de control

Casi todos los dispositivos electrónicos que controlan un objeto físico usan un mecanismo de control por realimentación llamado controlador PID, que significa "Proporcional Integral Derivativo". Para explicar que es tomaremos como ejemplo un cuadricóptero que intenta mantenerse nivelado. Si una brusca ráfaga de viento lo hace rodar hacia un lado, el controlador PID mide la diferencia entre el ángulo de inclinación actual de cuadricóptero y su ángulo de balanceo de cero. En base a este error, el controlador PID determinara la mejor acción de control para enviar a los motores y poder nivelar el cuadricóptero lo más rápido posible sin sobrepasar la posición de vuelo. Otro ejemplo puede ser para nuestro caso es el de mantener el cuadricóptero en una posición especifica el control debe ser capaz de mantener esa posición sin importar las perturbaciones que le afecten.

Los controladores PID se utilizan en muchas áreas del circuito de control, incluido mantener el cuadricóptero a una altitud fija y permanecer en una posición de GPS concreta en caso de agregar un módulo de GPS, así también en los ESC y los servomotores.

Termino proporcional

El termino proporcional enviara una corrección proporcional a la cantidad de error. Para nuestro caso si el cuadricóptero está lejos de la posición definida, la acción proporcional ordenara a los motores que giren más rápido para llegar a la posición definida. Tener una ganancia demasiada baja implicará que el sistema será inestable, ya que la corrección nunca será la suficientemente fuerte para llevarlo de nuevo a la posición requerida. Aumentar el valor de P afectara en función de lo fuerte que sea la corrección, acelerando así la respuesta del cuadricóptero; no obstante, el valor es demasiado grande se sobrepasara de la posición que hayas establecido y se inclinara hacia la otra dirección.

Para evitar el rebasamiento podemos reducir el valor del término proporcional (P), pero al hacer esto el cuadricóptero tardara en volver a la posición definida; por lo tanto, introducimos el termino integral.

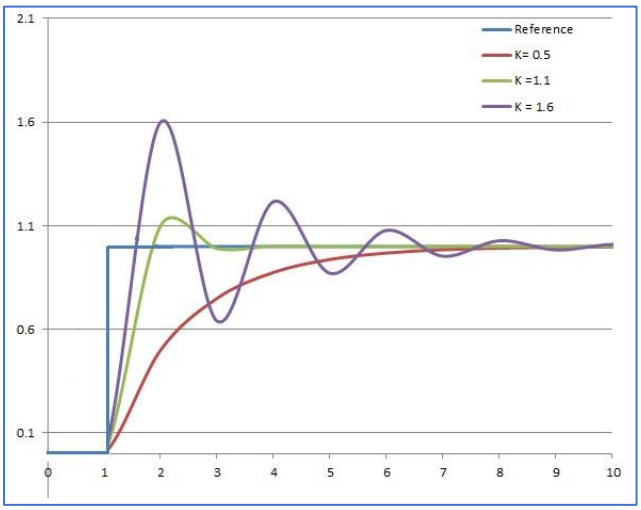


Figura 15. *Diferentes valores de la ganancia proporcional. [9]*

Termino integral

El termino integral controla esencialmente la cantidad de corrección máxima que puede reducir la cantidad de rebasamiento; esto se basa en cuanto error hay entre la posición actual y la posición deseada. El termino integral aumentara o reducirá la corrección en función de lejos que este de la referencia y de la rapidez con la que cambie.

Se puede decir que el termino integral es una forma de variar dinámicamente el termino proporcional dependiendo de la situación. En el caso del cuadricóptero alterado por una ráfaga de viento, como el error es inicialmente muy grande la ganancia integral garantizara una amplia señal de corrección; se enviará a los motores para que el cuadricóptero vuelva a nivel lo antes posible, pero cuando se aproxime a esa posición reducirá la señal de control para evitar el rebasamiento.

Termino derivativo

El termino derivativo contrala la velocidad a la que el cuadricóptero vuelve a su posición de referencia. Establecer un valor muy alto para la ganancia derivativo da como resultado movimientos más lentos y que llegue más lento hacia la posición de referencia.

Algunas de las ventajas de la acción derivativa son:

-Anticipa el error y actúa en función del error que iría a ocurrir.

-Estabiliza la respuesta de bucle cerrado.

3.2.2 Selección del controlador

Existen diversos tipos de controladores, caracterizados por la forma en que relacionan y(t) [Salida] con u(t) [Entrada] dependiendo de las variables ya sean una señal neumática, un voltaje, una corriente, entre otros.

-Los tipos básicos de controlador son:

- Controlador PROPORCIONAL ("P")
- Controlador PROPORCIONAL INTEGRAL ("PI")
- Control PROPORCIONAL DERIVATIVO (“PD”)
- Controlador PROPORCIONAL INTEGRAL DERIVATIVO ("PID").

A continuación, veremos la definición y las características de cada controlador para de esta forma elegir el cumpla con nuestras especificaciones de control y conseguir un buen control de nuestro sistema.

Control Proporcional (P)

En estos controladores la señal de accionamiento es proporcional a la señal de error del sistema.

Es el más sencillo de los distintos tipos de control y consiste en amplificar la señal de error antes de aplicarla al proceso.

La función de transferencia de este tipo de reguladores es una variable real, denominada Kp (constante de proporcionalidad) que determinará el grado de amplificación del elemento de control.

Si y(t) es la señal de salida (salida del controlador) y e(t) la señal de error (entrada al controlador), en un sistema de control proporcional tendremos:

$$y(t) = Kp \cdot e(t)$$

En el dominio de Laplace, será: $$Y(s) = Kp \cdot E(s)$$

Por lo que la función de transferencia será: $$G(s) = \frac{Y(s)}{E(s)} = Kp$$

Donde Y(s) es la salida del regulador o controlador, E(s) la señal de error y K_p la ganancia del bloque de control.

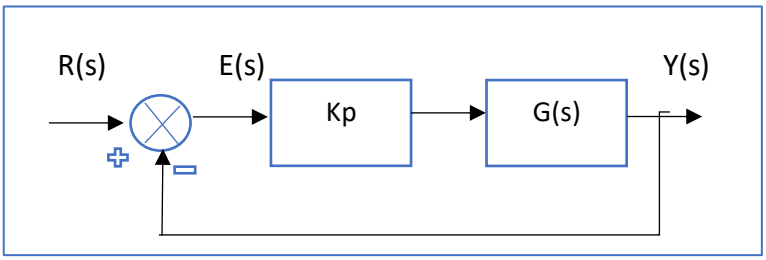


Figura 16. *Controlador P.*

La respuesta, en teoría es instantánea, con lo cual el tiempo no intervendría en el control. En la práctica, no ocurre esto, si la variación de la señal de entrada es muy rápida, el controlador no puede seguir dicha variación y presentará una trayectoria exponencial hasta alcanzar la salida deseada.

Control Proporcional Integral (PI)

En estos reguladores el valor de la acción de control es proporcional a la integral de la señal de error, por lo que en este tipo de control la acción varía en función de la desviación de la salida y del tiempo en el que se mantiene esta desviación.

La salida de este regulador es: $$y(t) = Kp \cdot e(t) + \frac{Kp}{Ti}\int e(t)\,dt$$

En el dominio de Laplace, será: $$Y(t) = Kp \cdot E(s) + \frac{Kp}{Ti(s)} \cdot E(s)$$

Por lo que la función de transferencia será: $$Y(t) = \frac{Y(s)}{E(s)} = Kp \cdot (1 + \frac{1}{Ti \cdot s})$$

Donde K_p y T_i son parámetros que se pueden modificar según las necesidades del sistema. Si T_i es grande la pendiente de la rampa, correspondiente al efecto integral será pequeña y, su efecto será atenuado, y viceversa.

Por lo tanto, la respuesta de un regulador PI será la suma de las respuestas debidas a un control proporcional P, que será instantánea a detección de la señal de error, y con un cierto retardo entrará en acción el control integral I, que será el encargado de anular totalmente la señal de error.

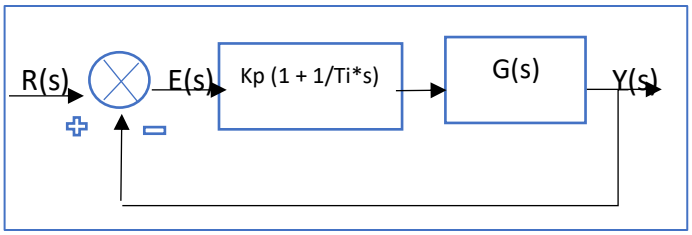


Figura 17. *Control Proporcional Integral PI.*

Control Proporcional Derivativo (PD)

El controlador derivativo se opone a desviaciones de la señal de entrada, con una respuesta que es proporcional a la rapidez con que se producen éstas

La salida de este regulador es: $$y(t) = Kp \cdot e(t) + Kp \cdot Td \cdot \frac{de(t)}{dt}$$

En el dominio de Laplace, será: $$Y(t) = Kp \cdot E(s) + Kp \cdot Td \cdot s \cdot E(s)$$

Por lo que la función de transferencia será: $$Y(t) = \frac{Y(s)}{E(s)} = Kp \cdot (1 + Td \cdot s)$$

K_P y T_d son parámetros ajustables del sistema. A T_d es llamado tiempo derivativo y es una medida de la rapidez con que un controlador PD compensa un cambio en la variable regulada, comparado con un controlador P puro.

Este tipo de controlador actúa cuando existen cambios sobre la variable a controlar, y es proporcional a la derivada del error, reduce el sobre impulso y el tiempo de estabilización, por lo que tendrá efecto de incrementar la estabilidad del sistema mejorando la respuesta del sistema, la acción derivativa ocurre donde la magnitud de la salida del controlados es proporcional a la velocidad de cambio de la señal de error.

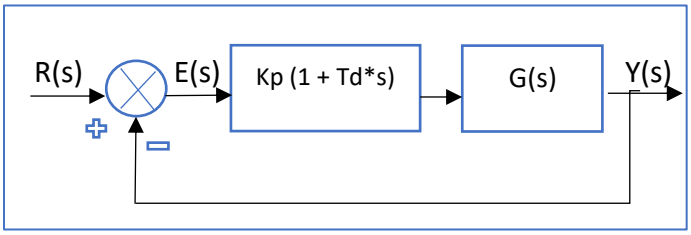


Figura 18. Controlador Proporcional Derivativo.

Control Proporcional Integral Derivativo (PID)

Es un sistema de regulación que trata de aprovechar las ventajas de cada uno de los controladores de acciones básicas, de manera que, si la señal de error varía lentamente en el tiempo, predomina la acción proporcional e integral y, mientras que, si la señal de error varía rápidamente, predomina la acción derivativa. Tiene la ventaja de ofrecer una respuesta muy rápida y una compensación de la señal de error inmediata en el caso de perturbaciones. Presenta el inconveniente de que este sistema es muy propenso a oscilar y los ajustes de los parámetros son mucho más difíciles de realizar.

La salida de este regulador es: $y(t) = Kp \cdot e(t) + \frac{Kp}{Ti}\int e(t)\,dt + Kp \cdot Td \cdot \frac{de(t)}{dt}$

En el dominio de Laplace, será: $Y(t) = Kp \cdot E(s) + \frac{Kp}{Ti(s)} \cdot E(s) + Kp \cdot Td \cdot s \cdot E(s)$

Por lo que la función de transferencia será: $Y(t) = \frac{Y(s)}{E(s)} = Kp \cdot (1 + \frac{1}{Ti \cdot s} + Td \cdot s)$

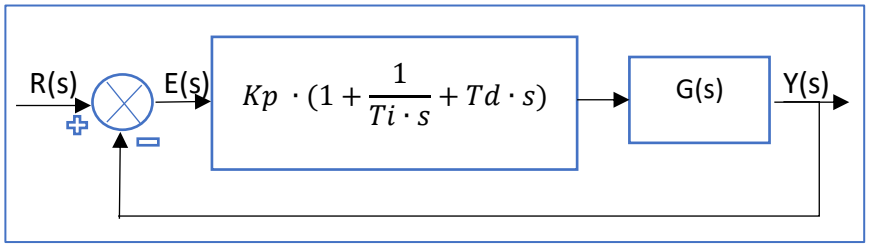


Figura 19. *Controlador Proporcional Integral Derivativo (PID)*

Luego de haber descritos los distintos tipos de controladores y ver sus características elegimos el **Control Proporcional Derivativo (PD),** ya que este tipo de control se anticipa al error y actúa en función al que va a ocurrir, es una ventaja que podemos aprovechar ya que necesitamos asegurar que el cuadricóptero se mantenga en un posición en la posición requerida y en caso de que exista alguna perturbación como una ráfaga de viento, la acción derivativa anticipara este cambio y reaccionara rápidamente para asegurar la posición definida.

La sintonización de los controladores Proporcional Integral Derivativo (PID), consiste en la determinación del ajuste de los parámetros (K_p, T_i, T_d), para lograr el comportamiento del sistema de control aceptable y robusto de conformidad con algún criterio de desempeño establecido. Para poder realizar la sintonización de los controladores, primero se debe identificar la dinámica del proceso, y a partir de ésta determinar los parámetros del controlador utilizando el método de sintonización seleccionado.

Para nuestro **controlador Proporcional Derivativo (PD),** necesitaremos determinar los parámetros de ajuste (K_p *y* T_d).

1.1.1. Selección del método de sintonización

Existen varios métodos para sintonizar controladores como por ejemplo los propuestos por Ziegler y Nichols [10], Cohen y Coon [10], López et al [11], Kaya y Sheib [11], y Sung et al [12], estos métodos suelen ser lo más empleados para sintonizar los controladores.

Utilizaremos el método de Ziegler y Nichols, que es uno de los métodos de sintonización más ampliamente difundido y utilizado. Los valores propuestos por este método intentan conseguir en el sistema realimentado una respuesta al escalón con un sobre impulso máximo del 25%, que es un valor robusto con buenas características de rapidez y estabilidad para la mayoría de los sistemas, además de que nos permite sintonizar el regulador PID de forma empírica, sin necesidad de conocer las ecuaciones de la planta o el sistema controlado, lo que facilita el desarrollo del controlador que se va a utilizar.

3.2.3 Sintonización del controlador

Método de Ziegler y Nichols

El primer procedimiento sistematizado para el cálculo de los parámetros de un controlador PID fue desarrollado por Ziegler y Nichols [13], en donde se explica el método de ajuste de manera experimental de un controlador PID.

De acuerdo con los autores, utilizando un controlador puramente proporcional y mediante un proceso iterativo, el procedimiento requiere aumentar paulatinamente la ganancia proporcional "K_p" del controlador hasta lograr que el sistema entre en una oscilación sostenida ante un cambio del escalón en el valor deseado. La ganancia en este punto es la *ganancia última "K_u"* y el periodo de la oscilación, el *periodo último* "T_u".

Para el ajuste proporcional seleccionaron, el criterio de desempeño que seleccionaron fue el de un decaimiento de $\frac{1}{4}$, es decir que el error decae en la cuarta parte de un periodo de oscilación. Encontraron que la ganancia proporcional para un controlador P debería ser la mitad de la ganancia última.

