Diseño, modelado y control de plataforma experimental bi-rotor con características VTOL

Jorge Manuel Arizaga León

Director de tesis: Dr. José Rafael Benito Noriega Luna

Co-director de tesis: Dr. Luis Arturo García Delgado

Departamento de Investigación en Física
Universidad de Sonora

Tesis que el autor presenta para obtener el grado de
Maestro en Ciencias en Electrónica
Posgrado en Electrónica Enero de 2018

Universidad de Sonora

Repositorio Institucional UNISON

Esta tesis está dedicada a mi familia, en especial a la memoria de mi querida madre ...

Declaración

Declaro que, salvo que se haga referencia específica al trabajo de otras personas, los contenidos de esta tesis son originales y no han sido presentados en su totalidad o en parte para la obtención de este o cualquier otro grado, en ésta o en cualquier otra universidad. Esta tesis es trabajo propio y no contiene nada que haya sido resultado del trabajo realizado en colaboración con otras personas, excepto como se especifica en el texto y en la sección de agradecimientos. Esta tesis contiene menos de 65,000 palabras incluyendo apéndices, bibliografía, notas de pie de página, tablas y ecuaciones y tiene menos de 150 figuras.

Jorge Manuel Arizaga León
Enero de 2018

Agradecimientos

El agradecimiento va para mi apreciable familia, que me ha demostrado que la base del progreso siempre será el conocimiento y su aplicación. A mis profesores y asesores por ser guías en la búsqueda de este conocimiento. Además, quiero reconocer al Consejo Nacional de Ciencia y Tecnología (CONACyT), al Instituto Tecnológico y de Estudios Superiores de Monterrey, y a la Universidad de Sonora por todo el apoyo brindado a lo largo de esta maestría.

Resumen

El siguiente texto expone una propuesta de diseño estructural, modelado matemático y control de una plataforma experimental bi-rotor, la cual se utilizará para investigación en cursos de control a nivel superior.

Dicha plataforma se define como un sistema con características de despegue y aterrizaje vertical. Dicho sistema está conformado por un par de rotores (motor-hélice), una computadora central, batería y un diseño mecánicamente restringido; elementos que harán posible controlar la elevación, y los ángulos de alabeo y guiñada en el estudio y aplicación de controladores para Vehículos Aéreos No Tripulados.

Índice general

Índice de figuras

Nomenclatura

Acrónimos / Abreviaturas

ST Control Super Twisting

ESC Controlador Electrónico de Velocidad (Electronic Speed Controller)

SMC Control por Modos Deslizantes (Sliding Mode Controller)

STA Control Super Twisting Adaptativo

VANT Vehículo Aéreo No Tripulado

VTOL Despegue y Aterrizaje Vertical (Vertical Take Off and Landing)

Capítulo 1

Introducción

Recientemente los avances teóricos y tecnológicos que se han logrado en el área de control y automatización de sistemas han contribuido al rápido desarrollo en áreas como la de los Vehículos Aéreos No Tripulados (VANTs) [17].

Un VANT, debido a su dinámica áltamente acoplada, su tamaño reducido y su costo relativamente bajo, es una plataforma ideal para evaluar técnicas de control sofisticadas [9], [15]. Por ello, el modelado matemático y el control de un VANT son temas de amplio interés, conduciendo a los investigadores a aplicar conocimientos avanzados y por ende, obtener resultados que puedan ser utilizados tanto en aplicaciones industriales, como en cursos de control automático a nivel superior.

La principal desventaja que presentan los VANTs a nivel académico, es que al ser robots aéreos móviles, éstos son susceptibles a sufrir daños catastróficos al manipularlos [10]. Una caída puede llegar a inhabilitar un VANT, e incluso ocasionar lesiones a sus operadores. Por lo tanto, en cuestiones didácticas y de investigación, surge la necesidad de contar con plataformas que puedan ser controladas de forma segura para el operador, y que estén mecánicamente restringidas, con el objetivo de reducir el riesgo de colisiones accidentales [16].

Existen en la actualidad herramientas que permiten la exploración y aplicación de diversas técnicas de control para la dinámica de vuelo y sustentación para VANTs, sin embargo, estas plataformas cumplen solo parcialmente con los requerimientos necesarios para el estudio abordado en esta tesis.

