CONTROL DE UN DRON

Universidad Católica Boliviana “San Pablo”

Carrera:

Ingeniería Mecatrónica

Materia:

Control II

Integrantes:

- Cachi Joel
- Chipana Carlos
- Ortiz Vania

Docente:

Ing. Benjamin Pinaya

Gestión:

2017

La Paz – Bolivia

Contenido

Tabla de figuras

Control de un dron

1. Resumen

Este proyecto consiste en la construcción y la programación de un quadcopter, es decir un helicóptero de cuatro hélices (drone).

Para comodidad se dividió en tres secciones, en la primera tenemos la parte mecánica donde se explica el proceso del armado, los componentes mecánicos y su funcionalidad, la segunda se centra en los componentes electrónicos y la tercera el software desarrollado así como la breve explicación de las librerías usadas

2. Introducción y Antecedentes

El objetivo del presente proyecto consiste en la realización de un drone. También el trabajo de programación de software de control. Desde la obtención de parámetros mediante sensores hasta la modificación del comportamiento de los actuadores en función de ellos.

Los elementos que forman nuestro cuadricoptero serán mencionados a continuación: en el lado mecánico está el esqueleto las 4 hélices y por el lado eléctrico tenemos el micro controlador arduino, el sensor IMU (MPU 6050), reguladores ESC y los motores brushless DC.

En el código de programación se incluye el tratamiento respectivo de la señal del sensor, el filtro complementario para promediar los datos del acelerómetro y giroscopio, el sistema de control PID, la generación de acción de control.

Hoy en día los drones son uno de los dispositivos de mayor presencia, pues desde que pasaron de ser de uso militar como un arma de guerra volador a un robot volador de acceso público.

Las aplicaciones civiles a las que son destinados los drones van aumentando con demasiada rapidez en los últimos años al permitir trabajos que por riesgo o dificultad de accesibilidad no son rentables de realizar de otra manera, las principales son grabación de video, control de infraestructuras industriales, vigilancia de fronteras, búsqueda de supervivientes en zonas afectadas por catástrofes, etc.

La principal ventaja que ofrecen los drone es la inexistencia de personas a bordo de ellos lo que elimina el riesgo de muerte de pilotos y permite realizar funciones que no serían posibles con aeronaves tripuladas como investigación en zonas de toxicidad química y radiológicas, por otro lado está la automatización del aparato siendo capaz de actuar por sí mismo con integración en la inteligencia artificial e incluso coordinarse con otros drones.

A pesar de las grandes posibilidades que nos brindan estos dispositivos, presentan también una serie de desventajas técnicas, económicas y éticas. Las desventajas técnicas tienen en cuenta es que el dispositivo no deja de ser una máquina y requiere una fuente de energía que se va consumiendo y puede quedarse sin energía, también pueden haber fallos de comunicación o inclusive ser hackeados como sucedió en Irak y Afganistán.

Las desventajas económicas influyen en el precio de estos vehículos no tripulados que en algunos casos son demasiado elevados debido a su precisión y funcionalidades extras. Las desventajas éticas son las más importantes como suele ocurrir al aparecer nuevas tecnologías siempre surgen preguntas sobre su dirección de desarrollo.

Por otro aspecto la comercialización indiscriminada de estos aparatos sin regulación, permite que cualquier aficionado disponga de estas unidades con las que tomar fotografías o grabaciones a personas constituyendo una grave amenaza para la privacidad personal. En la otra cara de la moneda la regulación por parte de sistemas de gobierno aumentaría el sofoco del estado policial.

3. Elaboración del proyecto

3.1. Funcionamiento y movimiento de Quadcopter

Antes de comenzar con la construcción del drone es preciso elegir el modelo de la estructura que se va a usar. Habitualmente existen distintos tipos de estructuras “+”, en “x”, en “Y” o en “H” los más usados son en “+” y “x” ya que de esta forma todos los brazos son iguales y el control de los mismos está en el centro de gravedad.

La estructura “+” es la más sencilla de controlar ya que para su desplazamiento solo es necesario actuar sobre dos motores dejando los otros dos constantes pero como hay demasiada información tanto sobre el modelo en “+” como en el “x” se eligió el modelo “x”.

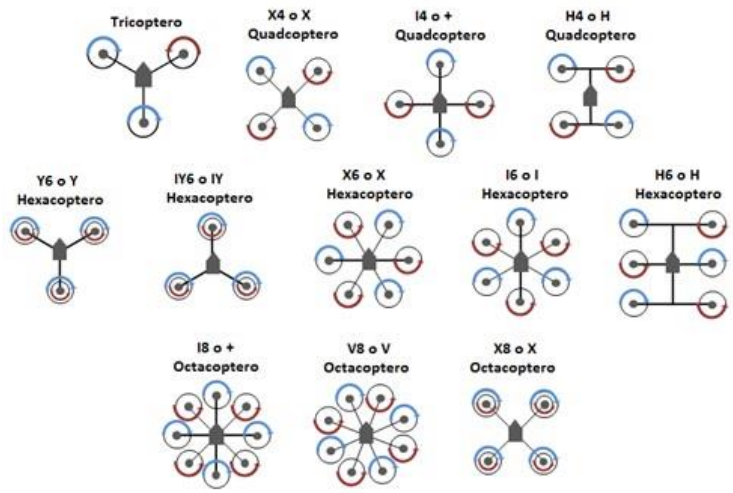


Figura 1. *Tipologías de drones más habituales y disposición de los motores.*[1]

La característica principal del Quadcopter son la disposición simétrica y su rotación asimétrica dos a dos. Esto le permite equilibrarse aplicando rotaciones iguales en los 4 motores, y al mismo tiempo hace calcular los giros.

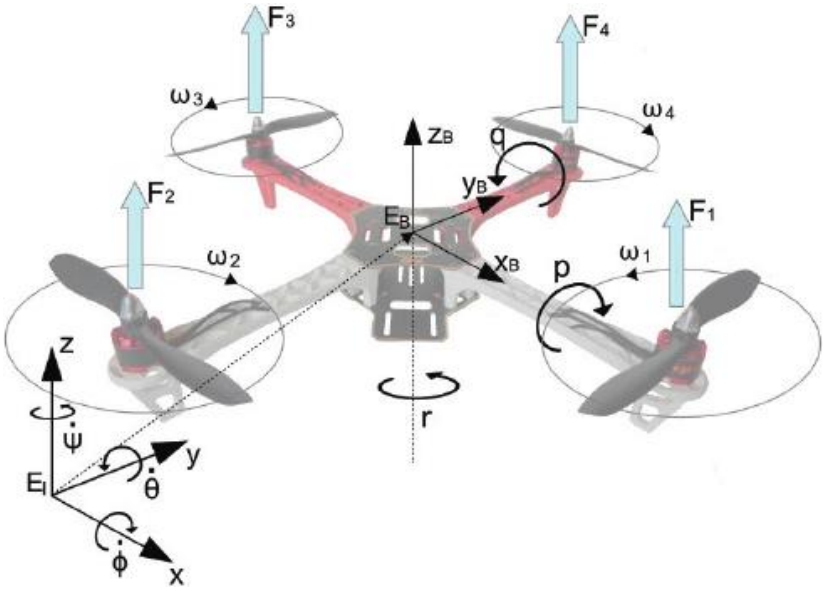


Figura 2. Disposición del Quadcopter[2]

Los pares generados por 1,3 y 2,4 de figura 2 son contrarios y eso auto-estabiliza el Quadcopter siempre y cuando se apliquen empujes de mismo módulo en los 4 motores. De esta manera no se necesita la hélice lateral de un helicóptero, simplificando el cálculo y el diseño.