Las ecuaciones de sintonización del controlador PID son:

$$Kp = 0{,}6 \cdot Ku \; hasta \; 1{,}0 \cdot Ku$$

$$Ti = 0{,}5 \cdot Tu$$

$$Td = 0{,}125 \cdot Tu$$

Estas son las ecuaciones que nos serán necesarias para sintonizar nuestro controlador como se mencionó antes, solamente nos hará falta encontrar a través de las pruebas descritas **Método de Ziegler y Nichols** la K_u y la T_u para de esta forma poder calcular la K_p y el T_d ya que nuestro controlador es PD.

Por lo que al final tendremos estas dos ecuaciones:

$$Kp = 0{,}6 \cdot Ku$$

$$Td = 0{,}125 \cdot Tu$$

3.2.4 Desarrollo del sistema de control

Luego de haber seleccionado el correcto controlador y utilizar el método adecuado para sintonizar el controlador para el cuadricóptero, pasamos a implementar el controlador en Simulink, a continuación, se observan los distintos controladores diseñados para para el control de los ángulos de roll, pitch y yaw (alabeo, cabeceo y guiñada).

Control para el *roll* (alabeo)

Para el control del ángulo de alabeo ϕ, que corresponde al movimiento del cuadricóptero con respecto al eje x. Utilizando el método descrito en el aparto anterior (*Ziegler y Nichols*), determinamos los valores de K_p y T_d, para el caso de la ganancia derivativa, $Kd = \frac{1}{Td}$.

Por lo que tenemos en el controlador PD: $Kp = 0.05$ y $Kd = 0.008$

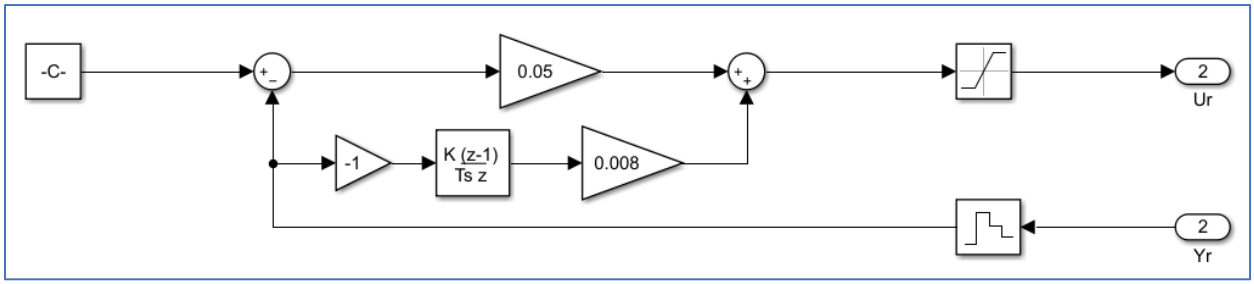


***Figura 20**.Controlador PD para el ángulo de roll (alabeo).*

Control para el *pitch* (cabeceo)

En el caso del ángulo de cabeceo θ, el control es muy similar al del roll. El ángulo de cabeceo corresponde al movimiento del cuadricóptero con respecto al eje y. Utilizando el método descrito en el aparto anterior (*Ziegler y Nichols*), determinamos los valores de K_p y T_d, para el caso de la ganancia derivativa, $Kd = \frac{1}{Td}$.

Por lo que tenemos en el controlador PD: $Kp = 0.05$ y $Kd = 0.008$

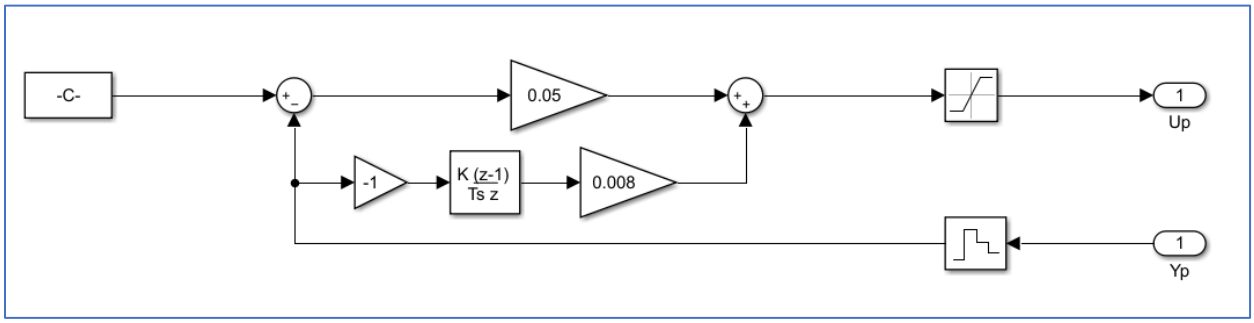


Figura 21. *Controlador PD para el ángulo de Pitch (cabeceo).*

Control para el *yaw* (Guiñada)

El ángulo de guiñada ϕ, corresponde al movimiento con respecto al eje z. Para este controlador solo usaremos la acción proporcional, por lo que anularemos la acción derivativa, solo tendremos un control proporcional P ya que será suficiente para el correcto funcionamiento. Nuevamente apoyándonos en el método de *Ziegler y Nichols,* determinamos el valor de K_p.

Por lo que tenemos en el controlador P: $Kp = 0.1$

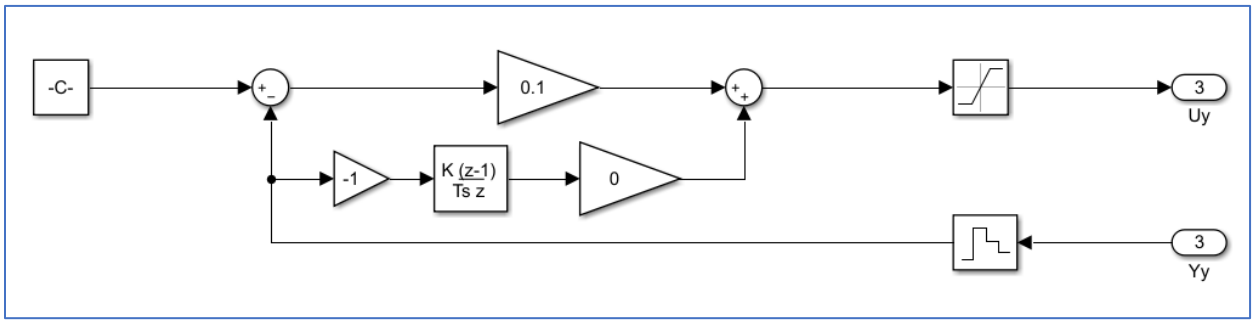


Figura 22. *Controlador P para el ángulo de yaw (guiñada).*

Luego de haber desarrollado los distintos controladores necesarios para el control del cuadricóptero pasamos a incluirlo en el modelo 3D y así obtenemos el conjunto completo.

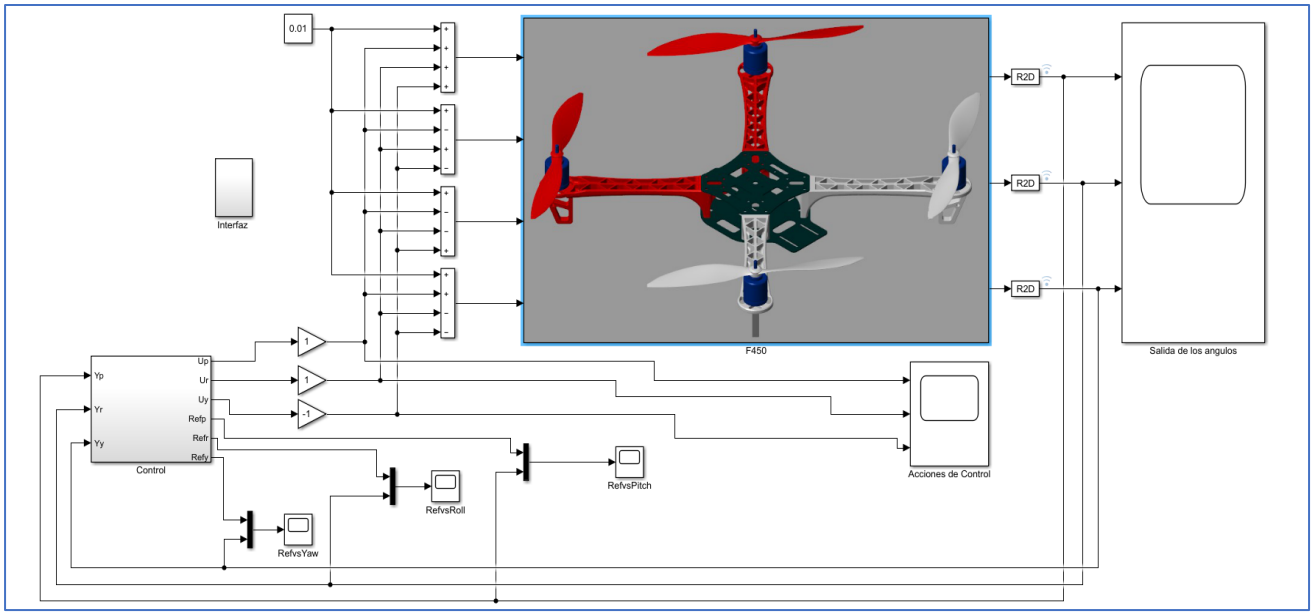


Figura 23. *Modelo en Simulink con los controladores aplicados.*

3.4. Referencia de escalón de 15°

La primera prueba es la de ensayar cada controlador por independiente y observar como el sistema se estabiliza y sigue la referencia, esta prueba es simulando una entrada de escalón a un valor de 15°

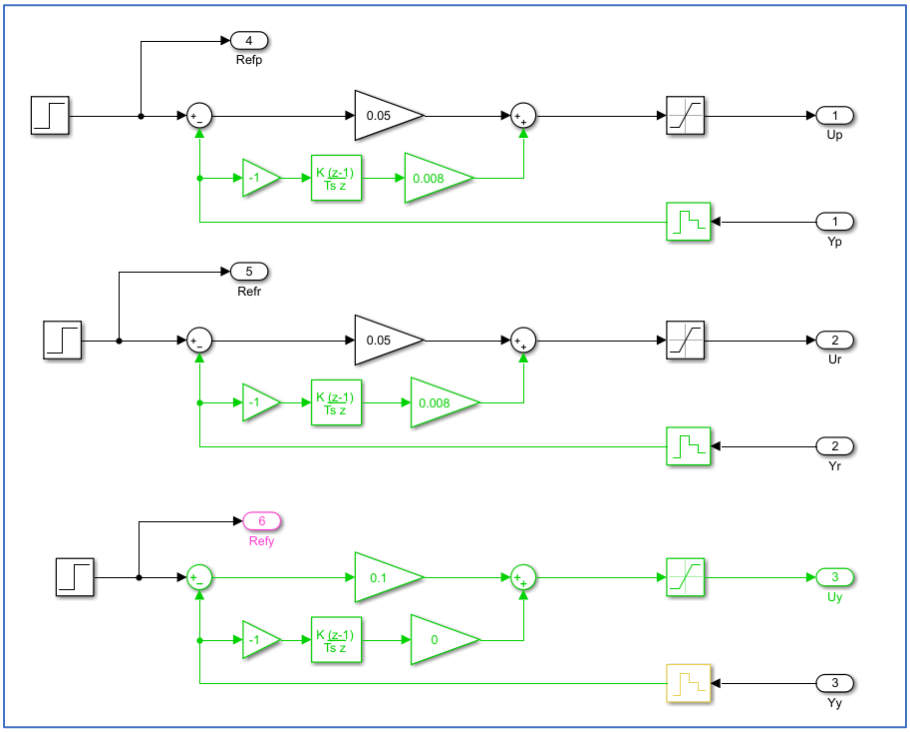


Figura 24. *Prueba referencia de escalón de 15°.*

Resultados

Ángulo *de alabeo* ϕ

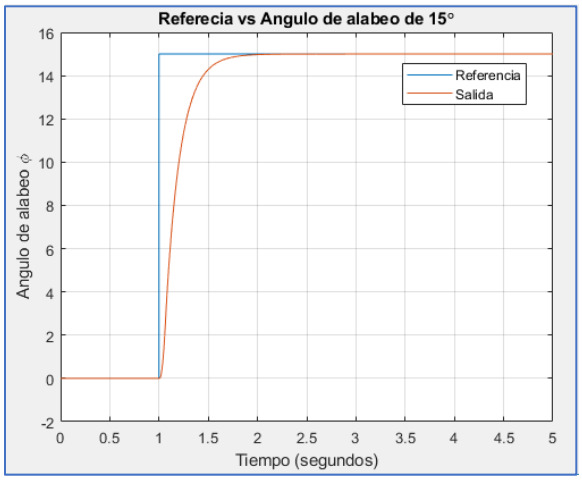


Figura 25. *Referencia vs ángulo de alabeo.*

Ángulo *de cabeceo* θ

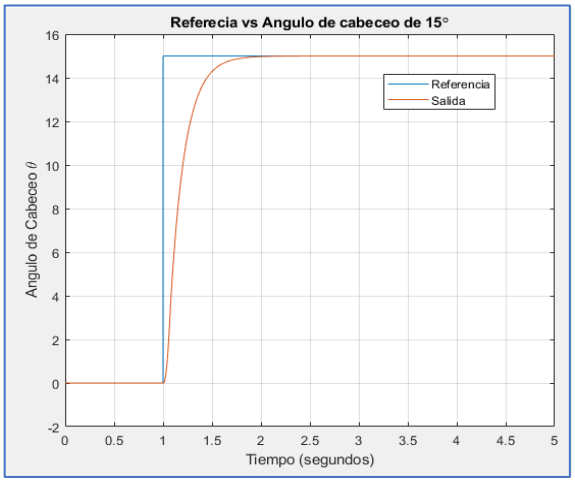


Figura 26. *Referencia vs ángulo de cabeceo.*

Ángulo *de* guiñada ϕ

Figura 27. *Referencia vs ángulo de guiñada.*

En la siguiente figura, notamos como la entrada escalón de 15° aplicada al ángulo de cabeceo se mantiene mientras que los demás ángulos están puesto a la posición de reposo (0°), y vemos como el sistema sigue la referencia de los 3 ángulos, esto quiere decir que los controladores están funcionando correctamente.