De aquí es donde surge la propuesta de una plataforma bi-rotor mecánicamente restringida, en la que se pudiera controlar la elevación, el ángulo de alabeo y el ángulo de guiñada.

1.1. Análisis del estado del arte

Se realizó una búsqueda en distintas bases de datos, con los términos *Vertical Take Off and Landing* (VTOL), *Vehículo Aéreo No Tripulado* (VANT), *Unmaned Aerial Vehicle* (UAV), para evaluar las herramientas actuales utilizadas en la experimentación de dinámica de vuelo para VANTS. Adicionalmente, se investigaron los métodos de control utilizados en estas plataformas. Se consultaron además los sitios web de algunos fabricantes de herramientas para investigación académica en el área de control automático.

Como resultado, para la dinámica de vuelo en VANTs, se encontraron plataformas y sistemas que se caracterizan por lo siguiente:

1.- Plataformas bi-rotor de 3 grados de libertad: elevación, cabeceo y recorrido [6] y [20].

2.- Plataformas bi-rotor de motores con libertad de inclinación [17] y [13].

3.- Plataformas cuadri-rotor de 3 grados de libertad: alabeo, cabeceo y guiñada [4].

4.- Plataformas bi-rotor de 2 grados de libertad: Helicóptero con restricción de desplazamiento [11], [23] y [1].

5.- Vehículos VTOL cuadri-rotor [3].

1.2. Objetivos de la Investigación

Modelar, diseñar y construir una plataforma experimental bi-rotor mecánicamente restringida como herramienta de investigación para evaluación de controladores matemáticos y dinámica de vuelo de VANTs.

1.3. Hipótesis

Es posible la fabricación de una plataforma experimental bi-rotor mecánicamente restringida y la aplicación de controladores matemáticos no lineales para 3 grados de libertad.

1.4. Descripción de la contribución

La primer contribución de esta investigación es el desarrollo de una plataforma bi-rotor (modelado matemático, diseño mecánico e integración de componentes electrónicos y de

programación), restringida en movimientos de desplazamiento y cabeceo, teniendo tres grados de libertad: elevación, alabeo y guiñada. La segunda contribución es la exploración y aplicación de distintos algoritmos de control.

Con ello, se están cubriendo las características requeridas para realizar los estudios previamente planeados.

Capítulo 2

Descripción del sistema y modelado

En este capítulo se describe el sistema bi-rotor, tanto en diseño y componentes, como en comportamiento dinámico. En la sección 2.1 se muestran los componentes mecánicos propuestos mediante un software de diseño asistido por computadora (CAD). Posteriormente, en la sección 2.2 se expone la integración electromecánica de componentes utilizados en la experimentación. En la sección 2.3 se presenta el modelado matemático propuesto para el sistema bi-rotor. Como sección 2.4 se tiene la descripción del sistema de propulsión de la plataforma bi-rotor y las pruebas experimentales realizadas para su caracterización.

2.1. Diseño CAD

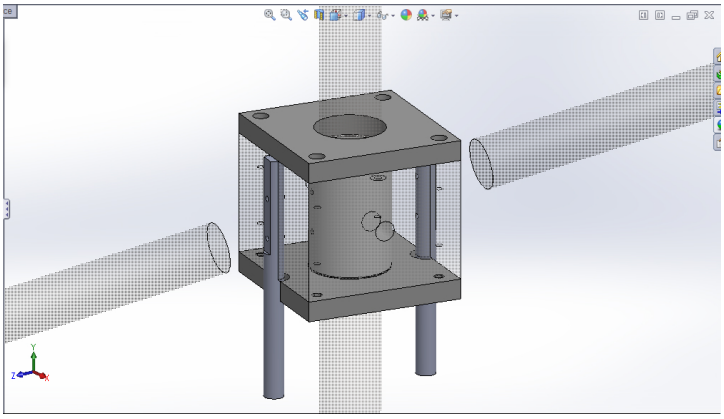

Figura 2.1 Cubo central bi-Rotor

Para poder construir la plataforma se requirió realizar un diseño asistido por computadora (CAD) en SolidWorks. Este diseño se conformó en primera instancia por un conjunto de 3 piezas de aluminio, un buje de nylon y dos barras de acero. Una de las piezas de aluminio corresponde al cubo central de la parte móvil del bi-rotor. Dentro de este cubo central se diseñó un alojamiento para el buje de nylon. Dos tapas de aluminio son las que funcionan para retener el buje, tal como se muestra en la Figura 2.1. Las dos barras de acero están posicionadas para servir como limitadores de movimiento. La función de las barras se detallará en una sección posterior. La unión de las piezas se logra con tornillos, cuyas medidas tanto de longitud como de cuerda de rosca fueron seleccionadas para garantizar una correcta sujeción entre elementos, sin añadir estrés al aluminio y peso no necesario a la plataforma.