Así pues las cuatro configuraciones de motores y sus movimientos asociados son:

3.1.1. Control de altitud

La posición base es con los 4 motores aplicando el mismo empuje. Con la cantidad total de empuje podemos llegar a 3 movimientos:

- Estabilizarse en el aire
- Subir altura
- Bajar altura

Figura 3. Control de altura[3]

3.1.2. Control del Yaw

Para ajustar el yaw se sube el empuje de dos motores opuestos mientras los otros dos se mantienen estables.

Figura 4. Control del Yaw[3]

3.1.3. Control del Roll

El movimiento de roll se consigue modificando dos motores opuestos con la misma diferencia, pero en uno se sube y en otro se baja.

Figura 5. Control del Roll[3]

3.1.4. Control del Pitch

El movimiento de pitch se hace de la misma manera que el roll, pero tocando los otros dos motores.

3.2. Componentes utilizados

Una vez que se eligió el modelo del chasis del drone se hará un breve detalle de los componentes utilizados.

3.2.1. Arduino

Para la realización del proyecto se ha decidido trabajar con esta plataforma Arduino ya que se trabajó con él en anteriores cursos, otra razón es la gran cantidad de soporte que existe sobre esta plataforma y grandes comunidades que colaboran con sus aportaciones a la hora de proponer soluciones en el foro de Arduino.

Entre qué modelo de Arduino elegir se usó el más común de todos ya que contábamos con el este es el Arduino UNO ya que cumple con lo que se necesita para la elaboración del proyecto las características básicas requeridas son las siguientes:

- Comunicación serie para intercambiar datos con un módulo bluetooth (TX y RX)
- Comunicación I2C para los distintos sensores, en nuestro caso se utiliza únicamente el MPU 6050.
- 4 Salidas Digitales para los actuadores
- 1 Entrada Analógica para medir el nivel de batería disponible

- Pin de alimentación para periféricos a 5V y 3.3V.

3.2.2. IMU (Unidad de medición inercial)

Un detalle importante que ayudó a tomar la decisión de usar el arduino fue la disposición de un bus i2c el cual pone en comunicación distintos circuitos integrados esto permite la inclusión de este sensor con el que se medirá la inclinación en cada eje y en definitiva la orientación del drone. Este sensor está formado por tres acelerómetros y tres giroscopios con opción de agregarle un magnetómetro igual de tres ejes con el cual se complementa para medir el ángulo z.

Se utiliza este sensor por dos razones principalmente, la primera es su precio ya que es el más competitivo del mercado en los últimos años y que por su uso extendido esta verificada su fiabilidad, la segunda razón es que se puede ampliar con la conexión de un magnetómetro

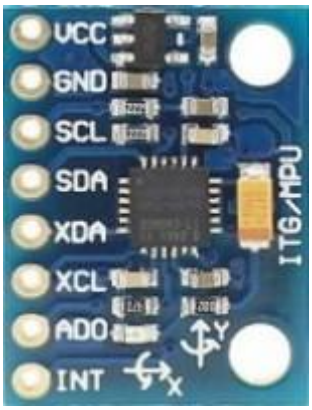


Figura 6. MPU 6050

3.2.3. Motores

Para decidir qué tipo de motor nos hace falta primero se diferenció dos tipos de motores con escobillas y sin escobillas. Los primeros tienen desgaste por que las escobillas están en contacto con el eje y son ineficientes al menos para este proyecto. En contraparte los sin escobillas (brushless) son más eficientes ya que no hacen contacto con el eje pero tiene un coste más alto.

Dadas estas condiciones se eligieron los motores A2212/13T 1000kv ya que se acomoda a nuestro sistema mecánico y nos permite colocar hélices de 10”

Figura 7. Motor Brushless. (Fuente propia)

3.2.4. Batería

A la hora de seleccionar la batería tuvimos en cuenta principalmente tres cosas:

- ❏ la intensidad de la batería
- ❏ el voltaje de la batería
- ❏ la capacidad de la batería

Se eligió usar baterías lipo por que tienen la capacidad de suministrar mucha energía en poco tiempo y son ligeras en comparación a otras típicas, se escogió de la marca HRB de 3000mAh de tres celdas (11.1v)

Figura 8. Batería Lipo. (Fuente propia)

Tabla 1. Descripción de la batería Lipo

n° de celdas	3
Dimensiones	26x34x106mm
Capacidad	3000 mAh
Peso	190 g

3.2.5. ESC (Electronic Speed Controller)

El ESC es el circuito que se encarga de generar una señal trifásica que alimenta al motor. La velocidad de giro se varía mediante una señal suministrada de pulsos PWM.

El ESC seleccionado es un ESC de 30A con fin de sobredimensionar la corriente que consume cada motor, este ESC permite ajustar el rango del PWM a un mínimo 1ms a un máximo de 2ms

Figura 9. ESC: Electronic Speed Controller. 30A (Fuente propia)

3.3. Configuración final

Para tener en cuenta una visión global del sistema se ha hecho un esquema donde se puede ver a grandes rasgos como es el conexionado del sistema. Tanto como los ESC así como el mpu 6050 y el módulo bluetooth

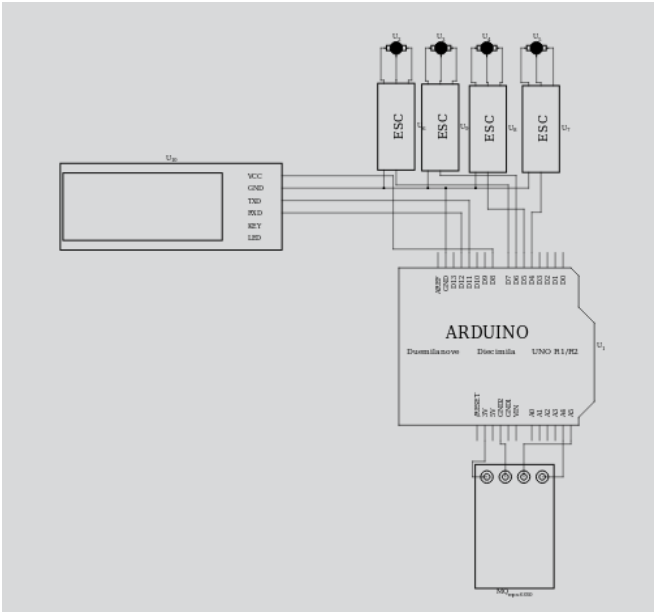


Figura 10. Esquema del armado final (Fuente propia)