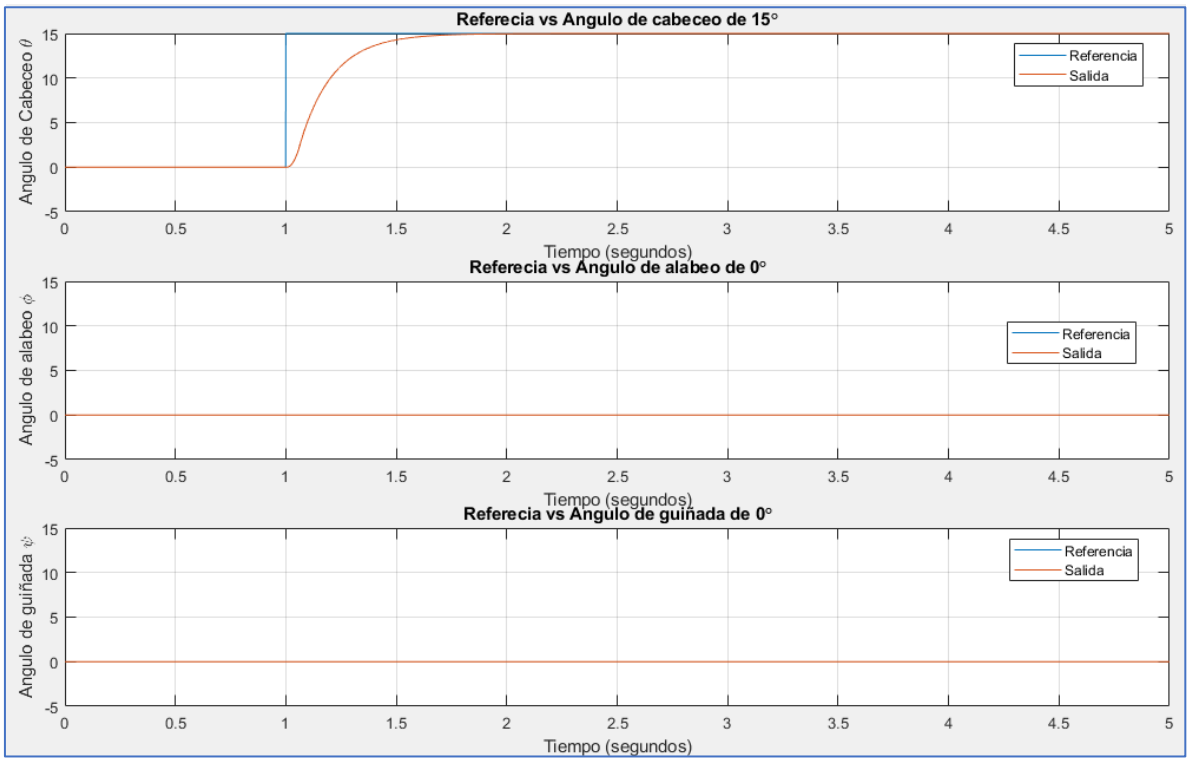


Figura 28. *Seguimiento de la referencia de escalón 15°.*

3.5. Referencia senoidal de -20° a 20°

Esta prueba consiste en aplicarle una entrada senoidal a cada controlador, con valores que irán variando desde -20° a 20° y observar el seguimiento del sistema a la referencia.

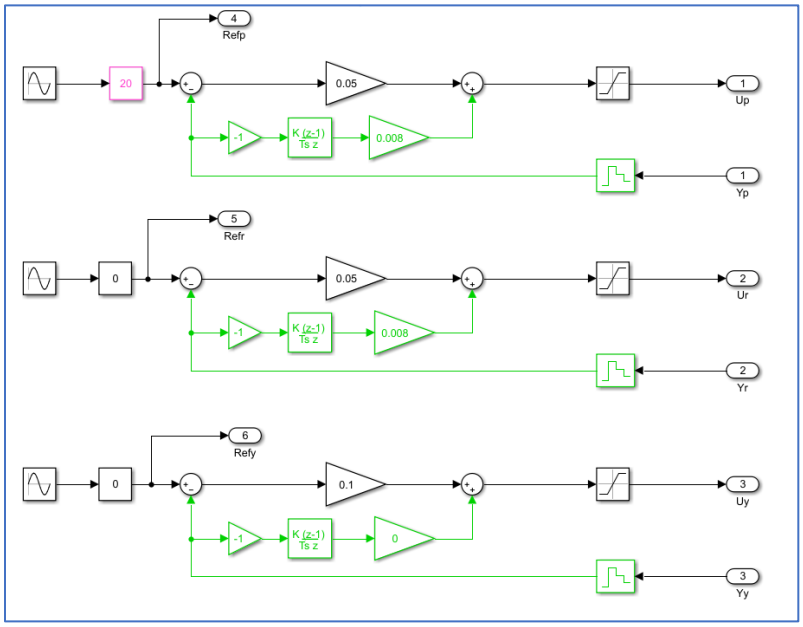


Figura 29. *Prueba referencia senoidal -20° a 20°.*

Resultados

Ángulo de cabeceo θ

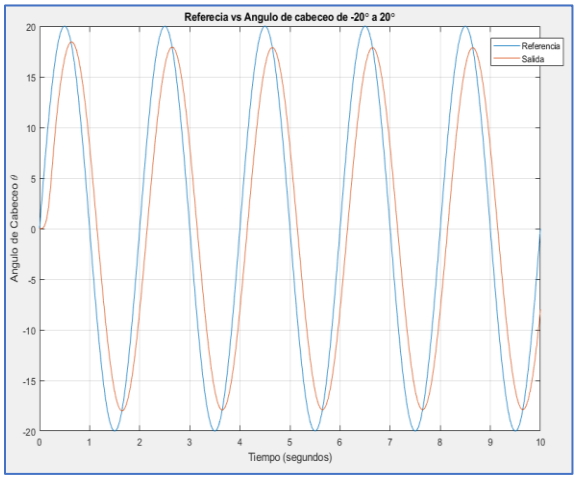


Figura 31. *Referencia vs ángulo de cabeceo.*

Ángulo de alabeo ϕ

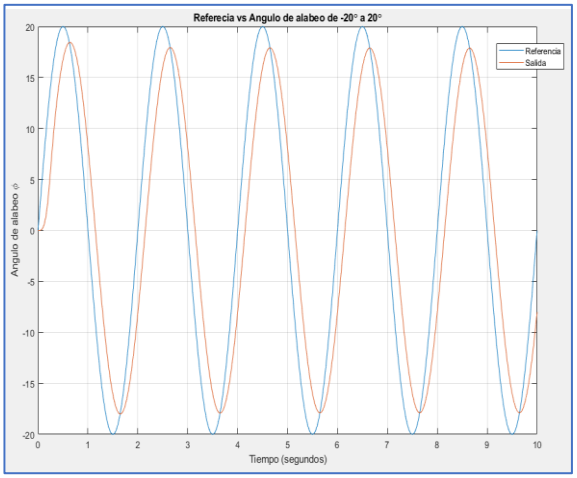


Figura 30*. Referencia vs ángulo de alabeo.*

Ángulo de guiñada ϕ

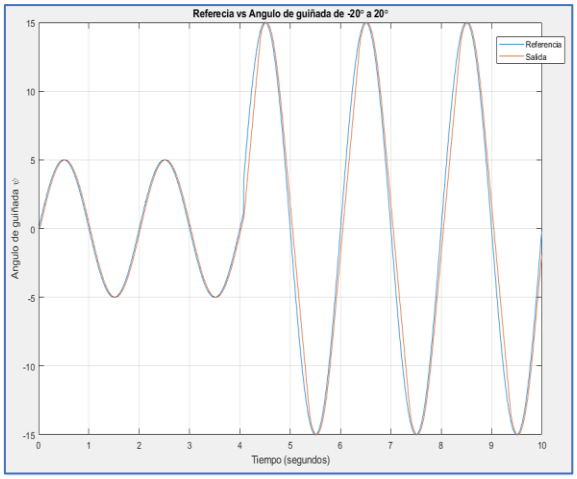


Figura 32. *Referencia vs ángulo de guiñada.*

Observando las figuras anteriores podemos observar como el controlador está actuando y siguiendo la referencia de manera correcta, podemos destacar un pequeño retraso entre la referencia y el seguimiento, pero esto no supone un gran problema ya que la diferencia es de 1 a 2°. Al igual que la prueba anterior, veremos en la siguiente figura la entrada senoidal de -20° a 20° aplicada esta vez al ángulo de alabeo mientras que los demás ángulos están puesto a la posición de reposo (0°), y vemos como como el sistema es capaz de seguir la referencia de los 3 ángulos, esto quiere decir que los controladores están funcionando correctamente.

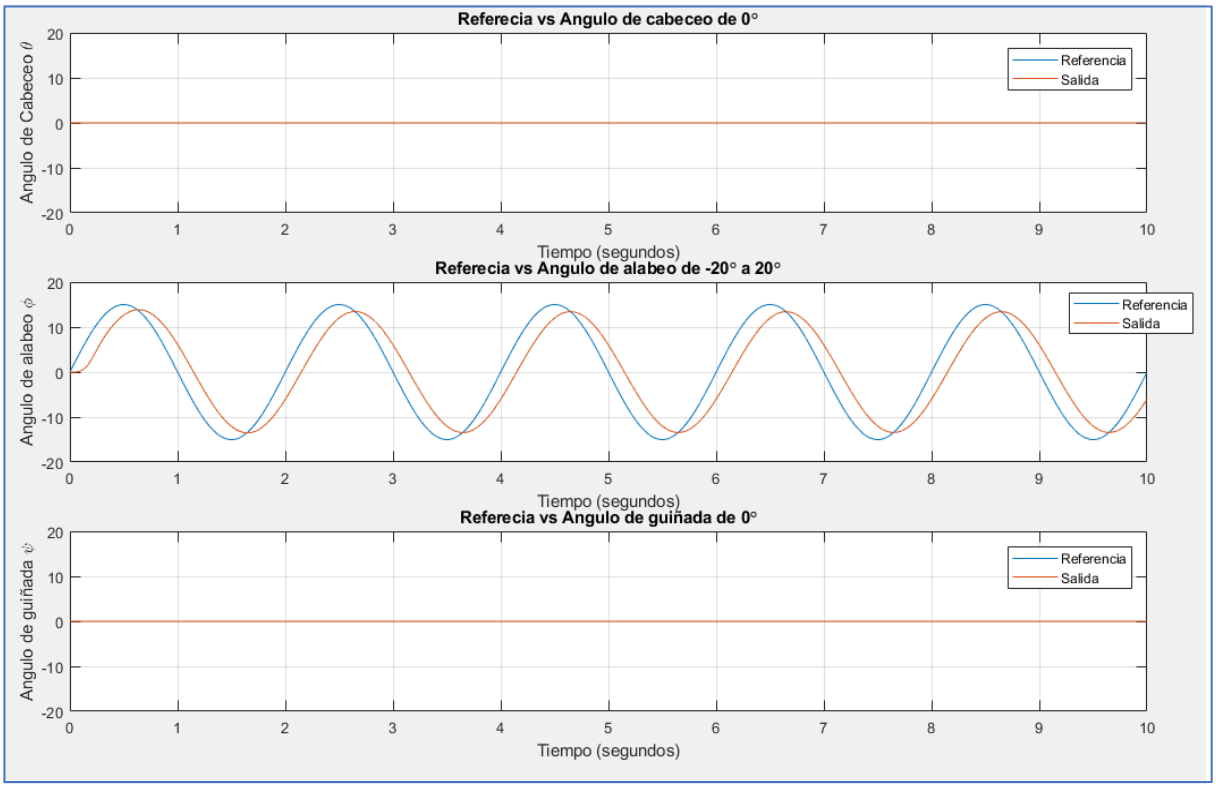


Figura 33. *Seguimiento de la referencia senoidal de -20° a 20°.*

3.6. Referencia de escalón variable

En esta última prueba sometemos al cuadricóptero a una entrada de escalón variable a los tres ángulos y ver el comportamiento del sistema y observar como el control es capaz de seguir la referencia de los ángulos que le indiquemos.