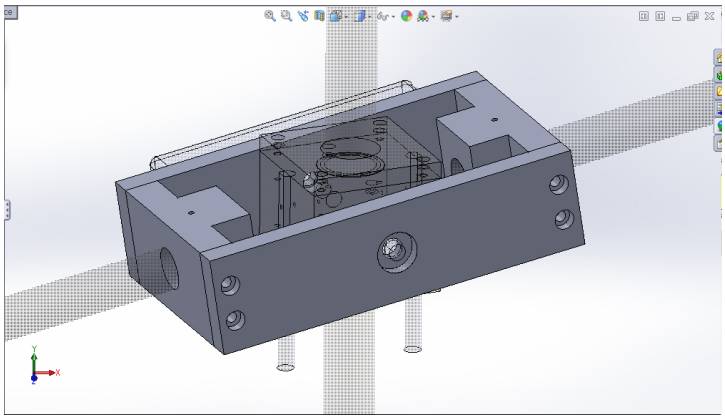

Figura 2.2 Marco externo bi-rotor

Se diseñó la estructura externa en aluminio, que al igual que el cubo central, es parte fundamental del bi-rotor. Esta estructura está conformada por 4 piezas individuales, diseñadas en pares simétricos, unidas mediante tornillos (Se muestra mayor detalle en la Figura 2.2). Sobre este marco externo van montados los brazos del bi rotor, la batería, los sensores y la computadora de control.

Para acoplar mecánicamente las dos estructuras anteriormente descritas (Cubo central y marco externo) y garantizar que exista la menor cantidad de fricción asociada al movimiento, se utilizaron dos vástagos de acero y dos baleros metálicos. Obsérvese el detalle del acoplamiento en la Figura 2.3.

El cubo central cuenta con dos perforaciones en dos de sus lados, en donde se introducen y se fijan los vástagos con apoyo de tornillos opresores. De manera concéntrica a las perforaciones en el cubo central, las caras de mayor longitud de la estructura externa

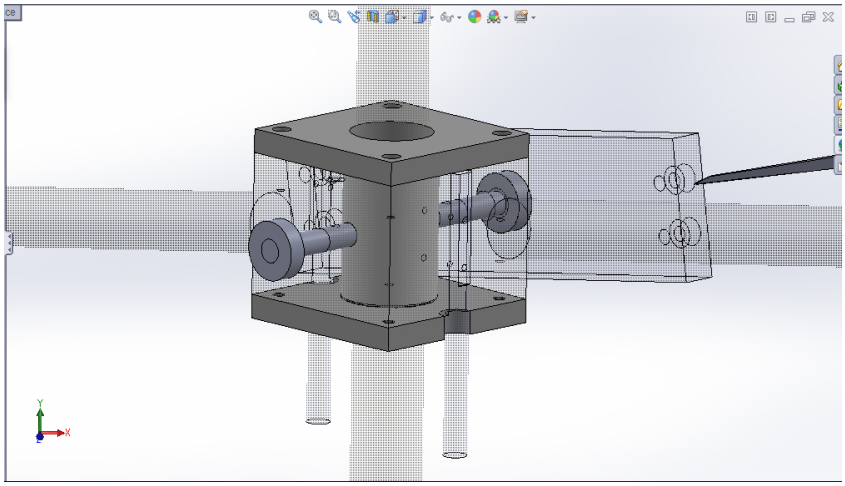

Figura 2.3 Acoplamiento vástago-balero

cuentan con perforaciones y cajas para la colocación de baleros, los cuales son sujetados también con apoyo de opresores.