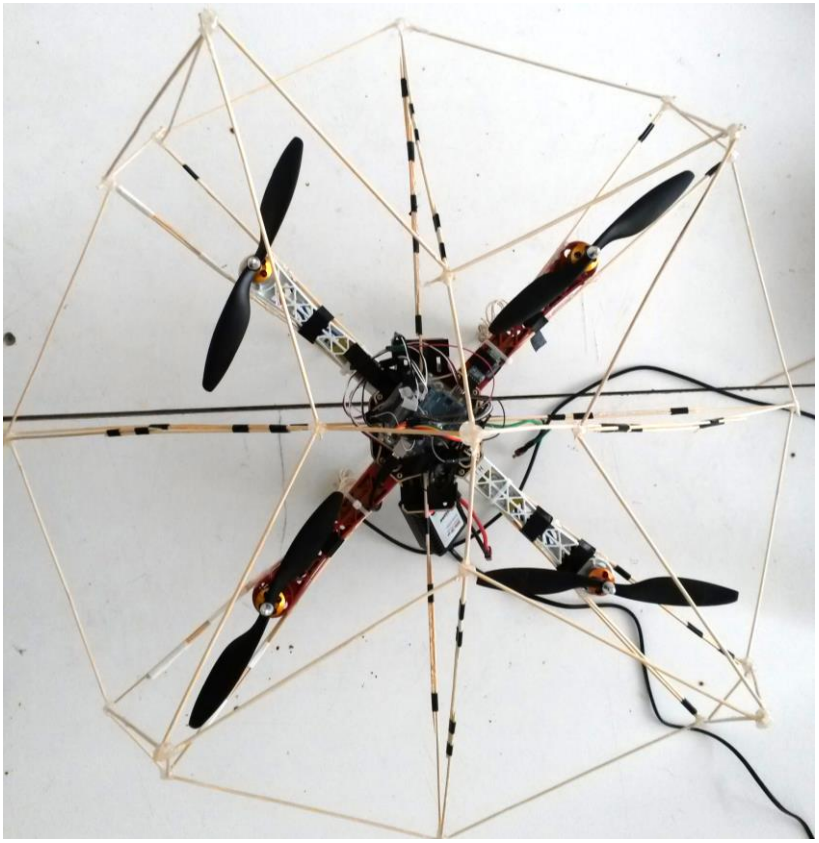


Figura 11. Armado final (Fuente propia)

3.4. Modelo Matemático

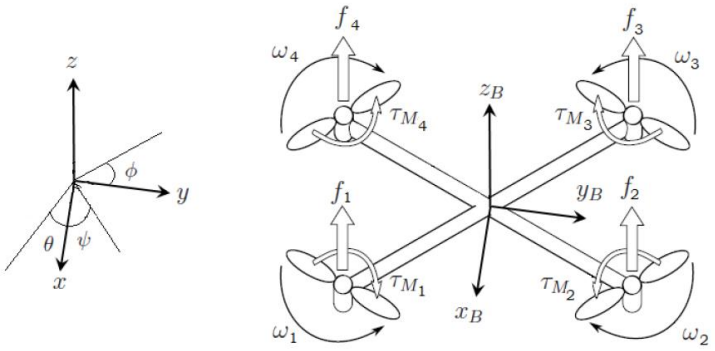


Figura 12. Esquema de referencia para el modelado matemático

3.4.1. Matriz de rotación

a) Roll (ϕ)

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

b) Pitch (θ)

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

c) Yaw (ψ)

$$R_z(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Para obtener la matriz de rotación en los tres ejes, se multiplicara las tres matrices mostradas anteriormente, donde se considerar primeramente el giro en z (yaw), luego en y (pitch) y finalmente en x (roll).

$$R = R_x(\phi) * R_y(\theta) * R_z(\psi)$$

$$R = \begin{bmatrix} \cos\psi\cos\theta & -\sin\psi\cos\theta & \sin\theta \\ \cos\psi\sin\theta\sin\phi + \sin\psi\cos\phi & \cos\psi\cos\phi - \sin\psi\sin\theta\sin\phi & -\cos\theta\sin\phi \\ \sin\psi\sin\phi - \cos\psi\sin\theta\cos\phi & \sin\psi\sin\theta\cos\phi + \cos\psi\sin\phi & \cos\theta\cos\phi \end{bmatrix}$$

3.4.2. Lagrangiano

Para determinar la posición del dron se utilizaran seis parámetros, las coordenadas en cada eje (x, y, z) y los tres ángulos formados (ϕ, θ, ψ)

$$\xi = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

$$\eta = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}$$

$$q = \begin{bmatrix} \xi \\ \eta \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \psi \end{bmatrix}$$

Para las velocidades lineales se utilizara:

$$\dot{\xi} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}$$

Dado que la velocidad angular tiene una dirección arbitraria después de cada movimiento, se hallara cada una por medio de la suma de las tres componentes, siendo así:

$$\vec{\omega} = \vec{\psi} + \vec{\theta} + \vec{\phi}$$

Considerando el grafico anterior, se puede hallar las velocidades angulares como:

$$\omega_\phi = \dot{\phi} - \dot{\psi}\sin\theta$$

$$\omega_\theta = \dot{\theta}\cos\phi + \dot{\psi}\cos\theta\sin\phi$$

$$\omega_\psi = \dot{\psi}\cos\theta\cos\phi - \dot{\theta}\sin\phi$$

Que será expresado como:

$$\omega = W_\eta\dot{\eta} = \begin{bmatrix}1 & 0 & -\sin\theta \\ 0 & \cos\phi & \cos\theta\sin\phi \\ 0 & -\sin\phi & \cos\theta\cos\phi\end{bmatrix}\begin{bmatrix}\dot{\phi} \\ \dot{\theta} \\ \dot{\psi}\end{bmatrix}$$

Despejando las velocidades angulares:

$$\dot{\phi} = \omega_\phi + \left(\omega_\theta\sin\phi + \omega_\psi\cos\phi\right)\tan\theta$$

$$\dot{\theta} = \omega_\theta\cos\phi - \omega_\psi\sin\phi$$

$$\dot{\psi} = \left(\omega_\theta\sin\phi + \omega_\psi\cos\phi\right)\sec\theta$$

La fuerza de empuje producida por cada rotor es:

$$f_i = k_i {w_i}^2$$

Donde k_i es la constante de sustentación

Y el par generado es:

$$\tau_j = b_j {\omega_j}^2$$

Donde b_j es la constante de arrastre. Para los pares generados por los rotores se tiene

$$\tau = \begin{bmatrix}\tau_\phi \\ \tau_\theta \\ \tau_\psi\end{bmatrix} = \begin{bmatrix}d(k_2\omega_2^2 + k_3\omega_3^2 - k_1\omega_1^2 - k_4\omega_4^2) \\ d(k_1\omega_1^2 + k_2\omega_2^2 - k_3\omega_3^2 - k_4\omega_4^2) \\ b_2\omega_2^2 + b_4\omega_4^2 - b_1\omega_1^2 - b_3\omega_3^2\end{bmatrix}$$

Donde $d = l\ \cos 45°$ y l es la longitud de los brazos al centro de masa.

Dadas las ecuaciones anteriores, se puede empezar el análisis del lagrangiano para lo cual las energías son:

$$T = \frac{1}{2}m\dot{\xi}^T\dot{\xi} + \frac{1}{2}\omega^T I\omega$$

$$V = mgz$$

Donde m es la masa total, e I es igual a:

$$I = \begin{bmatrix}I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz}\end{bmatrix}$$

Reordenando la energía cinética rotacional:

$$T_{rot} = \frac{1}{2}\omega^T I\omega = \frac{1}{2}(W_n\dot{\eta})^T I(W_n\dot{\eta}) = \frac{1}{2}\dot{\eta}^T(W_n^T I W_n)\dot{\eta}$$

Por tanto se tiene:

$$L = \frac{1}{2}m\dot{\xi}^T\dot{\xi} + \frac{1}{2}\dot{\eta}^T(W_n^T I W_n)\dot{\eta} - mg[0 \quad 0 \quad 1]\xi$$

Desarrollando:

$$L = \frac{1}{2}m(\dot{x}^2 + \dot{y}^2 + \dot{z}^2)$$
$$+ \frac{1}{2}\left(I_{yy}\cos^2\phi\,\dot{\theta}^2 + I_{zz}\sin^2\phi\,\dot{\theta}^2 + I_{xx}\dot{\phi}^2 + \left(I_{yy} - I_{zz}\right)\cos\theta\sin 2\phi\,\dot{\theta}\dot{\psi}\right.$$
$$\left. - 2I_{xx}\sin\theta\,\dot{\phi}\dot{\psi} + \left(I_{zz}\cos^2\theta\cos^2\phi + I_{xx}\sin^2\theta + I_{yy}\cos^2\theta\sin^2\phi\right)\dot{\psi}^2\right)$$
$$- mgz$$