Resultados

Ángulo de cabeceo θ

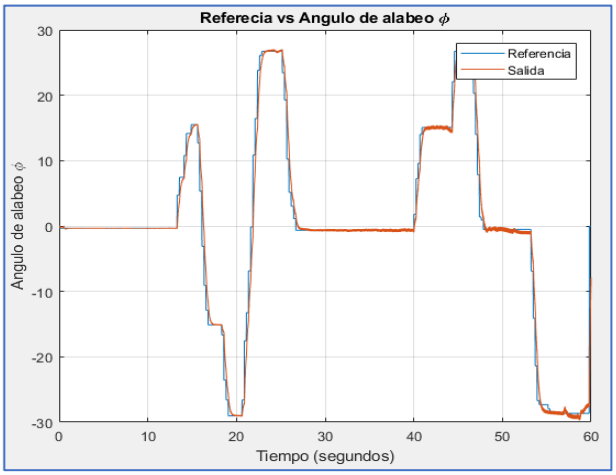


Figura 34. *Referencia de escalón variable vs ángulo de cabeceo.*

Ángulo de alabeo ϕ

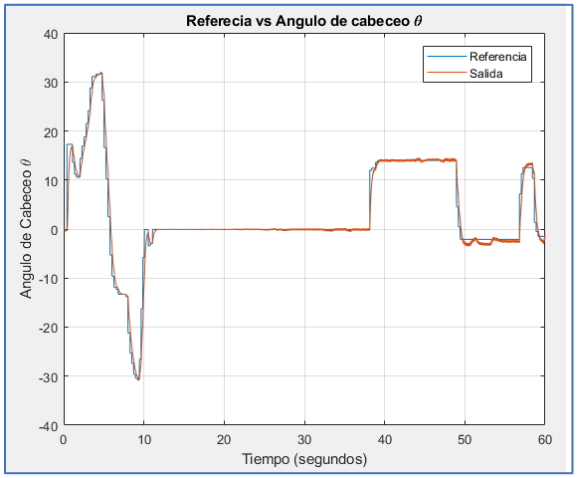


Figura 36. *Referencia vs ángulo de alabeo.*

Ángulo *de* guiñada ϕ

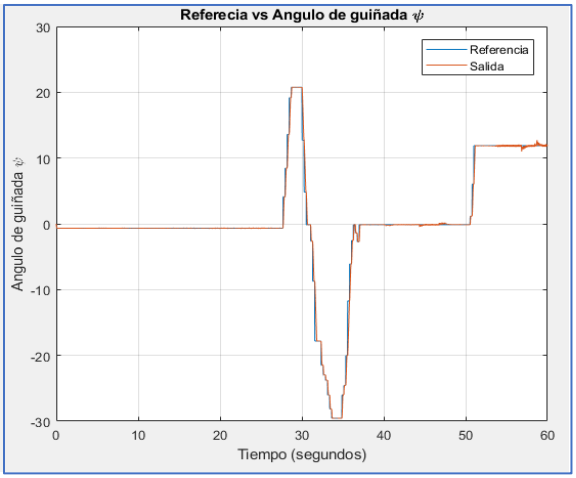


Figura 35. *Referencia vs ángulo de guiñada.*

Movimientos de los ángulos

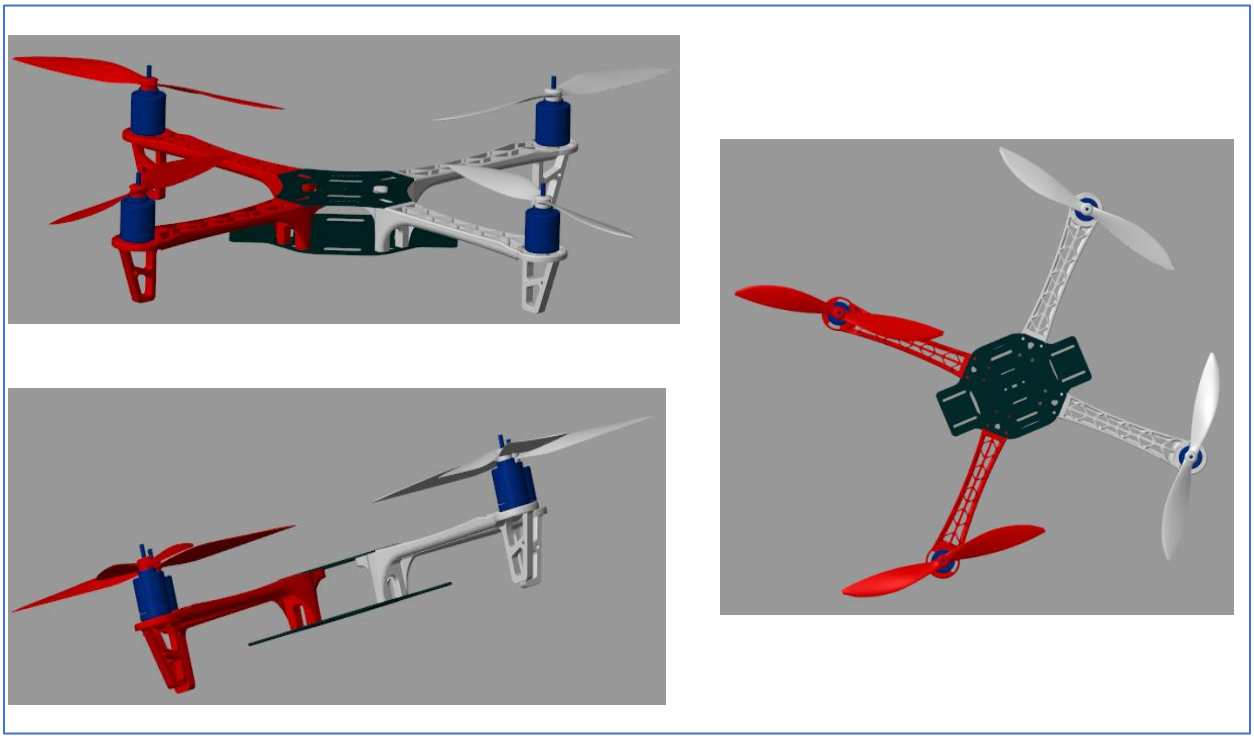


Figura 37. *Movimiento de los ángulos de cabeceo, alabeo y guiñada.*

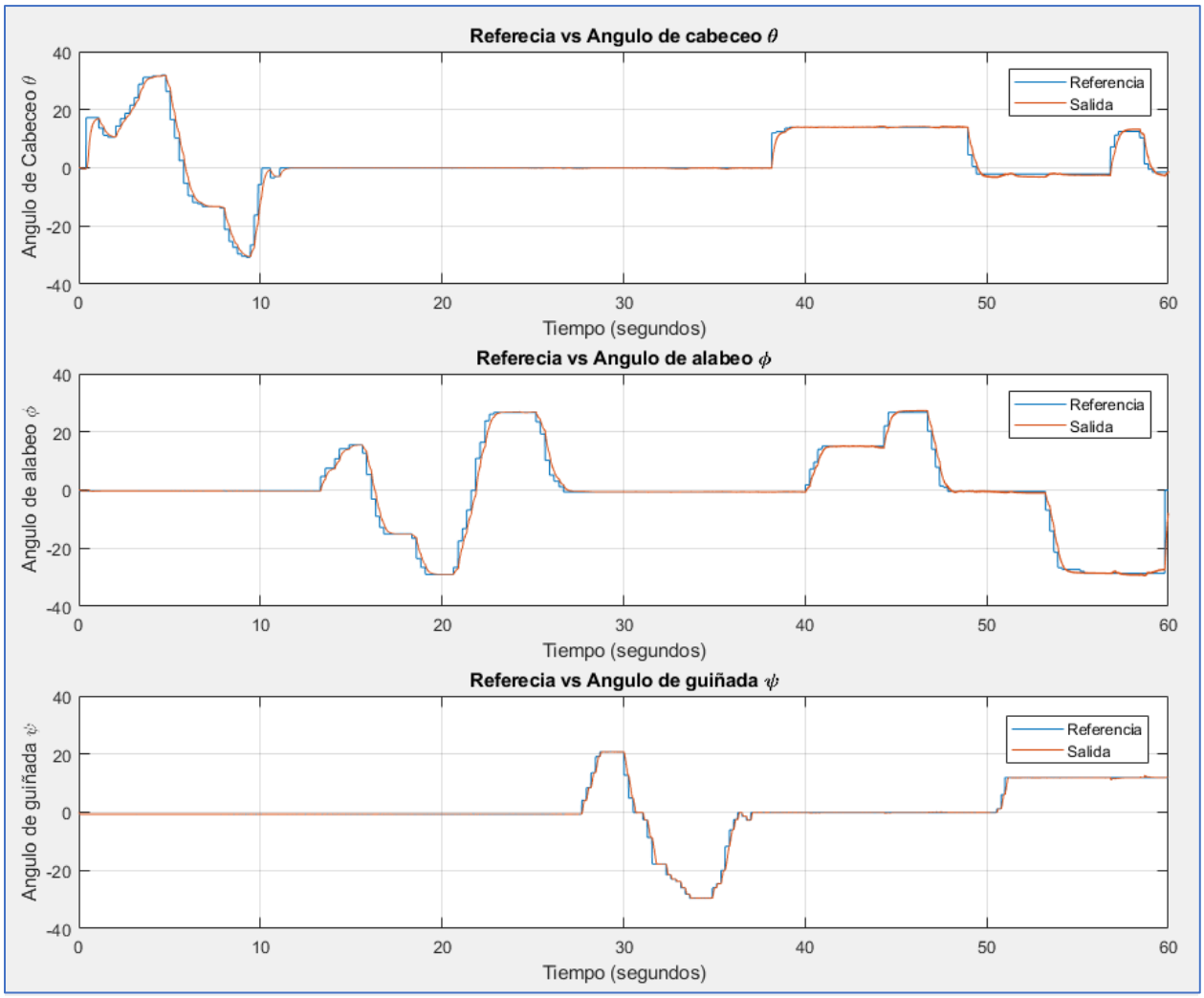


Figura 38. *Seguimiento de la referencia variable.*

Vista la figura anterior notamos un buen seguimiento de la referencia por parte del controlador aplicado, notamos como a pesar de aplicar varios escalones simultáneos en los tres ángulos, el control se mantiene, sigue la referencia y mantiene el sistema estable.

4. Implementación Real

Una vez realizado el modelado y simulación del cuadricóptero, se procede a implementar el sistema de control diseñado en un cuadricóptero real. Para esta etapa se seleccionan los componentes necesarios para el correcto funcionamiento.

4.1. Selección de componentes

4.1.1 Estructura

Es la parte donde se ensamblan y apoyan el resto de los componentes Puede tener diferentes tamaños y diseños, pero su principal objetivo o función será la de reducir al máximo las vibraciones producidas por los motores. Por otro lado, es importante que el material empleado en la fabricación de la estructura tenga las propiedades de ser fuerte, rígido y ligero, para nuestro caso el material elegido que cumple con esas propiedades es la fibra de carbono.

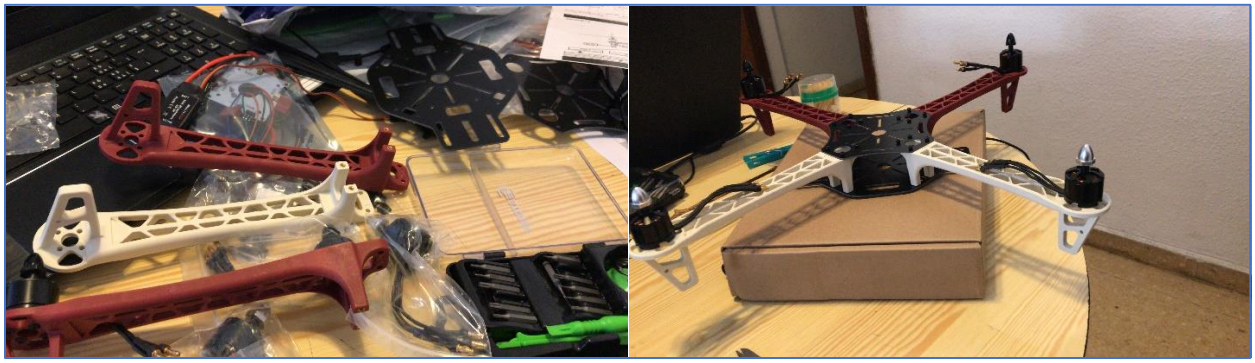


***Figura 39**. Cuerpo y brazos del cuadricóptero.*

1.1.1. Hélices

Habrá cuatro en total, una por cada motor. Las hélices son el elemento que va a permitir volar a él cuadricóptero. A través de la fuerza que les transmiten los motores, generan una fuerza opuesta a la gravedad y superior a la que ésta ejerce sobre el cuadricóptero. Esta fuerza opuesta al peso es conocida como fuerza de sustentación.

4.1.2 Motores

El cuadricóptero tiene cuatro motores y en cada uno de ellos se conecta una hélice. Se sitúan en la parte exterior de los brazos de la estructura, son los responsables de producir el movimiento del cuadricóptero.

Características:

Motor Brushless motor 2212 (900KV).

Máxima eficiencia: El 80 %.

Corriente de eficiencia máxima: 4-10A (> el 75 %).

Capacidad corriente: 12A/60s.

Ninguna corriente de Carga 10V: 0.5A.

Dimensiones de motor: 27.5mm x 30mm.

Diámetro de Eje: 3.17mm

Peso: 63g.

Un motor Brushless o motor sin escobilla, es el tipo de motor utilizado por los multirrotores para mover sus hélices. Su principal ventaja frente a los motores eléctricos de escobillas es que no se produce tanto desgaste en su funcionamiento como en estos últimos. Además, que este tipo de motor son ideales para los cuadricópteros por el hecho de ofrecer gran potencia, ofreciendo un bajo peso lo que haría más fácil el vuelo del cuadricóptero.

4.1.3 Control electrónico de velocidad (ESC)

Por sus siglas (Electronic Speed Control) nos encontramos ante un controlador de velocidad electrónico. El propósito del variador es variar la velocidad de un motor eléctrico junto con el sentido de giro. Es un componente esencial en el cuadricóptero, ya que interpreta la información que recibe de la placa controladora y varía la velocidad y dirección en el movimiento del cuadricóptero.

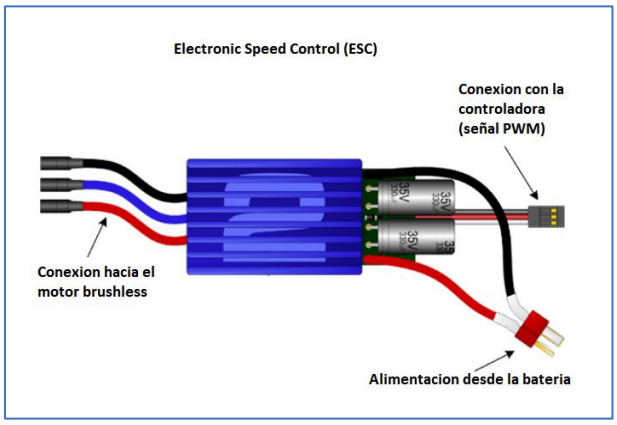


Figura 40. *Partes de un ESC.*

Lo recomendable a la hora de elegir nuestro variador es optar por uno que se encuentre por encima de la demanda del motor del cuadricóptero.

Características:

- Voltaje de entrada: 2-6S.
- Con BEC. (Battery Eliminator Circuit)
- Peso: 39g.
- Dimensiones: 55x27x7mm.

4.1.4 Batería

Alimenta de energía todos los componentes eléctricos. La batería recomendada basado en nuestros componentes electrónicos es una 2200 11.1v, tipo LiPO.

4.1.5 Tarjeta de distribución eléctrica

Permite el conexionado de todos los variadores con la batería y de todos los conectores que van al receptor de cada variador y el cable de señal que va a la placa controladora.

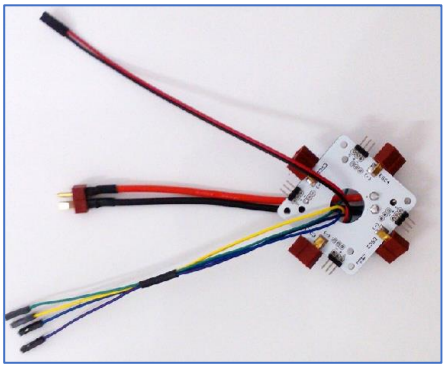


Figura 41. *Tarjeta de distribución eléctrica.*

4.1.6 Placa controladora o circuito de control

Es la tarjeta madre del cuadricóptero; recoge datos de todo el sistema, velocidades de los motores, de los giróscopos y acelerómetros, los procesa y da las órdenes oportunas para mantener la estabilidad del sistema, además de transmitir convenientemente a cada motor las órdenes para el control.

Existen diversas placas controladoras que llevan incluido el sistema de control, incluso incorporan un autopiloto que se encarga de programar los vuelos, sin necesidad de intervención del usuario, pero para nuestro caso, diseñaremos el sistema de control desde cero, por lo que necesitaremos una placa de desarrollo que cumpla con nuestros requerimientos.

La tarjeta de control tiene que sea capaz de realizar todos los cálculos requeridos por el sistema de control diseñado para estabilizar el sistema lo más rápido posible sin pérdida de información ni cuelgue, además de que ser de bajo consumo para que el cuadricóptero posea buena autonomía.

Arduino Due

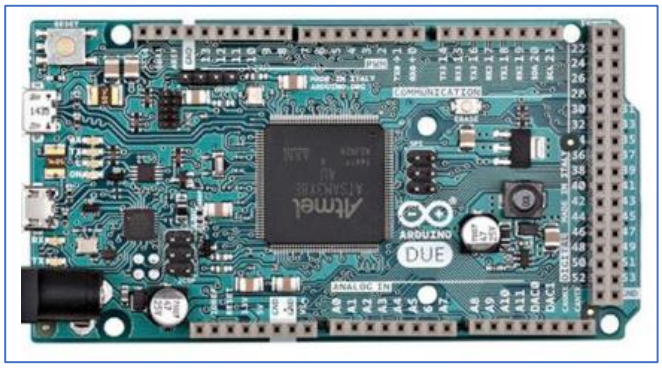


Figura 42. *Placa de desarrollo Arduino Due.*

Arduino Due es una placa de microcontrolador basada en la CPU Atmel SAM3X8E ARM Cortex-M3. Es la primera placa Arduino basada en un microcontrolador de núcleo ARM de 32 bits. Tiene 54 pines digitales de entrada / salida (de los cuales 12 se pueden usarse como salidas PWM), 12 entradas analógicas, 4 UART (puertos serie de hardware), un reloj de 84 MHz, una conexión compatible con OTG USB, 2 DAC (digital a analógico), 2 TWI, un conector de alimentación, un conector SPI, un conector JTAG, un botón de reinicio y un botón de borrado. Es la placa perfecta para proyectos de Arduino potentes y de mayor escala como en nuestro caso para el sistema de control. [14]

Especificaciones técnicas:

Microcontroller	AT91SAM3X8E
Operating Voltage	3.3V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-16V
Digital I/O Pins	54 (of which 12 provide PWM output)
Analog Input Pins	12
Analog Output Pins	2 (DAC)
Total DC Output Current on all I/O lines	130 mA
DC Current for 3.3V Pin	800 mA
DC Current for 5V Pin	800 mA
Flash Memory	512 KB all available for the user applications
SRAM	96 KB (two banks: 64KB and 32KB)
Clock Speed	84 MHz
Length	101.52 mm
Width	53.3 mm
Weight	36 g

En la tabla anterior, cabe resaltar tres parámetros importantes en las especificaciones técnicas del Arduino, la alimentación de la placa es de 7-12v y el voltaje de operación es de 3.3v, lo cual es excelente ya que estamos trabajando en baja tensión comparada con otras placas y muy importante la velocidad del Clock que es de 84MHz ya que al tratarse de un microprocesador de 32bits, obtenemos grandes velocidades de computación que es junto lo que necesitamos para mantener estabilizado el sistema.