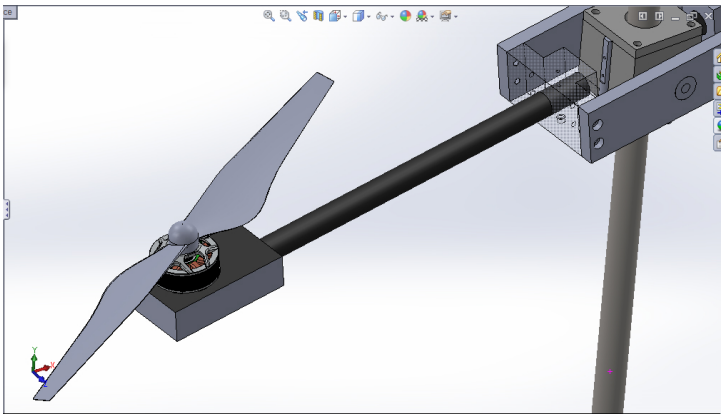

Figura 2.4 Brazo con rotor

Refiriéndonos a las dos caras de menores dimensiones de la estructura externa, se aprecia en el diseño que están perforadas para permitir la sujeción de los brazos del bi-rotor. Cada brazo fue diseñado como un tubo de fibra de carbono en el cual está montada una base. Sobre la base en el extremo del brazo se encuentra sujeto cada motor con su respectiva hélice. El detalle se aprecia en la Figura 2.4.

La estructura móvil del sistema bi-rotor se diseñó para que pudiese desplazarse verticalmente sobre un vástago de acero cromado. Para garantizar la mínima vibración

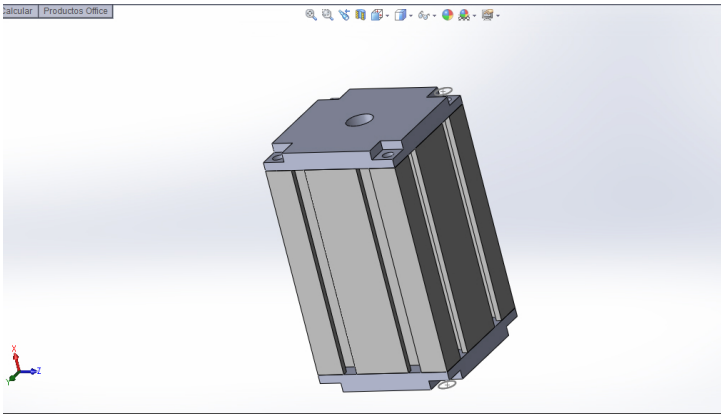

Figura 2.5 Perfil con tapas

del vástago, se propuso una base con una tabla de madera. Sobre la tabla de madera se colocó un perfil de aluminio, sujeto a la madera con apoyo de 4 consolas.

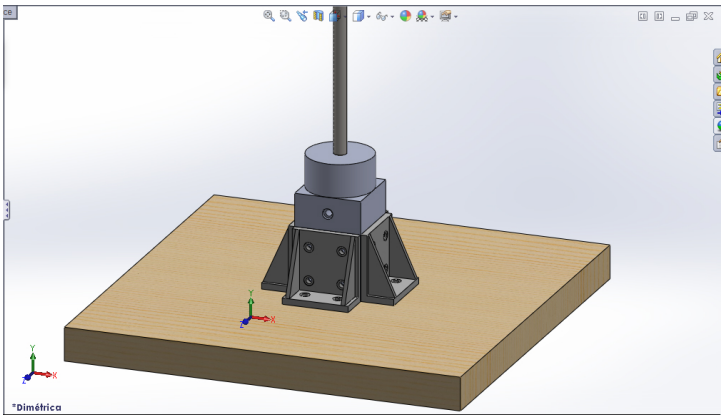

Figura 2.6 Base

El perfil, al ser una estructura vacía por dentro, requirió del diseño de 2 tapas de aluminio, con un orificio central concéntrico sobre el cual pasara el vástago. El detalle del diseño de esta pieza se muestra en la Figura 2.5, mientras que en la Figura 2.6 se observa la manera en que la base metálica está fija a la tabla con apoyo de las consolas.