3.4.3. Euler - Lagrange

Para la ecuación de Euler Lagrange:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = Q_i \qquad i = 1,2,3,4,5,6$$

Donde Q es igual a:

$$Q = \begin{bmatrix} f \\ \tau \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \\ f_z \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix}$$

Para f igual a:

$$f = \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} = RE = \begin{bmatrix} \sin\theta\,(k_1{\omega_1}^2 + k_2{\omega_2}^2 + k_3{\omega_3}^2 + k_4{\omega_4}^2) \\ -\cos\theta\sin\phi\,(k_1{\omega_1}^2 + k_2{\omega_2}^2 + k_3{\omega_3}^2 + k_4{\omega_4}^2) \\ \cos\theta\cos\phi\,(k_1{\omega_1}^2 + k_2{\omega_2}^2 + k_3{\omega_3}^2 + k_4{\omega_4}^2) \end{bmatrix}$$

Aplicando Euler-Lagrange con las condiciones dada, se obtiene:

$$m\ddot{x} = \sin\theta\,(k_1{\omega_1}^2 + k_2{\omega_2}^2 + k_3{\omega_3}^2 + k_4{\omega_4}^2)$$

$$m\ddot{y} = -\cos\theta\sin\phi\,(k_1{\omega_1}^2 + k_2{\omega_2}^2 + k_3{\omega_3}^2 + k_4{\omega_4}^2)$$

$$m\ddot{z} + mg = \cos\theta\cos\phi\,(k_1{\omega_1}^2 + k_2{\omega_2}^2 + k_3{\omega_3}^2 + k_4{\omega_4}^2)$$

$$\left(I_{yy} - I_{zz}\right)\cos\phi\sin\phi\,\dot{\theta}^2 - \cos\theta\left(I_{xx} + \left(I_{yy} - I_{zz}\right)\cos 2\phi\right)\dot{\theta}\dot{\psi}$$
$$- \left(I_{yy} - I_{zz}\right)\cos^2\theta\cos\phi\sin\phi\,\dot{\psi}^2 + I_{xx}\left(\ddot{\phi} - \sin\theta\,\ddot{\psi}\right)$$
$$= d(k_2\omega_2^2 + k_3\omega_3^2 - k_1\omega_1^2 - k_4\omega_4^2)$$

$$\left(I_{zz} - I_{yy}\right)\sin 2\phi\,\dot{\theta}\dot{\phi} + \cos\theta\left(I_{xx} + \left(I_{yy} - I_{zz}\right)\cos 2\phi\right)\dot{\phi}\dot{\psi}$$
$$+ \left(I_{zz}\cos\theta\cos^2\phi\sin\theta + I_{yy}\cos\theta\sin\theta\sin^2\phi - I_{xx}\cos\theta\sin\theta\right)\dot{\psi}^2$$
$$+ \left(I_{yy}\cos^2\phi + I_{zz}\sin^2\phi\right)\ddot{\theta} + \left(I_{yy} - I_{zz}\right)\cos\theta\cos\phi\sin\phi\,\ddot{\psi}$$
$$= d(k_1\omega_1^2 + k_2\omega_2^2 - k_3\omega_3^2 - k_4\omega_4^2)$$

$$\frac{1}{2}\left(\left(I_{zz} - I_{yy}\right)\sin\theta\sin 2\phi\,\dot{\theta}^2 + 2\left(I_{yy} - I_{zz}\right)\cos^2\theta\sin 2\phi\,\dot{\phi}\dot{\psi}\right.$$
$$- 2\dot{\theta}\left(\cos\theta\left(I_{xx} + \left(I_{zz} - I_{yy}\right)\cos 2\phi\right)\dot{\phi}\right.$$
$$\left. + \sin 2\theta\left(-I_{xx} + I_{zz}\cos^2\phi + I_{yy}\sin^2\phi\right)\dot{\psi}\right) + \left(I_{yy} - I_{zz}\right)\cos\theta\sin 2\phi\,\ddot{\theta}$$
$$\left. - 2I_{xx}\sin\theta\,\ddot{\phi} + \left(2I_{zz}\cos^2\theta\cos^2\phi + 2I_{xx}\sin^2\theta + 2I_{yy}\cos^2\theta\sin^2\phi\right)\ddot{\psi}\right)$$
$$= b_2\omega_2^2 + b_4\omega_4^2 - b_1\omega_1^2 - b_3\omega_3^2$$

Estas ecuaciones pueden ser representadas de la forma:

$$M(\vec{q})\ddot{\vec{q}} + C\left(\vec{q},\dot{\vec{q}}\right)\dot{\vec{q}} + D\left(\dot{\vec{q}}\right) + K(\vec{q}) + G(\vec{q}) = Q(t)$$

Donde:

$M(\vec{q}) = Matriz\ de\ Inercia$

$C(\vec{q}, \dot{\vec{q}}) = Matriz\ de\ Coriolis$

$D(\dot{\vec{q}}) = Vector\ de\ fuerzas\ disipativas$

$K(\vec{q}) = Vector\ de\ fuerzas\ elasticas$

$G(\vec{q}) = Vector\ de\ terminos\ gravitacionales$

$Q(t) = Vector\ de\ fuerzas\ generalizadas\ externas$

Siendo:

$$M(\vec{q}) = \begin{bmatrix} m & 0 & 0 & 0 & 0 & 0 \\ 0 & m & 0 & 0 & 0 & 0 \\ 0 & 0 & m & 0 & 0 & 0 \\ 0 & 0 & 0 & I_{xx} & 0 & -I_{xx}\sin\theta \\ 0 & 0 & 0 & 0 & I_{yy}\cos^2\phi + I_{zz}\sin^2\phi & \left(I_{zz} - I_{yy}\right)\cos\theta\cos\phi\sin\phi \\ 0 & 0 & 0 & -I_{xx}\sin\theta & \left(I_{yy} - I_{zz}\right)\cos\theta\cos\phi\sin\phi & \left(I_{xx} + I_{yy}\cos^2\theta\right)\sin^2\phi + I_{zz}\cos^2\phi\cos^2\theta \end{bmatrix}$$

La matriz $C(\vec{q}, \dot{\vec{q}})$ es de dimensión 6x6 cuyos elementos distintos de cero son:

$$C_{45} = \left(I_{yy} - I_{zz}\right)\left(\cos\phi\sin\phi\,\dot{\theta} + \cos\theta\sin^2\phi\,\dot{\psi}\right) + \left(\left(I_{zz} - I_{yy}\right)\cos\theta\cos^2\phi - I_{xx}\cos\theta\right)\dot{\psi}$$

$$C_{46} = \left(I_{zz} - I_{yy}\right)\cos^2\theta\cos\phi\sin\phi\,\dot{\psi}$$

$$C_{54} = \left(I_{zz} - I_{yy}\right)\left(\cos\phi\sin\phi\,\dot{\theta} + \cos\theta\sin^2\phi\,\dot{\psi}\right) + \left(I_{xx} + \left(I_{yy} - I_{zz}\right)\cos^2\phi\right)\cos\theta\,\dot{\psi}$$

$$C_{55} = \left(I_{zz} - I_{yy}\right)\cos\phi\sin\phi\,\dot{\phi}$$

$$C_{56} = \left(I_{zz}\cos^2\phi + I_{yy}\sin^2\phi - I_{xx}\right)\cos\theta\sin\theta\,\dot{\psi}$$

$$C_{64} = \left(I_{yy} - I_{zz}\right)\cos^2\theta\sin\phi\cos\phi\,\dot{\psi} - I_{xx}\cos\theta\,\dot{\theta}$$

$$C_{65} = \left(I_{zz} - I_{yy}\right)\left(\cos\phi\sin\phi\sin\theta\,\dot{\theta} + \cos\theta\sin^2\phi\,\dot{\phi}\right) + \left(I_{yy} - I_{zz}\right)\cos^2\phi\cos\theta\,\dot{\phi} + \left(I_{xx} - I_{zz}\cos^2\phi - I_{yy}\sin^2\phi\right)\sin\theta\cos\theta\,\dot{\psi}$$

$$C_{66} = \left(I_{xx} - I_{zz}\cos^2\phi - I_{yy}\sin^2\phi\right)\sin\theta\cos\theta\,\dot{\theta} + \left(I_{yy} - I_{zz}\right)\cos\phi\sin\phi\cos^2\theta\,\dot{\phi}$$

$$G(\vec{q}) = \begin{bmatrix} 0 \\ 0 \\ mg \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Las matrices $D(\dot{\vec{q}})$ de 6x1, $K(\vec{q})$ de 6x1 tienen ceros en todas sus casillas