Otra ventaja de desarrollar en la plataforma de Arduino es que contamos con un alto soporte por parte de otros desarrolladores, ya que Arduino es código libre, además de ser una de las placas para desarrollo electrónico más utilizadas gracias a su fácil programación ya que está basada en C++ y posee un alto catálogo de librerías que facilitan la implementación del sistema de control.

4.1.7 Sensores

Acelerómetro, giróscopo, magnetómetro

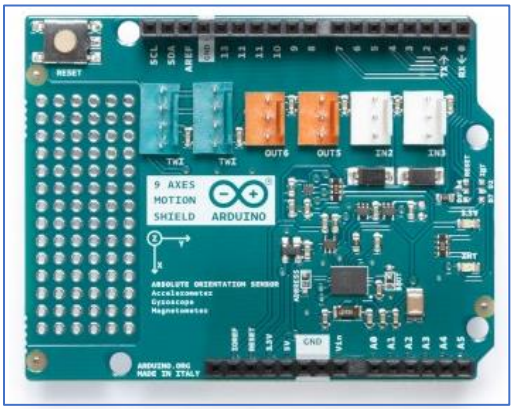


Figura 43. *Arduino 9 Axis Motion Shield.*

El Arduino 9 Axis Motion Shield se basa en el sensor de orientación absoluta BNO055 de Bosch Sensortec GmbH que integra un acelerómetro triaxial de 14 bits, un giroscopio triaxial de 16 bits con un rango de ± 2000 grados por segundo y un sensor geomagnético triaxial con un 32 microcontrolador de bits que ejecuta el software BSX3.0 FusionLib. El sensor presenta datos de aceleración tridimensional, velocidad de guiñada y fuerza de campo magnético, cada uno en 3 ejes perpendiculares. [15]

Características

Proporciona las siguientes señales o salidas:

- Cuaternarios
- Ángulos de Euler
- Vector de rotación y vector de gravedad
- Aceleración lineal

Además, incorpora un motor de interrupción inteligente, que permite activar interrupciones basadas en

- reconocimiento de no movimiento o lento movimiento
- cualquier detección de movimiento (pendiente)

Utilizaremos este Shield de Arduino ya que nos ofrece una gran resolución para los giróscopos y el acelerómetro lo cual nos permitirá una mayor precisión para determinar los ángulos, este sensor cuenta con una librería de Arduino que nos permite obtener los ángulos de Euler, que nos serán necesarios para conseguir los ángulos de cabeceo, alabeo y guiñada en el cuadricóptero.

Tarjeta Wifi

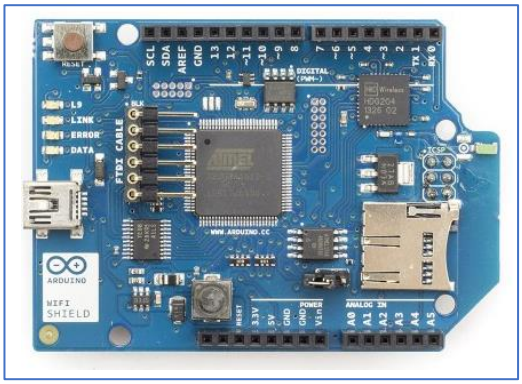


Figura 44. *Arduino WIFI Shield*

Básicamente la función de este Shield es la de proporcionar el Arduino con conexión internet de forma inalámbrica para de esta forma poder interactuar de manera remota con la plataforma de Matlab/Simulink.

4.2. Programacion/Código Arduino

Luego de seleccionar los componentes necesarios para la realización de la implementación real del sistema de control, pasamos a programar el cuadricóptero sobre la IDE de Arduino, donde se incluirá la programación de los sensores de acelerómetro, giróscopo y el magnetómetro al igual que la tarjeta wifi para comunicar con Simulink.

Se implementará un control PID para controlar los ángulos de cabeceo, alabeo y guiñada, de esta forma se intentará estabilizar el sistema y conseguir una respuesta ante las perturbaciones.

A continuación, se presenta el código implementado en el Arduino.

```
#include "NAxisMotion.h"
#include <Wire.h>
#include <Servo.h>
#include <SPI.h>
#include <WiFi.h>
```

1

```
char ssid[] = "NetGear" ;
char pass[] = "abcd1234" ;
int status = WL_IDLE_STATUS ;
WiFiServer server(5000) ;
WiFiClient client ;
```

2

```
#define MAX_P 10
#define MIN_P -10
#define MAX_R 10
#define MIN_R -10
#define MAX_KP 10
#define MAX_KI 1
#define MAX_KD 1
#define MAX_AC 15
#define M_NOM1 40
#define M_NOM2 40
#define M_NOM3 40
#define M_NOM4 40
```

```
int t1 = 0 ;
int t2 = 0 ;
int T = 10 ;
float Tm = T/1000.0 ;
int N = 10 ;
byte cc = 0 ;

byte D1 = 0 ;
byte D2 = 0 ;
byte D3 = 0 ;
byte D4 = 0 ;
byte D5 = 0 ;
```

```
NAxisMotion mySensor ;
unsigned long lastStreamTime = 0 ;
const int streamPeriod = 8 ;
bool updateSensorData = true ;
```

```
float Pitch = 0.0 ;
float Roll = 0.0 ;
float Yaw = 0.0 ;
byte NC = 0 ;
```

3

```
Servo esc1 ;
Servo esc2 ;
Servo esc3 ;
Servo esc4 ;
int motor2 ;
int motor3 ;
int motor4 ;

byte M1 = 0 ;
byte M2 = 0 ;
byte M3 = 0 ;
byte M4 = 0 ;
```

4

```
//PID
float ref_p = 0.0 ; float ref_r = 0.0 ; float ref_y = 0 ;  //Referencia de los PID
float err_p = 0.0 ; float err_r = 0.0 ; float err_y = 0.0 ;
float err_p_f = 0.0 ; float err_r_f = 0.0 ; float err_y_f = 0.0 ;
float err_p1 = 0.0 ; float err_r1 = 0.0 ; float err_y1 = 0.0 ;
float err_p1_f = 0.0 ; float err_r1_f = 0.0 ; float err_y1_f = 0.0 ;
float acc_p = 0.0 ; float acc_r = 0.0 ; float acc_y = 0.0 ;
float Ik_p = 0.0 ; float Ik_r = 0.0 ; float Ik_y = 0.0 ;

int comp = 0; //Compensacion para los motores 3 y 2
float Kp = 0.0 ; float Ki = 0.0 ; float Kd = 0.0 ;
```

5

```
//FILTRO
float Sip = 0.0 ; float Sip_1 = 0.0 ; float Sip_2 = 0.0 ;
float Sop = 0.0 ; float Sop_1 = 0.0 ; float Sop_2 = 0.0 ;
float Sir = 0.0 ; float Sir_1 = 0.0 ; float Sir_2 = 0.0 ;
float Sor = 0.0 ; float Sor_1 = 0.0 ; float Sor_2 = 0.0 ;
float Siy = 0.0 ; float Siy_1 = 0.0 ; float Siy_2 = 0.0 ;
float Soy = 0.0 ; float Soy_1 = 0.0 ; float Soy_2 = 0.0 ;
```

6

(1) Librerías utilizadas.
(2) Variables para inicializar el módulo wifi.
(3) Variables para inicializar el módulo 9 axis (acelerómetro, giróscopo, magnetómetro).
(4) Variables para inicializar los ESC para controlar los motores.
(5) Variables para el controlador PID.
(6) Variables para el filtro.

```
void setup()
{
  esc3.attach(11) ;
  esc4.attach(12) ;
  esc1.attach(8) ;
  esc2.attach(9) ;

  esc3.writeMicroseconds(1000) ;
  esc4.writeMicroseconds(1000) ;
  esc1.writeMicroseconds(1000) ;
  esc2.writeMicroseconds(1000) ;

  delay(5000); //Esperar 5 segundos para hacer la activacion
```

Configuración de los ESC, configurando los pines que están conectados e iniciándolos con los pulsos de inicio y activándolos.

```
Serial.begin(115200) ;
I2C.begin() ;

mySensor.initSensor() ;
mySensor.setOperationMode(OPERATION_MODE_NDOF) ;
mySensor.setUpdateMode(MANUAL) ;
mySensor.updateAccelConfig() ;
updateSensorData = true ;
Serial.println() ;
Serial.println("Default accelerometer configuration settings...") ;
Serial.print("Range: ") ;
Serial.println(mySensor.readAccelRange()) ;
Serial.print("Bandwidth: ") ;
Serial.println(mySensor.readAccelBandwidth()) ;
Serial.print("Power Mode: ") ;
Serial.println(mySensor.readAccelPowerMode()) ;
Serial.println("Streaming in ...") ;
```

Configuración de los sensores acelerómetros, giróscopos y magnetómetro, además de seleccionar el modo de operación para recibir las señales del ángulo de cabeceo, alabeo y guiñada.

```
if (WiFi.status() == WL_NO_SHIELD) {
  Serial.println("WiFi shield not present") ;
  while (true) ;
}
String fv = WiFi.firmwareVersion() ;
if (fv != "1.1.0") Serial.println("Please upgrade the firmware") ;
while ( status != WL_CONNECTED) {
  Serial.print("Attempting to connect to WPA SSID: ") ;
  Serial.println(ssid) ;
  status = WiFi.begin(ssid, pass) ;
  delay(1000) ;
}
server.begin() ;
Serial.print("You're connected to the network") ;
printCurrentNet() ;
printWifiData() ;
digitalWrite(51,HIGH) ;
```

Configuración el módulo wifi, pasándole como parámetro en nombre de la red a la que queremos conectarnos, al igual que la contraseña.

En esta parte obtenemos los valores de los ángulos de pitch (cabeceo), roll (alabeo), heading (guiñada), cabe destacar que se encuentra en el loop, quiere decir que estos valores se actualizan acorde al tiempo de muestreo.

```
void loop()
{

  // ACTUALIZACIÖN SENSOR
  if ((millis()-lastStreamTime)>=streamPeriod) {
    lastStreamTime=millis() ;
    mySensor.updateEuler() ;
    Pitch=mySensor.readEulerPitch()+3.50 ; if (Pitch>45) {Pitch=45 ;} ; if (Pitch<-45) {Pitch=-45 ;}
    Roll=mySensor.readEulerRoll()+0.1; if (Roll>45) {Roll=45 ;} ; if (Roll<-45) {Roll=-45 ;}
    Yaw=mySensor.readEulerHeading() ; if (Yaw>90) {Yaw=90 ;} ; if (Yaw<-90) {Yaw=-90 ;}
  }
```

```
// COMUNICACIÓN TCP/IP
if (cc>=N) {
  cc=0 ;
  client=server.available() ;
  if (client.connected()) {
    Serial.println(client.available()) ;
    while (client.available()>5) {
      ref_p=(D1*(MAX_P-MIN_P)/255.0)+MIN_P ;
      ref_r=(D2*(MAX_R-MIN_R)/255.0)+MIN_R ;
      Kp=(D3*MAX_KP/255.0) ;
      Ki=(D4*MAX_KI/255.0) ;
      Kd=(D5*MAX_KD/255.0) ;
    }
    client.flush() ;
    server.write((byte)((Pitch+45)*(255.0/90.0))) ;
    server.write((byte)((Roll+45)*(255.0/90.0))) ;
    server.write((byte)((Yaw)*(255.0/360.0))) ;
    server.write((byte)((acc_p+MAX_AC)*(255.0/(2*MAX_AC)))) ;
    server.write((byte)((acc_r+MAX_AC)*(255.0/(2*MAX_AC)))) ;
  } else {
    D1=0 ; D2=0 ; D3=0 ; D4=0 ;
    Ik_p=0.0 ; Ik_r=0.0 ; Ik_y=0.0 ;
  }
} else {cc++ ;}
```

Una vez configurada la comunión wifi, recibimos datos vía tcp/ip a través de Simulink, recibimos la acción de control de la Kp, Ki y Kd, en donde con estos valores sintonizaremos el controlador PID para nuestro sistema de control.

Después de recibir los valores de Kp, Ki y Kd, implementamos el sistema de control, que es controlador PID además de agregar un filtro pasa baja para mejorar el comportamiento y darle mejor estabilidad al sistema eliminando los ruidos producidos.

```
// CALCULO ACCIONES DE CONTROL
Serial.print(Roll); Serial.print("  ") ;
Serial.print(Pitch); Serial.print("  ") ;
err_p=ref_p-Pitch ; err_r=ref_r-Roll ; err_y=ref_y-Yaw ;
Ik_p=err_p*Tm+Ik_p ; Ik_r=err_r*Tm+Ik_r ; Ik_y=err_y*Tm+Ik_y ;

Sip=err_p ; Sop=0.0976*Sip+0.1953*Sip_1+0.0976*Sip_2+0.9428*Sop_1-0.3333*Sop_2 ;
Sip_2=Sip_1 ; Sip_1=Sip ; Sop_2=Sop_1 ; Sop_1=Sop ; err_p_f=Sop ;

Sir=err_r ; Sor=0.0976*Sir+0.1953*Sir_1+0.0976*Sir_2+0.9428*Sor_1-0.3333*Sor_2 ;
Sir_2=Sir_1 ; Sir_1=Sir ; Sor_2=Sor_1 ; Sor_1=Sor ; err_r_f=Sor ;

Siy=err_y ; Soy=0.0976*Siy+0.1953*Siy_1+0.0976*Siy_2+0.9428*Soy_1-0.3333*Soy_2 ;
Siy_2=Siy_1 ; Siy_1=Siy ; Soy_2=Soy_1 ; Soy_1=Soy ; err_y_f=Soy ;

acc_p=Kp*err_p + Ki*Ik_p + Kd*(err_p_f-err_p1_f)/Tm ; acc_r=Kp*err_r + Ki*Ik_r + Kd*(err_r_f-err_r1_f)/Tm ; acc_y=Kp*err_y + Ki*Ik_y + Kd*(err_y_f-err_y1_f)/Tm ;
err_p1_f=err_p_f ; err_r1_f=err_r_f ; err_y1_f=err_y_f ;

if (acc_p>MAX_AC) {acc_p=MAX_AC ;} ; if (acc_r>MAX_AC) {acc_r=MAX_AC ;} ; if (acc_y>MAX_AC) {acc_y=MAX_AC ;}
if (acc_p<-1*MAX_AC) {acc_p=-1*MAX_AC ;} ; if (acc_r<-1*MAX_AC) {acc_r=-1*MAX_AC ;} ; if (acc_y<-1*MAX_AC) {acc_y=-1*MAX_AC ;}
```

```
M1=(byte)(M_NOM1-acc_p+acc_r+acc_y) ;
M2=(byte)(M_NOM2+acc_p+acc_r-acc_y) ;
M3=(byte)(M_NOM3+acc_p-acc_r+acc_y) ;
M4=(byte)(M_NOM4-acc_p-acc_r-acc_y) ;

motor3=map(M3,0,255,1000,2000) ;
motor4=map(M4,0,255,1000,2000) ;
motor1=map(M1,0,255,1000,2000) ;
motor2=map(M2,0,255,1000,2000) ;

esc3.writeMicroseconds(motor3) ;
esc4.writeMicroseconds(motor4) ;
esc1.writeMicroseconds(motor1) ;
esc2.writeMicroseconds(motor2) ;
```

Por último se aplican estos valores de control a los motores para así de esta forma realizar el movimiento del cuadricóptero, ya sea sumando la acción de control del cabeceo para aumentar la velocidad de los motores 2 y 3; disminuyendo 1 y 4, así como sumando la acción de control del alabeo para aumentar la velocidad para los motores 1 y 2; disminuyendo 3 y 4, y finalmente para el ángulo de guiñada sumamos la acción de control en los motores 1 y 3, mientras que 2 y 4 restamos la acción de control.