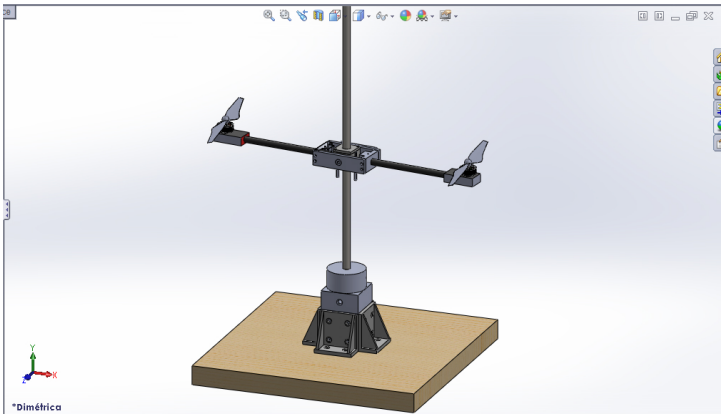

Figura 2.7 BiRotor

El diseño del sistema completo se puede apreciar en la Figura 2.7. La libertad de movimiento del diseño propuesto se ilustra en la Figura 2.8, Figura 2.9 y Figura 2.10.

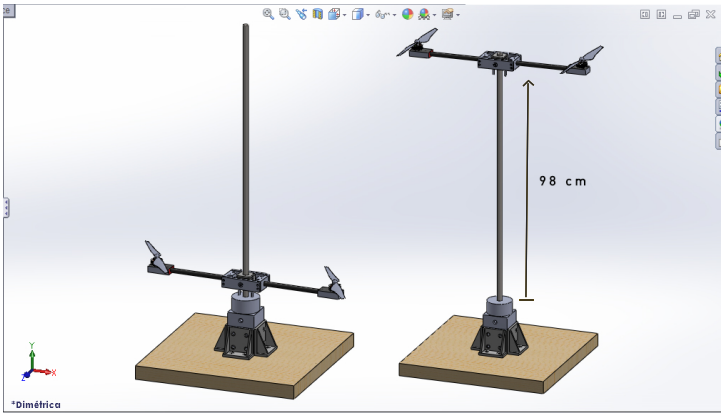

Figura 2.8 Desplazamiento en elevación

Con el diseño mecánico propuesto, se realizó el maquinado y la unión de las piezas. Se describe en la sección 2.2 los dispositivos que se integraron para poner en función la plataforma.

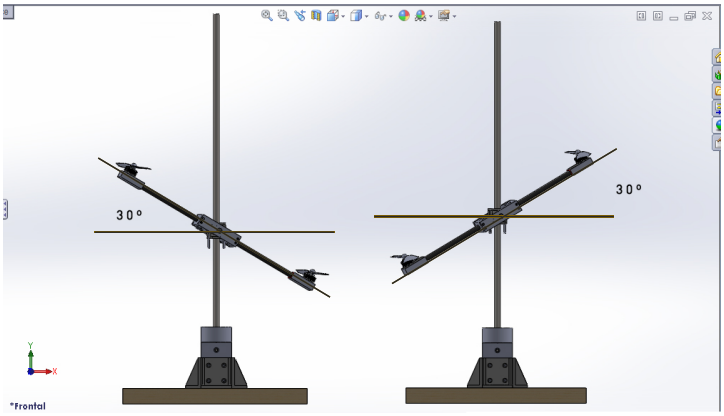

Figura 2.9 Ángulo de alabeo

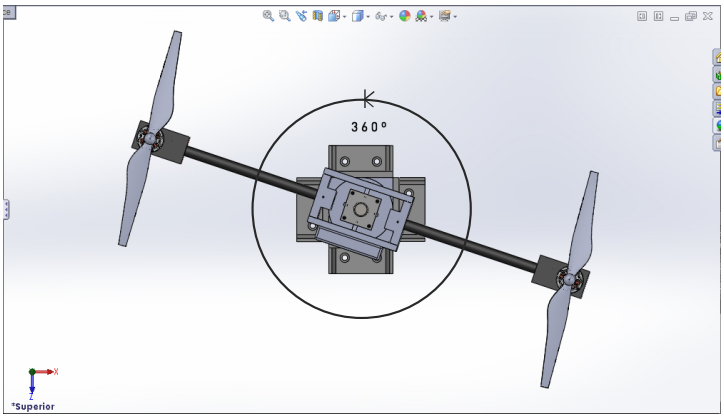

Figura 2.10 Ángulo de guiñada

2.2. Descripción del sistema electromecánico

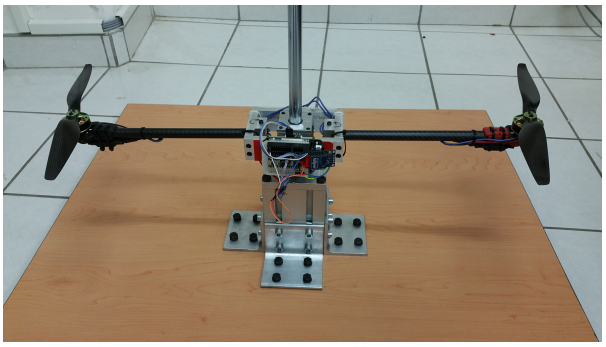

Figura 2.11 Plataforma Bi-Rotor

En la Figura 2.11 se puede observar la plataforma en su implementación real.