3.5. Espacio de Estados

Dado:

$$x = \begin{bmatrix} \vec{q} \\ \dot{\vec{q}} \end{bmatrix}$$

La derivada está dada por:

$$\dot{x} = \begin{bmatrix} \dot{\vec{q}} \\ \ddot{\vec{q}} \end{bmatrix}$$

Reemplazando

$$\dot{x} = f(x,u) = \begin{bmatrix} \dot{\vec{q}} \\ M^{-1}(\vec{q})\left(Q(t) - C(\vec{q},\dot{\vec{q}})\dot{\vec{q}} - D(\dot{\vec{q}}) - K(\vec{q}) - G(\vec{q})\right) \end{bmatrix}$$

Se llevara el sistema presentado a la forma:

$$\dot{x} = Ax + Bu$$

$$y = Cx + Du$$

3.5.1. Linealizacion

Dado que la operación anterior, nos dará un modelo no lineal, se busca linealizar este sistema, para lo cual la posición de equilibrio será igual a:

$$x_{eq} = \begin{bmatrix} x \\ y \\ z \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Reemplazando estas condiciones en las ecuaciones de movimiento, se encuentra la velocidad angular a la cual el sistema estaría en equilibrio.

Tabla 2. Valores de las variables utilizadas en el modelado

Variable	Valor
g	$9{,}78\frac{m}{s^2}$
m	$1{,}1\ Kg$
d	$0{,}1301\ m$
I_{xx}	$1694.39 \times 10^{-6}\ Kg\ m^2$
I_{yy}	$1708.46 \times 10^{-6}\ Kg\ m^2$
I_{zz}	$2948.248 \times 10^{-6}\ Kg\ m^2$

k	$7{,}7 \times 10^{-7}\ N\ s^2$
b	$6{,}1 \times 10^{-8}\ N\ m\ s^2$
ω	$1868{,}91\ \frac{rad}{s}$

3.5.1.1. Matriz A

Para hallar la matriz A, se utilizara la ecuación:

$$A = \frac{\partial f(x,u)}{\partial x}$$

Para luego evaluar este resultado en el punto de equilibrio, con lo cual la matriz llega a ser:

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 9.7799 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -9.7799 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

3.5.1.2. Matriz B

Para hallar la matriz B, se utilizara la función:

$$B = \frac{\partial f(x,u)}{\partial u}$$

Reemplazando todos los valores, la matriz B es igual a:

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0.0026 & 0.0026 & 0.0026 & 0.0026 \\ -0.2210 & 0.2210 & 0.2210 & -0.2210 \\ 0.2192 & 0.2192 & -0.2192 & -0.2192 \\ -0.0773 & 0.0773 & -0.0773 & 0.0773 \end{bmatrix}$$

3.5.1.3. Matriz C

Para la matriz C se colocara el valor de 1 a las variables que se puedan observar, así C resulta ser:

$$C = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

3.5.1.4. Matriz D

Dado que no se consideran externas, la matriz D es igual a:

$$D = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

3.6. Controlabilidad y Observabilidad

3.6.1. Controlabilidad

Para verificar si el sistema era controlable, se utilizó la matriz A y B, en la plataforma Matlab se usó el comando:

$$n = rank(ctrb(A, B))$$

Donde el resultado fue igual a 12, que al ser igual al número de variables, se verifica que el sistema es controlable.

3.6.2. Observabilidad

Para verificar si el sistema era observable, en Matlab, se utilizó el comando

$$n = rank\big(obsv(A, C)\big)$$

Donde n tuvo el valor de 6, que es menor al número total de variables, por lo que el sistema es no observable.

3.7. Programación

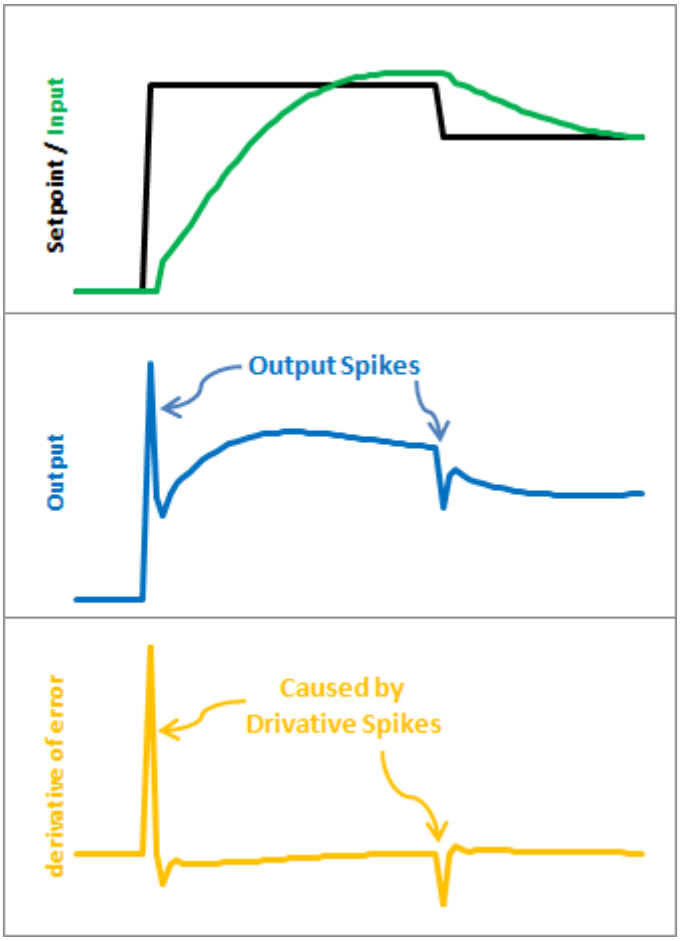
La elección de arduino como microprocesador implica el uso del IDE para el desarrollo. Tiene un gran abanico de librerías que simplifican enormemente cualquier tarea a realizar como lectura y escritura de puertos digitales y analógicos, implementación de los protocolos de comunicación como SPI, I2C y como también permite el control de timer y de los PWM

3.7.1. PID

Para el controlador de este sistema, se utilizó el PID (Proporcional, Integral, Derivativo), con algunas modificaciones, dado que para la parte integral se usó un constrain para delimitar la acumulación del error integral, esto para evitar el efecto Windup. Este se da cuando un gran cambio ocurre en el setpoint y la parte integral acumula un error significativo, y continua aumentando este error, provocando error.