4.2.2 Resultados de las pruebas reales

A continuación, se presentan los resultados de la implementación del sistema de control en el cuadricóptero.

Seguimiento de la referencia del ángulo de alabeo con entrada senoidal

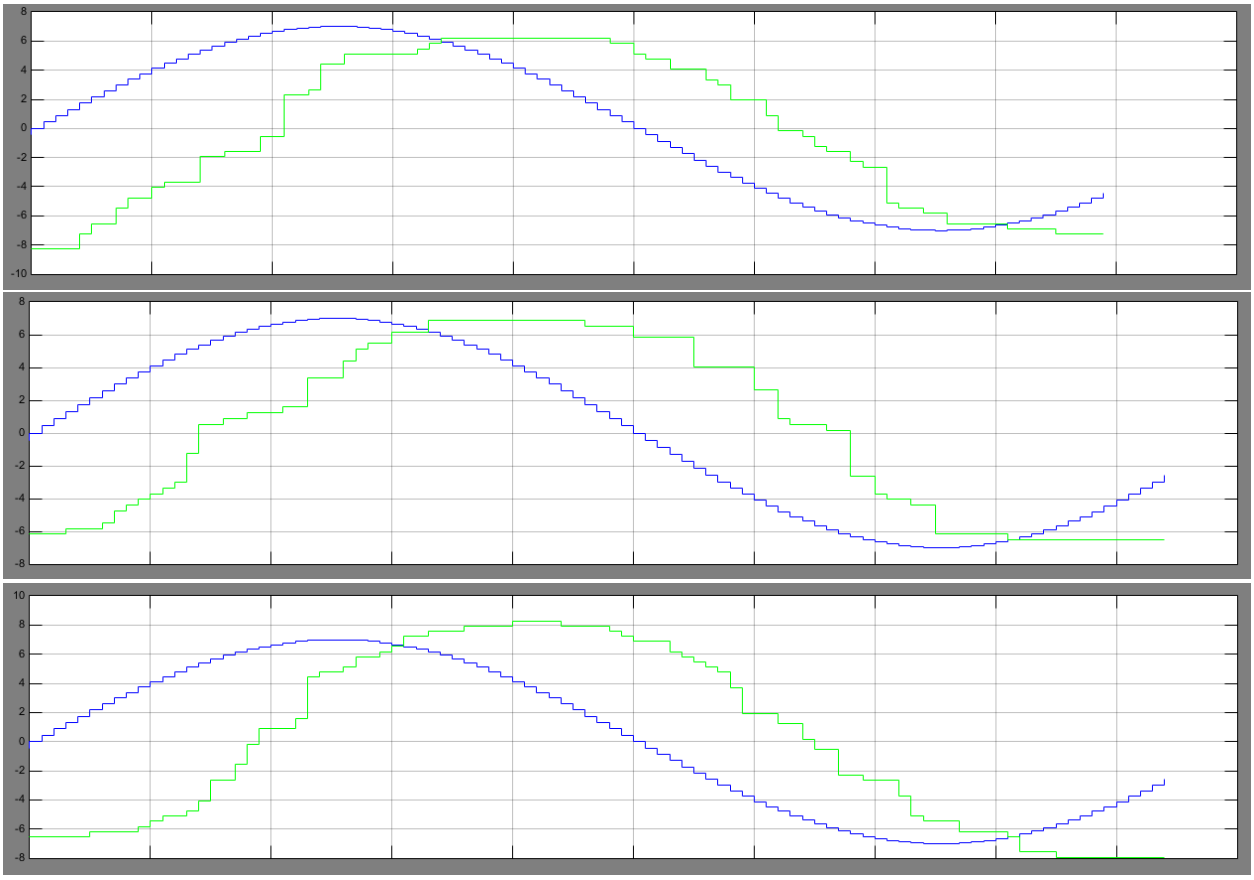
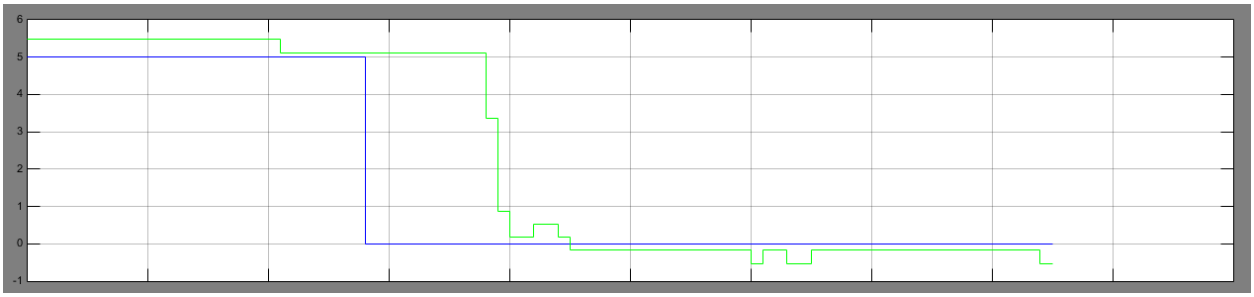


Figura 45. *Referencia vs ángulo de alabeo*

Seguimiento de la referencia del ángulo de alabeo con entrada de escalón



Leyenda:

Referencia

Angulo de cabeceo

5. Conclusión

Al finalizar este trabajo se conseguido alcanzar la mayoría de los objetivos propuestos durante el desarrollo de este proyecto, se ha desarrollado un diseño 3D de los elementos que conforman el cuadricóptero, además de realizar el correcto modelado y simulación de este.

A partir de las pruebas y simulaciones realizadas se obtienen las siguientes conclusiones:

El sistema de control propuesto y diseñado cumple con el objetivo principal la controlar de manera correcta la actitud del cuadricóptero, mediante el PD, nos aseguramos de que el sistema es capaz de seguir la referencia en todo momento de los tres ángulos (cabeceo, alabeo y guiñada). Esto lo podemos observar en el capítulo "Resultados de la simulación" en donde se abordan con más detalles las pruebas realizadas y confirmar el funcionamiento del sistema de control.

Es importante señalar que esta tarea no fue fácil ya que supuso un reto a la hora de diseñar dicho controlador en Simulink ya que tuvieron que implementarse tres controladores para realizar el control y entrelazarlos con los 4 motores para que puedan actuar de manera simultánea.

Uno de los objetivos que se tenía previsto, pero no fue alcanzando fue la implementación real del sistema de control diseñado en un cuadricóptero real, debido a diversos problemas de tiempo y de complejidad del sistema no se logró implementar en su totalidad ya que solo se pudo realizar el control del ángulo de alabeo.

6. Bibliografía

[1] K. Mizokami, «Popular Mechanics,» 9 Febrero 2018. [En línea]. Disponible: https://www.popularmechanics.com/military/a16762519/the-marine-corps-latest-weapon-is-a-quadcopter/. [Último acceso: 20 Junio 2018].

[2] Ardupilot, «Ardupilot.org,» [En línea]. Disponible: http://ardupilot.org/. [Último acceso: 15 Junio 2018].

[3] T. Luukkonen, «Modelling and contol of quadcopter,» *Aolto University,* pp. 1-26, 2011.

[4] J. Hewitt, «extremetech,» 14 Junio 2013. [En línea]. Disponible: https://www.extremetech.com/wp-content/uploads/2013/06/Quadrotor-dynamics-348x196.jpg. [Último acceso: 22 Abril 2018].

[5] T. S. Alderete, «Simulator aero modele implementation,» *NASA Ames Researh Center,* pp. 1-21, 2007.

[6] P. Castillo, R. Lozano y A. Dzul, «Stabilisation of a mini rotocraft with four rotors,» *IEEE Control Ssytem Magazine,* pp. 45-55, 2005.

[7] G. V. Raffo, M. G. Ortega y F. R. Rubio, «An integral predictive/nonlinear H∞ control structure for a quadrotor helicopter,» *Automatic,* vol. 46, nº 1, pp. 29-39, 2010.

[8] H. Bouadi y M. Tadjine, «Nonlinear observer design and sliding mode control of four rotors helicopter,» *World Academy of Science, Engineering and Technology, International Journal of Aerospace and Mechanical Engineering,* vol. 1, nº 7, pp. 1-6, 2007.

[9] TimmmyK, «Wikimedia Commons,» 7 Noviembre 2014. [En línea]. Disponible: https://commons.wikimedia.org/wiki/File:PID_varyingP.jpg. [Último acceso: 22 Abril 2018].

[10] V. M. Alfaro Ruiz, «Metodos de sintonizacion de controladores PID que operan como reguladores,» *Ingenieria,* vol. 12, nº 1 y 2, pp. 21-36, 2002.

[11] V. M. Alfaro Ruiz , «Metodos de sintonizacion de controladores PID que operan como servomecanismos,» *Ingenieria,* vol. 13, nº 1 y 2, pp. 13-29, 2003.

[12] E. Solera Saborio y V. M. Alfaro Ruiz, «Sintonizacion de controladores PI Y PID utilizando modelos de polo doble mas tiempo muerto,» *Ingenieria,* vol. 16, nº 2, pp. 23-31, 2006.

[13] J. G. Ziegler y B. N. Nichols, «Optimum Settings for Automatic Controls,» *ASME Transactions (EUA),* vol. 64, pp. 759-768, 1942.

[14] «Arduino Due,» [En línea]. Disponible: https://store.arduino.cc/arduino-due. [Último acceso: 15 Junio 2018].

[15] «ARDUINO SHIELD 9 AXIS MOTION,» [En línea]. Disponible: https://store.arduino.cc/arduino-9-axis-motion-shield. [Último acceso: 15 Junio 2018].

[16] «http://e-ducativa.catedu.es,» [En línea]. Disponible: http://e-ducativa.catedu.es/44700165/aula/archivos/repositorio/4750/4926/html/15_controlador_de_accin_proporcional_integral_y_derivativa_pid.html. [Último acceso: 14 Junio 2018].

[17] R. Diaz, «muydrones,» [En línea]. Disponible: https://www.muydrones.com/funcionamiento-del-cuadricoptero/. [Último acceso: 15 Junio 2018].

[18] «comprardrones,» [En línea]. Disponible: https://www.comprardrones.online/como-funcionan-las-helices-de-un-drone/. [Último acceso: 16 Junio 2018].

7. Anexos

7.1. Presupuesto del Proyecto

A continuación, se detallará el cálculo del presupuesto correspondiente a los componentes del cuadricóptero.

Producto	Fabricante	Descripción	Cantidad	Precio (€)
KIT de montaje F450 ARTF CC3D EVO	ModelTronic	Kit completo para el ensamblaje del dron. Incluye: - Estructura - Hélices - Motores - ESC (Variadores) -Placa distribución eléctrica	1	109.99
Batería	Turnigy	Batería para la alimentación del Cuadricóptero (2200mah 3S)	1	9.42
Arduino DUE	Arduino	Placa de desarrollo	1	34.00
Arduino 9 Axis Motion Shield		Sensores	1	23.90
Arduino WIFI Shield		Placa de comunicación Wifi	1	10.55
			Total	187.86

En cuanto los programas utilizados para el desarrollo del proyecto se encuentran los siguientes:

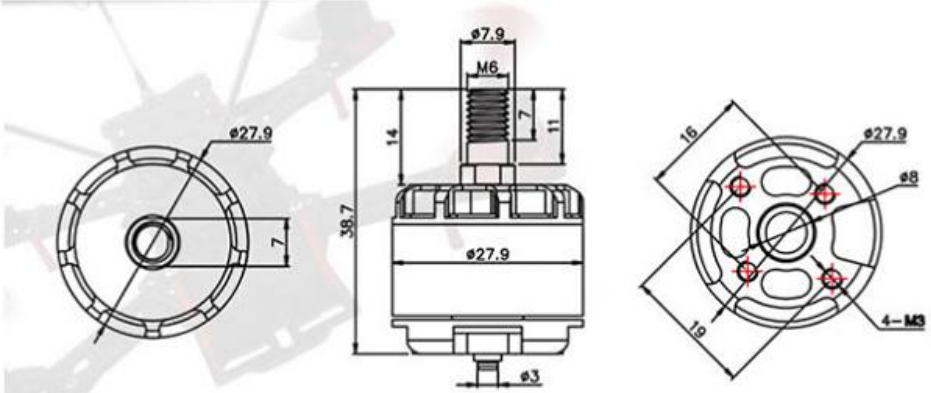
Producto	Fabricante	Descripción	Cantidad	Precio (€)
Licencia Matlab Student R2018a	MathWorks	Herramientas para el desarrollo del sistema de control Incluye: -Matlab/Simulink - Simscape -Simscape Multibody -Simulink 3D Animation -Simulink Design Optimization	1	97.00
Licencia SolidWorks Standard	Dassault Systèmes	Herramientas para el diseño 3D del modelo	1	8100.00
			Total	8,197.00

Precio TOTAL del Proyecto: 187.86 + 8197.00 = 8,384.86

7.2. Planos y hoja de especificaciones

7.2.1 Motor (x4)

Specification
- Threads : CCW (for clockwise motor)
- KV : 900kv
- Poles : 14
- No. of Li-Po cell : 3~4S
- Motor Dimensions(Dia.*Len) : Φ27.9×38.7mm
- Shaft Dia. M6
- Weight : 50g
- Mounting Hole : M3 screws / 16mm x 19mm
- Recommended Propeller Size : 8~10 inch
- Recommended ESC : 20A or above

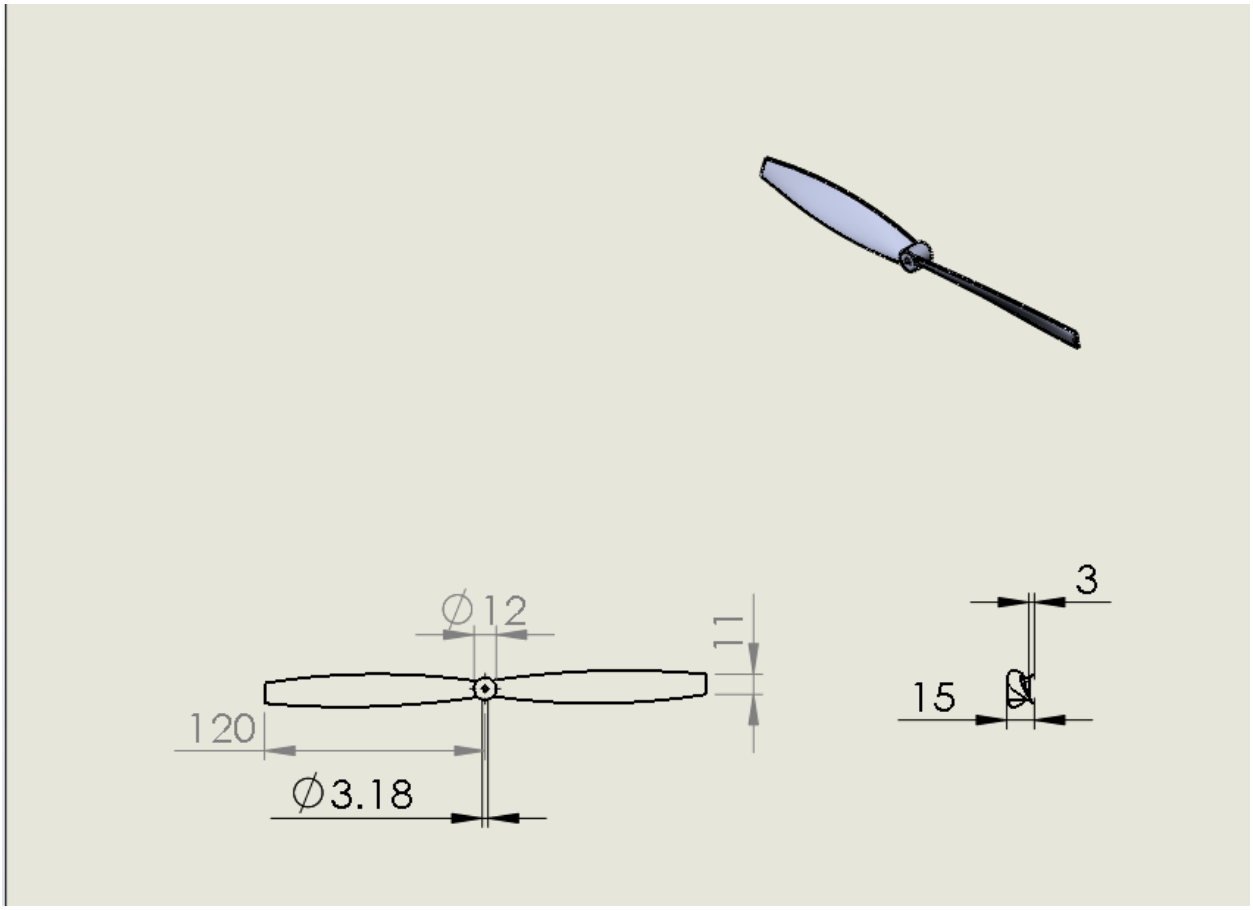


Framework:12N14P
KV:900KV
Length:38.7MM
Diameter:27.9MM

No.of cells:3-4S
Max.thrust:720g
Shaft:3MM
Propeller:8"-10"
weight:50g

Motor type	The voltage (V)	Propeller size	current (A)	thrust (G)	power (W)	efficiency (G/W)	speed (RPM)
MT2212 II - 900KV	11.1	EMAX8045	7.3	470	81.0	5.8	7400
		DJI9450	8.4	590	93.2	6.3	7090
		EMAX1045	9.5	640	105.5	6.1	6530
	14.8	EMAX8045	10.7	720	158.4	4.5	9030
		DJI9450	11.8	850	174.6	4.9	8680

7.2.2 Hélices (x4)



7.2.3 ESC (Variadores de velocidad) (x4)

FPV VERSION ESC
USER MANUAL

DISCLAIMER ►

Thanks for purchasing our Electronic Speed Controller(ESC). High power system for RC model can be very dangerous. Any improper use may result in injury and damage to human and devices.We strongly recomme -nd that you read this manual carefully before use, and abide by its rules. We assume no responsibility for personal injury, property damage or consequential losses resulting from the product.

NOTES ►

1. To avoid any damage to the ESCs from irrational propulsion system and aircraft collocation, please read relevant manual before ESC utilization.
2. Please make sure all the wires and joint parts are well insulated, short-circuits will cause damage to the ESCs. If welding is required, please use welding equipment of equivalent power. Bad welding will lead to disorder over aircraft control, and many other unpredictable problems.
3. Never lock the rotor of motors when ESCs are on the run. Otherwise, fatal damage will be caused to ESCs and possibly to the motors too. When motor stalling occurs, please cut off the throttle or pop batter -ies out immediately.
4. Never use ESCs in high temperature environment. Being used at high temperature, ESCs might trigger Temperature Protection and get damaged.
5. Do not forget to cut off the connection of ESCs and batteries after use.

FEATURES ►

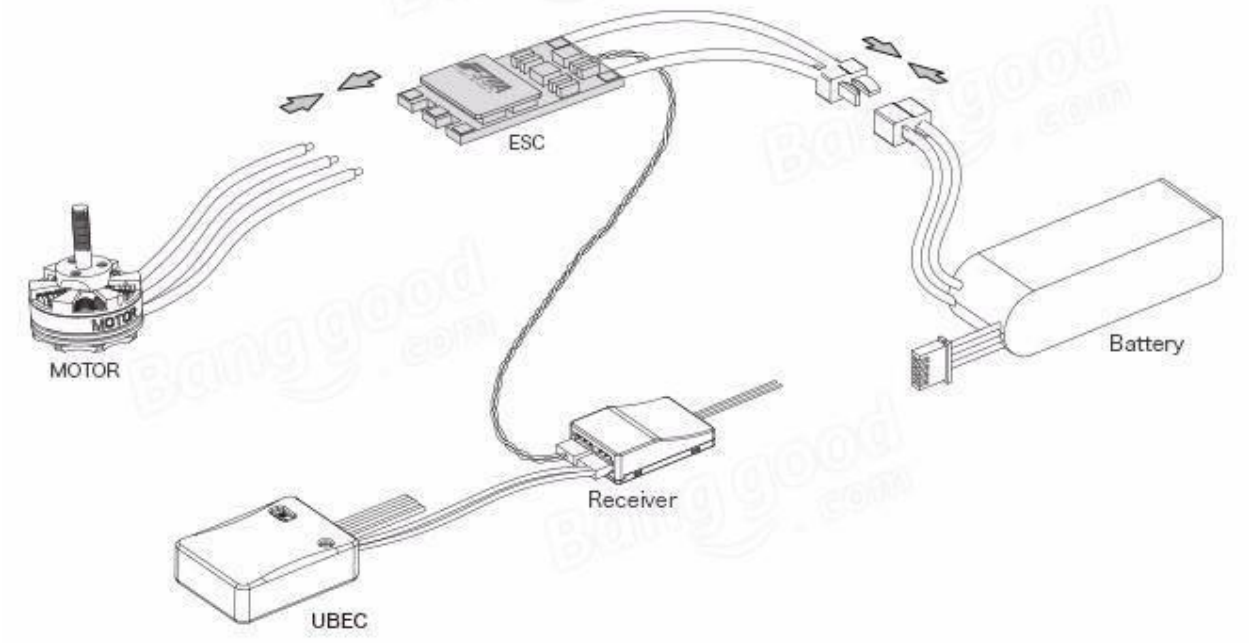
1. Original BLHeli-S firmware.
2. Mini size, ultra-light.
3. Quick parameter adjustment: all the ESC parameters can be adjusted simultaneously via flight controller.
4. Support various throttle signals: 1-2ms, Oneshot125 (125-250us), Oneshot42 (41.7-83.3us) and Multsh -ot (5-25us).
5. Twisted pair(TP) throttle signal cable reduces the crosstalk during signal transmission which guarantees stable flight.

FEATURES ►

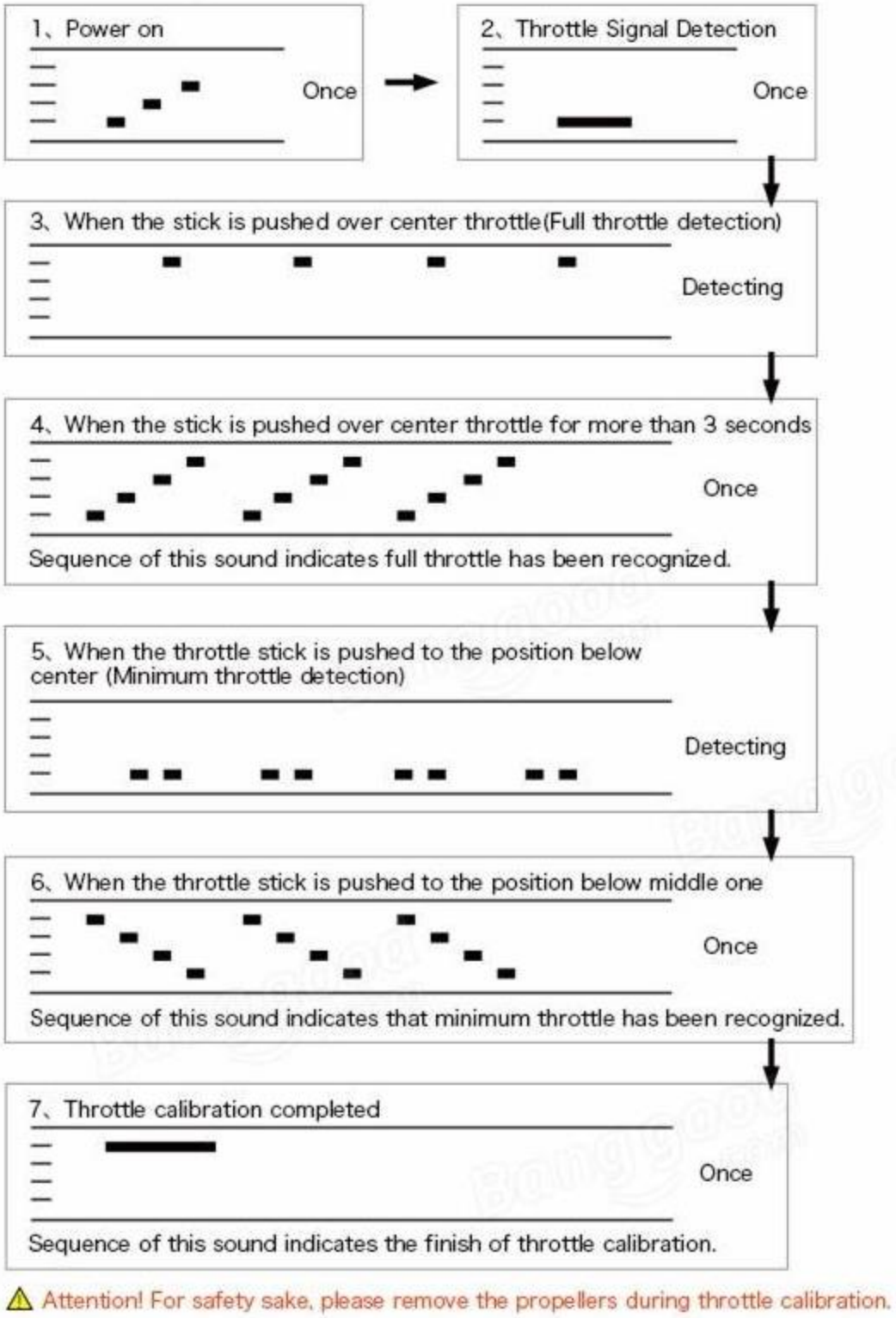
Item	F 30A 6S	F 30A 4S	F 20A 4S
Continuous Current	30A	30A	20A
Burst Current	40A	40A	30A
Lipo(cell)	2-6S	2-4S	2-4S
Weight Wires Included	6.6g	4.3g	3.7g
Dimension L x w x H(mm)	30.5*14.9*5	25.5*12.7*5	23.5*11.1*5
BEC	NO	NO	NO
MCU	EFM8bb21f16G	EFM8bb21f16G	EFM8bb21f16G

USER GUIDE ►

1. Wiring Diagram

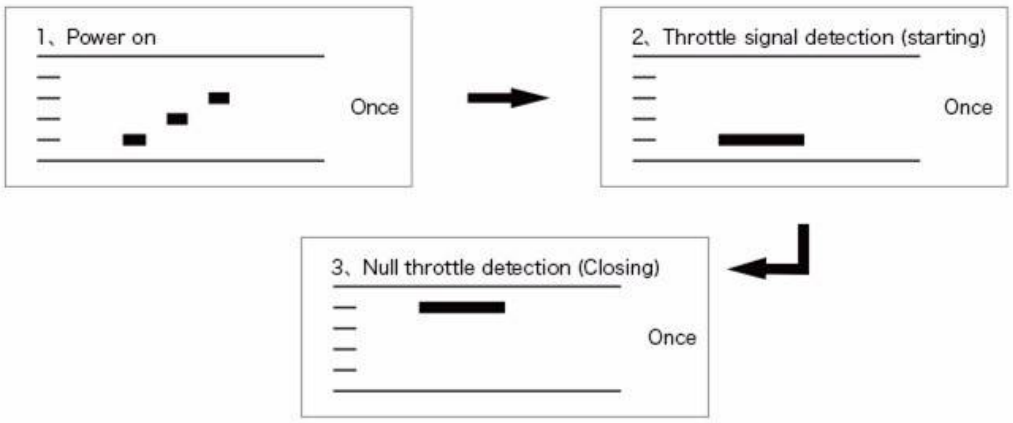


2. Throttle Range Calibration



⚠ Attention! For safety sake, please remove the propellers during throttle calibration.

3. Normal Startup Process



Motors are ready to get started.

PROGRAMMABLE PARAMETERS ▶

	Function	1	2	3	4	5
1	Startup Power	0.031	0.047	0.063	0.094	0.125
2	Temperature Protection	Off	80	90	100	110
3	Low RPM Power Protection	Off	On			
4	Motor Direction	Normal	Reversed	Bidirectional	Bidirectional Rev	High
5	Demag Compensation	Off	Low	High		
6	Motor Timing	Low	Medium Low	Medium	Medium High	High
7	PPM Min Throttle	1100-1692	1148			
8	PPM Max Throttle	1288-2020	1832			
9	PPM Center Throttle	1152-1828	1488			
10	Brake On Stop	Off	On			
11	Beep Strength	Off	2-255	40		
12	Beacon Strength	1-255	80			
13	Beacon Delay	1-10minutes	Infinite	10minutes		

6	7	8	9	10	11	12	13
0.188	0.25	0.38	0.50	0.75	1.00	1.20	1.50
120	130	140					

SETTINGS IN GREY ARE DEFAULT

Adjustable Parameters

1. Startup Power

Startup power refers to the max. power allowed at startup stage, which can be any relative value from 0.031 to 1.5 Real power depends on input throttle, but the min. value cannot be lower than 1/4 of the max. power. In addition, startup power, which restricts the power of rotating direction change, shows its influen -ce on bidirectional setting. In low RPM running, max power which can be adjusted via startup power parameters setting, is limited for BEMF voltage detection. In low RPM running (since from 16.1 version), low startup power means low max power.

2. Temperature Protection

There are off and on modes for this setting.

3. Low RPM Power Protection

It is recommended that this setting be invalid when motors of low kv powered with low voltage. However, invalidity will increase the risk of step out and get motors and ESCs burnt.

4.Motor Drection

Motor direction can be normal, reversed, bidirectional and bidirectional reversed. In bidirectional mode, center throttle stands for null throttle. Throttle position above center one, motors rotate normally, otherw -ise, motors will rotate in a reversed direction. Bidirectional mode will invalid RC parameter setting.