La segunda modificación, se hizo por parte de la parte derivativa, para evitar el efecto conocido como derivative-kick, que produce cambios bruscos en la salida de los actuadores, como se muestra en la figura

Figura 13. Ilustración del efecto Derivative-kick



Para evitar el efecto, se utilizó:

$$\frac{dError}{dt} = \frac{dSetpoint}{dt} - \frac{dInput}{dt}$$

Así, cuando el setpoint es constante:

$$\frac{dError}{dt} = -\frac{dInput}{dt}$$

Por tanto, el PID utilizado llega a tener la forma:

```
float accel_roll(float error, float setpoint, float Kp, float Ki, float Kd){
    float err = setpoint - error;
    roll_ierror += err;
    roll_ierror = constrain(roll_ierror, -300, 300);
    double derror = error - roll_last_error;
    double out = Kp*err + roll_ierror*Ki - derror*Kd;
    out = constrain(out, -300, 300);
    roll_last_error = error;
    return out;
}
```

Figura 14. PID implementado

Donde Constrain es una función parte de la plataforma Arduino, donde se delimita el error integral, para el windup. Y el signo negativo de la parte derivativa, para eliminar el efecto derivative kick.

Dado que se usan tres ejes, se colocó dos PID por cada eje excepto en el yaw, teniendo un total de 5 PID, donde dos ejes cuentan con PID en cascada.

3.7.1.1. Roll y pitch

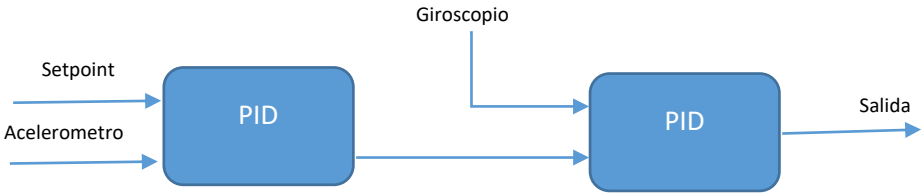


Figura 15. Esquema del PID en cascada

3.7.1.2. Yaw

Figura 16. PID implementado para yaw

3.7.2. Motores

3.7.2.1. Calibración de los motores

Para poder usar los motores, hay que tener claro el protocolo que usan los ESC para el inicio y calibrado.
Se conecta el ESC a la alimentación del sistema (normalmente mediante la conexión de las baterías). El ESC detecta el valor mínimo del PWM, y hace un largo sonido en forma de "beepppp". Entonces el sistema detecta voltaje de la batería y hace varios sonidos cortos en forma de "beep", lo que denota el número de células de la batería y

finalmente el ESC lleva a cabo la auto-comprobación. Si es normal, se oye una melodía "♪ 1 2 3".

3.7.2.2. Uso de los motores

Una vez los datos fueron procesados por el controlador, se aplican a los actuadores, en este caso, los motores, para ello, se sumaron cuatro valores: la potencia dada por el operador y los tres valores obtenidos por los PID, para cada eje. Los signos se hallaron por medio de la matriz B

También se aplicaron condicionales a las salidas de los motores, antes de aplicarlas, dado que si se enviaba un valor menor a 1050 (pulso mínimo para los motores) estos se hubieran detenido, además no se podía enviar un valor mayor a 2000, dado que era el pulso máximo admitido por los motores.

4. Resultados y conclusiones

Cabe destacar que el proyecto ha sido ambicioso y de gran dificultad técnica, el valor de lo aprendido en este proyecto es muy alto tener la experiencia de implementar un sistema de control, ya que sirvió para poner en práctica la capacidad para resolver casi cualquier tipo de problema.

La implementación de la estructura ha sido un éxito se logró hacerla lo suficientemente resistente ya que incorpora una protección elaborada por nosotros, dados los percances que se llevaron a cabo durante las pruebas.

En el aspecto de la programación del controlador hay que destacar que los parámetros encontrados son experimentales y varían respecto al lugar sobre el cual se lo quiere volar.

4.1. Mejoras

- Aunque al final se usó simplemente el puerto serial para la comunicación entre el drone y el arduino tiene la limitación de que es una conexión que nos limita a estar conectados si o si con el cable respectivo. Pero lo ideal es implementar un módulo bluetooth que nos quitará esta limitante o implementar wifi que es más rápido en los envíos de archivos en este proyecto no se contempló pero a futuro con más tiempo se podría implementar
- El control PID en este proyecto no es totalmente aplicable para todos los entornos, por tanto se debería aplicar un método que logre encontrar las constantes de manera autónoma.
- Dado que no se midieron varias variables, se debería considerar usar más sensores para la retroalimentación de las variables que no se tomaron en cuenta (las posiciones y velocidades lineales)
- Considerar factores externos al momento de comprar los motores, como la altura, ya que esto hace que los motores cierta cantidad de torque para levantar el peso que deberían.
- Trabajar con otro controlador en este caso Raspberry Pi dado que se planea trabajar con un sistema de control que se adapte tendríamos una limitante con el controlador actual es que nos quedaremos cortos con la capacidad de cálculo.

5. Calendario

Tabla 3. Calendario ejecutado

Nº	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin
1	Programada manualmente	Planificación del diseño del modelo del drone	1 sem	vie 22/8/17	mar 26/8/17
2	Programada manualmente	compra de los materiales necesarios para el armado del drone	2 días	jue 28/8/16	sab 30/8/16
3	Programada manualmente	verificación de compra y armado mecánico parcial del drone	3 días	mar 4/10/17	sab 7/10/17
4	Programada manualmente	primer modelado matemático del sistema aun sin considerar todas las variables	6 días	lun 9/10/17	sab 14/10/17
5	Programada manualmente	calibración de los ESC y sensores IMU	1 sem.	lun 16/10/17	lun 23/10/17
6	Programada manualmente	finalizado total del sistema mecánico	3 días	mar 24/10/17	vie 27/03/17
7	Programada manualmente	Remodelado Matemático	3 días	dom 29/10/2017	mar 31/10/17
8	Programada manualmente	programación básica del drone y primera prueba de vuelo	1 sem	lun 30/10/17	lun 6/11/17
9	Programada manualmente	programación final y pruebas de vuelo restantes	restantes hasta la presentación final	mar 7/11/17	hasta la presentación final

6. Anexos

6.1. Modelo matemático

modelo.m | codigo_base.m

```
%Constantes
g = 9.78;
m = 1.1;  %masa del dron en kg
l = 0.184;   %distancia de cada brazo al centro de masa
d = (sqrt(2)/2)*l;
%Inercias para cada eje
Ixx = 1694.39e-6;
Iyy = 1708.46e-6;
Izz = 2948.248e-6;
k = 7.7e-7; %
bm = 6.1e-8; %
syms x y z a b c xp yp zp ap bp cp xs ys zs as bs cs w1 w2 w3 w4 A B;
%Matriz de inercia
M = [m 0 0 0 0 0;
     0 m 0 0 0 0;
     0 0 m 0 0 0;
     0 0 0 Ixx 0 -Ixx*sin(b);
     0 0 0 0 Iyy*(cos(a))^2+Izz*(sin(a))^2 (Izz-Iyy)*cos(b)*cos(a)*sin(a);
     0 0 0 -Ixx*sin(b) (Iyy-Izz)*cos(b)*cos(a)*sin(a) (Ixx+Iyy*(cos(b))^2)*(sin(a))^2+Izz*(cos(a))^2*(cos(b))^2];
%Matriz de Coriolis
Co = [0 0 0 0 0 0;
      0 0 0 0 0 0;
      0 0 0 0 0 0;
      0 0 0 0 (Iyy-Izz)*(cos(a)*sin(a)*bp+cos(b)*(sin(a))^2*cp)+((Izz-Iyy)*cos(b)*(cos(a))^2-Ixx*cos(b))*cp (Izz-Iyy)*(cos(b))^
      0 0 0 (Izz-Iyy)*(cos(a)*sin(a)*bp+cos(b)*(sin(a))^2*cp)+(Ixx+(Iyy-Izz)*(cos(a))^2)*cos(b)*cp (Izz-Iyy)*cos(a)*sin(a)*ap (
      0 0 0 (Iyy-Izz)*(cos(b))^2*sin(a)*cos(a)*cp-Ixx*cos(b)*bp (Izz-Iyy)*(cos(a)*sin(a)*sin(b)*bp+cos(b)*(sin(a))^2*ap)+(Iyy-I
```