5.Demag. Compensation

Demag. compensation is meant to avoid motor stalling from long time wire demagnetization. High timing helps with that, however, it brings efficiency down. It begins to detect on demag. Compensation occurrence. When motor timing is unavailable, motor rotating direction will be changing according to motor timing esti -mate. Motor power will be cut off before a next direction change. Demag. compensation degree will be calculated. The severer demag. compensation is, the more power will be cut. When demag. compensation is on "OFF" mode, power won' t be cut off. Technically, higher demag. compensation parameter means better protection. Nevertheless, if demag. compensation parameter is set too high, max power drop slightly.

6.Motor Timing

There are low, medium low, medium, medium high and high timing settings, and they are 0° 、7.5° 、15° 、22.5° and 30° Generally, medium timing suits most of the requirements. In case of motor vibration, please try changing motor timing. It takes longer for high induction motors to demagnetize for direction change which leads to motor stalling or vibration on quick throttle increase. This phenomenon occurs especially at low RPM. High timing allows longer time for demagnetization, and thus helps to improve the above mentio -ned issue.

7.Min. Throttle, Max. Throttle & Center Throttle

These settings decide throttle pos. and usually for input signal of 1000-2000us. Other input signals should be interpreted proportionally.Center throttle is for bidirection only.

8.Brake On Stop

There are off and on modes for this setting. Validity of this setting ESCs will generate automatic braking at null throttle. This setting shows no influence at NZ throttle.

9.Beep Strength

Beep strength can be adjusted in compliance with normal operation.

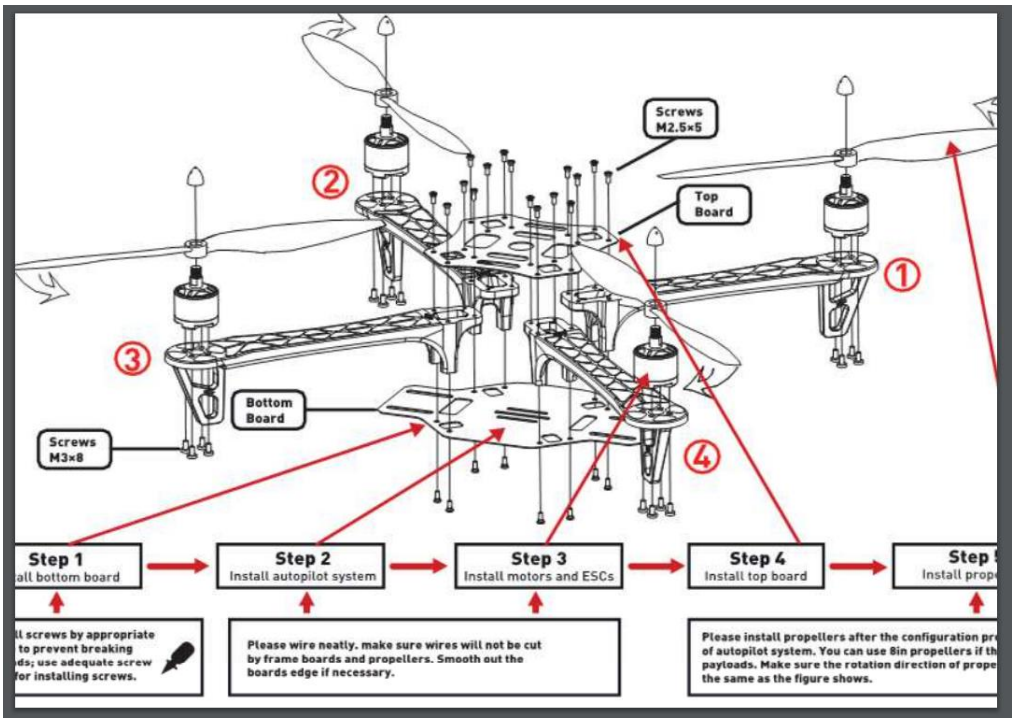
10.Beacon Strength

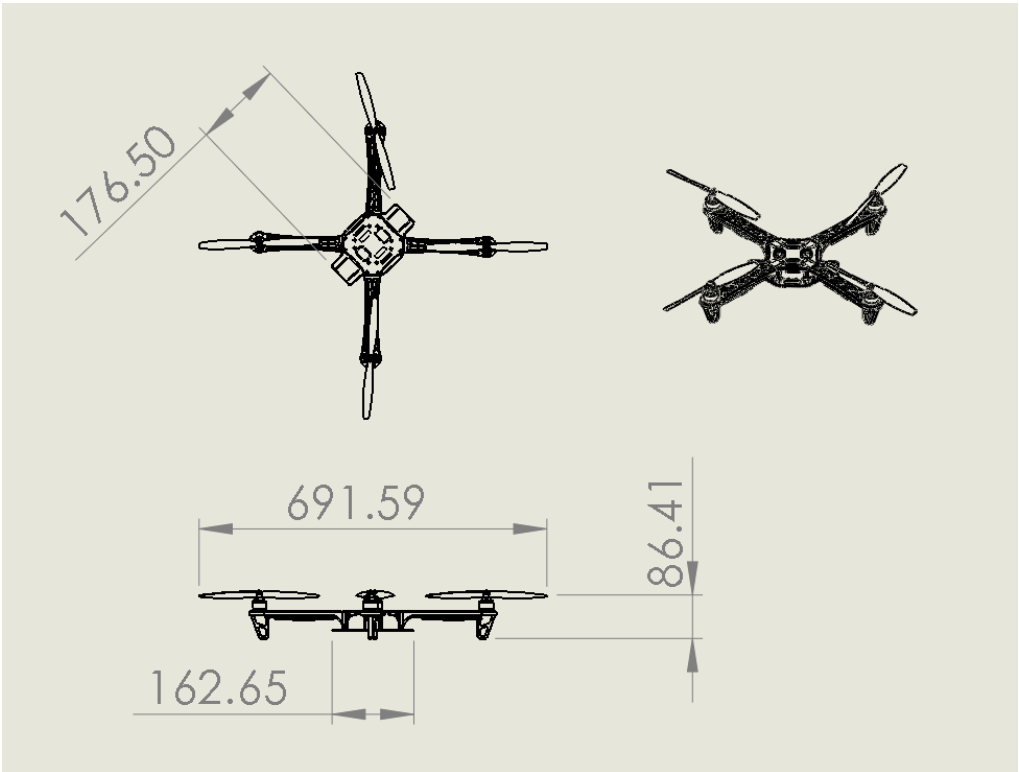
ESCs will emit beacon beeps, once null throttle signal lasts for some time. Please note that high beacon strength brings heat to ESCs and motors.

11.Beacon Delay

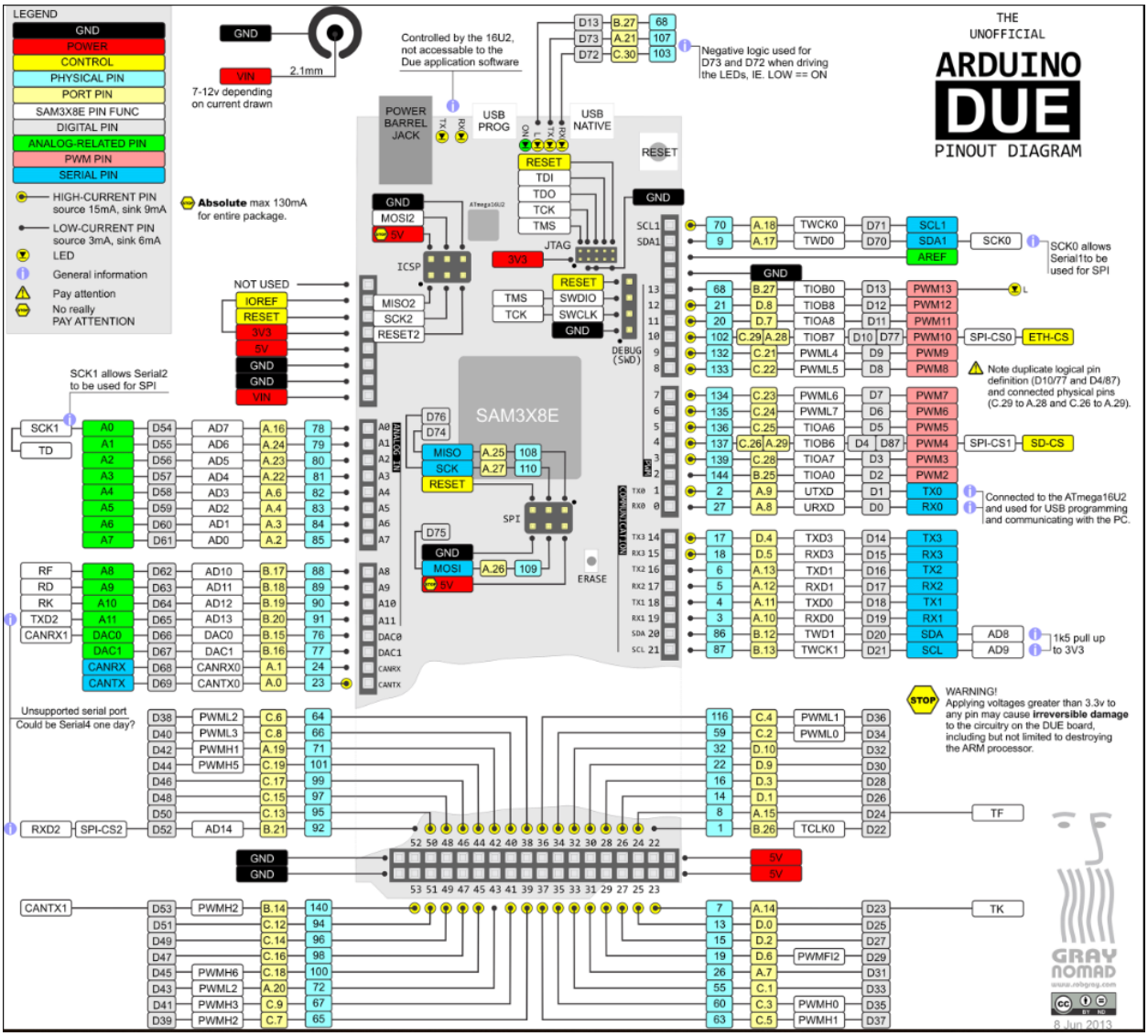
This setting determines the delay in time before beacon beeps.

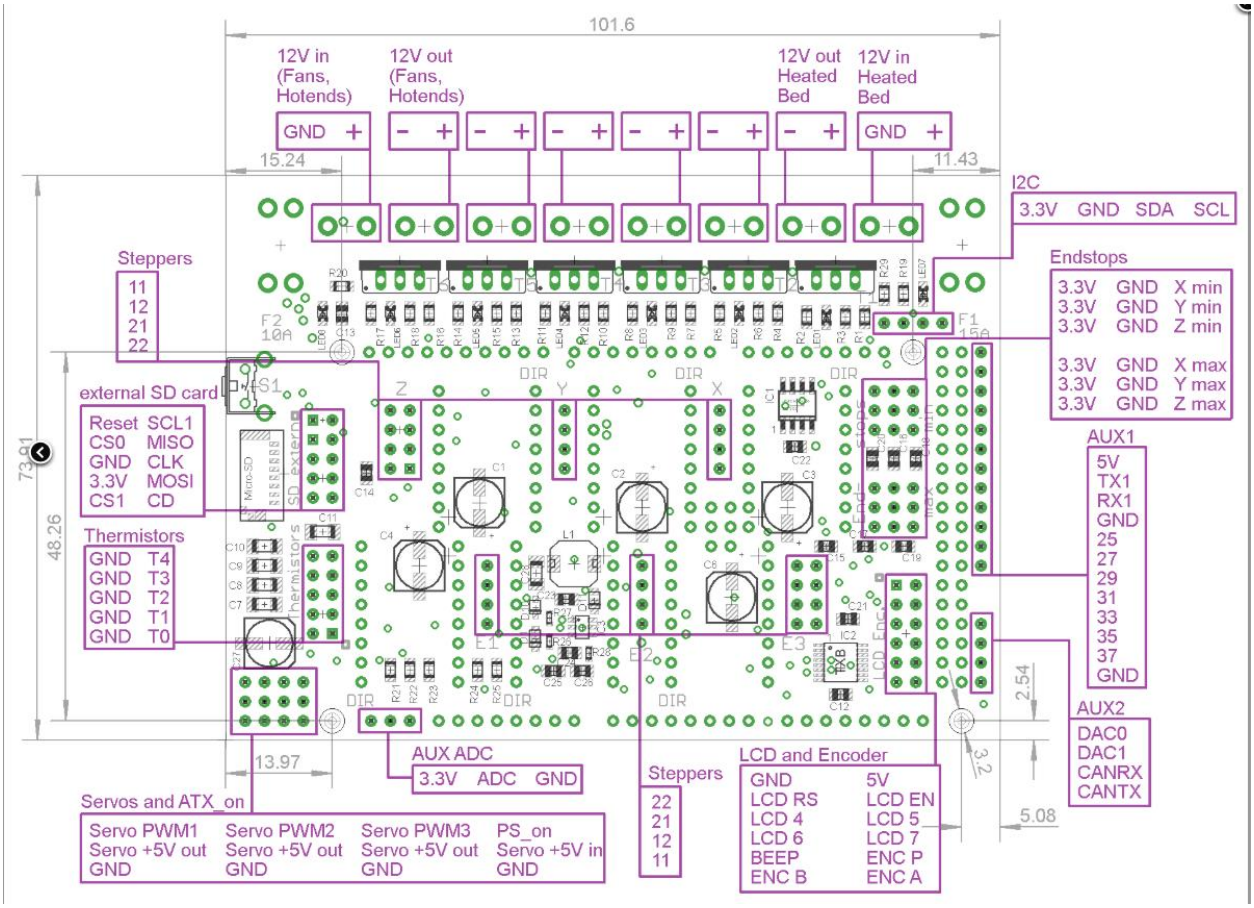
7.2.4 Ensamble del cuadricóptero





7.2.5 Arduino DUE




101.6
12V in (Fans, Hotends)
12V out (Fans, Hotends)
12V out Heated Bed
12V in Heated Bed
GND + - + - + - + - + - + GND +
15.24
11.43
I2C
3.3V GND SDA SCL
Steppers
11
12
21
22
Endstops
3.3V GND X min
3.3V GND Y min
3.3V GND Z min
3.3V GND X max
3.3V GND Y max
3.3V GND Z max
F2 10A
F1 15A
external SD card
Reset SCL1
CS0 MISO
GND CLK
3.3V MOSI
CS1 CD
AUX1
5V
TX1
RX1
GND
25
27
29
31
33
35
37
GND
Thermistors
GND T4
GND T3
GND T2
GND T1
GND T0
AUX2
DAC0
DAC1
CANRX
CANTX
AUX ADC
3.3V ADC GND
Steppers
22
21
12
11
LCD and Encoder
GND 5V
LCD RS LCD EN
LCD 4 LCD 5
LCD 6 LCD 7
BEEP ENC P
ENC B ENC A
Servos and ATX_on
Servo PWM1 Servo PWM2 Servo PWM3 PS_on
Servo +5V out Servo +5V out Servo +5V out Servo +5V in
GND GND GND GND
73.91
48.26
13.97
2.54
3.2
5.08