modelo.m | codigo_base.m

```
s(b))^2*cos(a)*sin(a)*cp;
)*ap (Izz*(cos(a))^2+Iyy*(sin(a))^2-Ixx)*cos(b)*sin(b)*cp;
(Iyy-Izz)*(cos(a))^2*cos(b)*ap+(Ixx-Izz*(cos(a))^2-Iyy*(sin(a))^2)*sin(b)*cos(b)*cp (Ixx-Izz*(cos(a))^2-Iyy*(sin(a))^2)*sin(b)*c
```

modelo.m | codigo_base.m

```
in(b)*cp;
)-Iyy*(sin(a))^2)*sin(b)*cos(b)*cp (Ixx-Izz*(cos(a))^2-Iyy*(sin(a))^2)*sin(b)*cos(b)*bp+(Iyy-Izz)*cos(a)*sin(a)*(cos(b))^2*ap];
% Vector de termino gravitacionales
G = [0;
     0;
     m*g;
     0;
     0;
     0];
% x,y,z son los desplazamientos en cada eje
% a,b,c son los angulos que se forman con cada eje
q = [x;
     y;
     z;
     a;
     b;
     c];
% primera derivada
qp = [xp;
      yp;
      zp;
      ap;
      bp;
      cp];
```

```
%Vector de fuerzas generalizadas externas
%Tres primeras filas fuerzas lineales, las ultimas fuerzas angulares
qt = [sin(b)*(k*w1^2+k*w2^2+k*w3^2+k*w4^2);
      -cos(b)*sin(a)*(k*w1^2+k*w2^2+k*w3^2+k*w4^2);
      cos(b)*cos(a)*(k*w1^2+k*w2^2+k*w3^2+k*w4^2);
      d*(k*w2^2+k*w3^2-k*w1^2-k*w4^2);
      d*(k*w1^2+k*w2^2-k*w3^2-k*w4^2);
      bm*w2^2+bm*w4^2-bm*w1^2-bm*w3^2];
f2 = inv(M);
f3 = f2*(qt-Co*qp-G);
%Segunda derivada despejada
xpf = [qp;
       f3];
Ad = jacobian(xpf, [x,y,z,a,b,c,xp,yp,zp,ap,bp,cp]);
Bd = jacobian(xpf, [w1,w2,w3,w4]);

%Substituyendo en el punto de equilibrio
%Donde los tres angulos y las tres velocidades tienen que ser igual a cero.
%Matriz A
A = subs(Ad,{a,b,c,ap,bp,cp,w1,w2,w3,w4},{0,0,0,0,0,0,1868.91,1868.91,1868.91,1868.91});
%Matriz B
B = subs(Bd,{a,b,c,ap,bp,cp,w1,w2,w3,w4},{0,0,0,0,0,0,1868.91,1868.91,1868.91,1868.91});

CC = ctrb(A,B);        % matriz de controlabilidad
n = rank(CC);   % Ver si es controlable

%Matriz C
C = [0 0 0 0 0 0 0 0 0 0 0 0;
     0 0 0 0 0 0 0 0 0 0 0 0;
     0 0 0 0 0 0 0 0 0 0 0 0;
     0 0 0 1 0 0 0 0 0 0 0 0;
     0 0 0 0 1 0 0 0 0 0 0 0;
     0 0 0 0 0 1 0 0 0 0 0 0;
     0 0 0 0 0 0 0 0 0 0 0 0;
     0 0 0 0 0 0 0 0 0 0 0 0;
     0 0 0 0 0 0 0 0 0 0 0 0;
     0 0 0 0 0 0 0 0 0 1 0 0;
     0 0 0 0 0 0 0 0 0 0 1 0;
     0 0 0 0 0 0 0 0 0 0 0 1];

%Matriz D
D = zeros(size(C,1),size(B,2));
```

6.2. Inicializacion de los ESC

```ino
#include<Servo.h>
// Crear objetos de la clase Servo para cada motor
Servo ESC1;
Servo ESC2;
Servo ESC3;
Servo ESC4;

int vel = 1000; //amplitud del pulso

void setup()
{
  //Asignar pines a cada ESC
  ESC1.attach(8);
  ESC2.attach(9);
  ESC3.attach(10);
  ESC4.attach(11);

  //Activacion de los ESC
  ESC1.writeMicroseconds(2000);
  ESC2.writeMicroseconds(2000);
  ESC3.writeMicroseconds(2000);
  ESC4.writeMicroseconds(2000);//1000 = 1ms
  //Cambia el 1000 anterior por 2000 si
  //tu ESC se activa con un pulso de 2ms
  delay(5000); //Esperar 5 segundos para hacer la activacion

  //Iniciar puerto serial
  Serial.begin(9600);
  Serial.setTimeout(10);
}

void loop()
{
  if(Serial.available() >= 1)
  {
    vel = Serial.parseInt(); //Leer un entero por serial
    if(vel != 0)
    {
      //Generar pulsos con los umeros recibidos
      ESC1.writeMicroseconds(vel);
      ESC2.writeMicroseconds(vel);
      ESC3.writeMicroseconds(vel);
      ESC4.writeMicroseconds(vel);
    }
  }
}
```

6.3. Programa principal

```
# include <Math.h>
# include <Servo.h>
#include "MPU6050.h"
#include <SoftwareSerial.h>
#include "I2Cdev.h"
#if I2CDEV_IMPLEMENTATION==I2CDEV_ARDUINO_WIRE
  #include "Wire.h"
#endif

MPU6050 mpu;

float pitchAccSet, rollAccSet;
float pitchGyroSet, rollGyroSet, yawSet;
double pitchAccSetAnt,rollAccSetAnt,yawSetAnt;
double pitchGyroSetAnt,rollGyroSetAnt;

float roll, pitch, yaw;
int ant_pot = 1050;
int pot = 1050;
//pulsos que se enviaran a los motores
int m1,m2,m3,m4;
// Variables para los PID
float pitchKp1 = 1;
float pitchKi1 = 0.02;
float pitchKd1 = 0.2;

float rollKp1 = 0.68;
float rollKi1 = 0.025;
float rollKd1 = 0.4;

float yawKp = 0.5;
float yawKi = 0.008;
float yawKd = 0.0;

float pitchKp2 = 1.3;
float pitchKi2 = 0.001;
float pitchKd2 = 0.5;

float rollKp2 = 1.0;
float rollKi2 = 0.001;
float rollKd2 = 0.2;

float accPitch_ierror = 0;
float accRoll_ierror = 0;
float gyroRoll_ierror = 0;
float gyroPitch_ierror = 0;
float yaw_ierror = 0;
float accPitch_last_error = 0 ;
float accRoll_last_error = 0;
float gyroPitch_last_error = 0;
float gyroRoll_last_error = 0;
float yaw_last_error = 0;
float pitchGyroAcc, rollGyroAcc;

//Variables para conseguir los datos del sensor MPU
int16_t ax,ay,az,gx,gy,gz;
float rollGyro,pitchGyro,yawGyro;
float accPitch,accRoll,accYaw;
float ciclo = 5;
float tiempoCiclo = ciclo/100000000;

//Variables para controlar la potencia de los motores
unsigned char comando;
```

```
      if (comando == 'b'){  // Menos altura
        Serial.println("Menos altura");
        pot-=50;
        pot = constrain(pot, 1050, 2000);
        ant_pot = pot;
        pitchAccSetAnt = 0.0;
        rollAccSetAnt = 0.0;
        pitchGyroSetAnt = 0.0;
        rollGyroSetAnt = 0.0;
        yawSetAnt = 0.0;
      }

    control();

    }
  }

void datosMPU(){
  mpu.getMotion6(&ax,&ay,&az,&gx,&gy,&gz);
  //De valores raw a grados
  float accYangle_raw = atan(((ax)/16384.0)/sqrt(pow(((ay)/16384.0),2)+pow(((az)/16384.0),2)))*180/3.1416;
  float accXangle_raw = atan(((ay)/16384.0)/sqrt(pow(((ax)/16384.0),2)+pow(((az)/16384.0),2)))*180/3.1416;

  pitchGyro = ((gx)/16.4);
  rollGyro = ((-1)*(gy)/16.4);
  yawGyro = ((-1)*(gz)/16.4);

  accPitch=(0.1*accXangle_raw)+(0.9*(accPitch+(pitchGyro*tiempoCiclo)));
  accRoll = (0.1*accYangle_raw)+(0.9*(accRoll+(rollGyro*tiempoCiclo)));
}

void control(){
  // Primera linea de PID
  pitchGyroAcc = accel_pitch(accPitch,pitchAccSet,pitchKp1,pitchKi1,pitchKd1);
  rollGyroAcc = accel_roll(accRoll,rollAccSet,rollKp1,rollKi1,rollKd1);
  //Segunda linea de PID
  roll = gyro_roll(rollGyro,rollGyroAcc,rollKp2,rollKi2,rollKd2);
  pitch = gyro_pitch(pitchGyro,pitchGyroAcc,pitchKp2,pitchKi2,pitchKd2);
  //Unico PID para yaw
  yaw = yaw_PID(yawGyro, yawSet, yawKp, yawKi, yawKd);
  //Pulsos para los motores
  m1 = pot + pitch + roll + yaw;
  m2 = pot + pitch - roll - yaw;
  m3 = pot - pitch - roll + yaw;
  m4 = pot - pitch + roll - yaw;
  //Condicionales para evitar el apagado de los motores
  if (m1<1050) m1 = 1050;
  if (m2<1050) m2 = 1050;
  if (m3<1050) m3 = 1050;
  if (m4<1050) m4 = 1050;
  //Condicionales para el limite de los motores
  if (m1>2000) m1 = 2000;
  if (m2>2000) m2 = 2000;
  if (m3>2000) m3 = 2000;
  if (m4>2000) m4 = 2000;

  motores(m1, m2, m3, m4);
}
  //primer PID de pitch con datos del acelerometro
  float accel_pitch(float error, float setpoint, float Kp, float Ki, float Kd){
    float err = setpoint - error;
    accPitch_ierror += err;
    accPitch_ierror = constrain(accPitch_ierror, -300, 300);
    double derror = error - accPitch_last_error;
    double out = Kp*err-0.04 + accPitch_ierror*Ki - derror*Kd;
    out = constrain(out, -300, 300);
    accPitch_last_error = error;
    return out;
  }
  //primer PID de roll con datos del acelerometro
  float accel_roll(float error, float setpoint, float Kp, float Ki, float Kd){
    float err = setpoint - error;
    accRoll_ierror += err;
    accRoll_ierror = constrain(accRoll_ierror, -300, 300);
    double derror = error - accRoll_last_error;
    double out = Kp*err + 0.02 + accRoll_ierror*Ki - derror*Kd;
    out = constrain(out, -300, 300);
    accRoll_last_error = error;
    return out;
  }
```

```
//segundo PID de pitch con datos del giroscopio y el primer PID
float gyro_pitch(float error, float setpoint, float Kp, float Ki, float Kd){
  float err = setpoint - error;
  gyroPitch_ierror += err;
  gyroPitch_ierror = constrain(gyroPitch_ierror, -300, 300);
  double derror = error - gyroPitch_last_error;
  double out = Kp*err + gyroPitch_ierror*Ki - derror*Kd;
  out = constrain(out, -300, 300);
  gyroPitch_last_error = error;
  return out;
}
//segundo PID de roll con datos del giroscopio y el primer PID
float gyro_roll(float error, float setpoint, float Kp, float Ki, float Kd){
  float err = setpoint - error;
  gyroRoll_ierror += err;
  gyroRoll_ierror = constrain(gyroRoll_ierror, -300, 300);
  double derror = error - gyroRoll_last_error;
  double out = Kp*err + gyroRoll_ierror*Ki - derror*Kd;
  out = constrain(out, -300, 300);
  gyroRoll_last_error = error;
  return out;
}
//PID de yaw
float yaw_PID(float error, float setpoint, float Kp, float Ki, float Kd){
  float err = setpoint - error;
  yaw_ierror += err;
  yaw_ierror = constrain(yaw_ierror, -250, 250);
  double derror = error - yaw_last_error;
  double out = Kp*err + 0.15 + yaw_ierror*Ki - derror*Kd;
  out = constrain(out, -250, 250);
  yaw_last_error = error;
  return out;
}

void iniciar_motores(){
  motor1.attach(9);
  motor2.attach(6);
  motor3.attach(5);
  motor4.attach(3);

  motor1.writeMicroseconds(1000);
  motor2.writeMicroseconds(1000);
  motor3.writeMicroseconds(1000);
  motor4.writeMicroseconds(1000);
}

void mover_motores(){
  //Comenzar el movimiento de los motores, con la potencia mas baja
  motor1.writeMicroseconds(1050);
  motor2.writeMicroseconds(1050);
  motor3.writeMicroseconds(1050);
  motor4.writeMicroseconds(1050);
}

void motores(int m1, int m2, int m3, int m4){
  /*Serial.print("Motor 1");Serial.println(m1);
  Serial.print("Motor 2");Serial.println(m2);
  Serial.print("Motor 3");Serial.println(m3);
  Serial.print("Motor 4");Serial.println(m4); */
  motor1.writeMicroseconds(m1);
  motor2.writeMicroseconds(m2);
  motor3.writeMicroseconds(m3);
  motor4.writeMicroseconds(m4);
}

```

7. Referencias

[1] Fuente *http://www.xdrones.es/tipos-de-drones-clasificacion-de-drones-categorias-de-drones/*

[2] Fuente *https://www.researchgate.net/publication/270163654_Mathematical_Modelling_and_Parameter_Identification_of_Quadrotor_a_survey*

[3] *Fuente https://commons.wikimedia.org/wiki/File:Quadrotorhover.svg?uselang=es*

Baguena F. (2016, Septiembre). Diseño y Control de un Cuadricoptero controlado por Bluetooth vía Android App. *Escuela Técnica Superior de Ingeniería del Diseño*

Fernández, A., Torres I., Ramírez U. (2016). Diseño, construcción y control de una aeronave tipo drone. *Universidad Nacional Autónoma de México*

Beauregar B. (2011, Abril). Improving the Beginner's – Derivative Kick. *Project Blog*. Recuperado de http://brettbeauregard.com/blog/2011/04/improving-the-beginner%E2%80%99s-pid-derivative-kick/

Wikipedia. Integral windup. Recuperado de https://en.wikipedia.org/wiki/Integral_windup

K. Shirriff y P. Badger, Secrets of Arduino PWM. https://www.arduino.cc/en/Tutorial/SecretsOfArduinoPWM