El sistema electrónico integrado en esta plataforma está constituido por los siguientes componentes:

- 2 Controladores ESC 30A.
- 2 Motores Brushless Turnigy Multistar 3508 700kv.
- 1 Batería Li-Po 4s 14.8 Volts 5000mAh.
- 1 Arduino Mega.
- 1 Modulo de comunicación WiFi ESP8266.
- 5 marcadores ópticos para sistema Optitrack.
- Cables y conectores eléctricos.

2.3. Modelo Matemático

Se debe de considerar el sistema bi-rotor de la Figura 2.12.

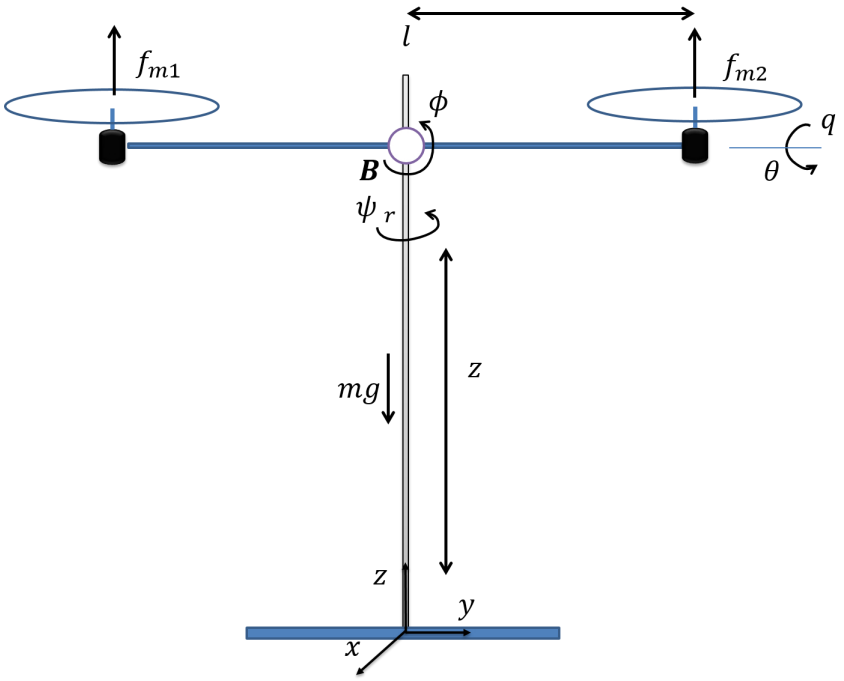

Figura 2.12 Esquema modelo bi-rotor

Las coordenadas generalizadas son

$$\mathbf{q} = \begin{bmatrix} z \\ \phi \\ \psi \end{bmatrix} \tag{2.1}$$

donde z indica la elevación del sistema, ϕ indica el ángulo de alabeo (*roll*) y ψ es el ángulo de guiñada (*yaw*).

Las velocidades angulares en el marco del cuerpo están dadas por

$$\omega = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.2}$$

donde $q \equiv 0$ dado que no se tiene variación en el ángulo de cabeceo (*pitch*).

Para modelar el sistema, se debe obtener la energía potencial y cinética del robot. La energía potencial del sistema depende de la masa y de la altitud del bi-rotor y está dada por

$$u = mgz \tag{2.3}$$

Para modelar las dinámicas rotacionales, se definen las matrices básicas de rotación que transforman la dirección de un marco móvil con respecto al inercial [9] y [22] . La rotación de un ángulo ψ sobre el eje z está dada por

$$\mathbf{R}_{z,\psi} = \begin{bmatrix} C_\psi & S_\psi & 0 \\ -S_\psi & C_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

mientras que la rotación de un ángulo ϕ con respecto al eje x está dada mediante

$$\mathbf{R}_{x,\phi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & C_\phi & S_\phi \\ 0 & -S_\phi & C_\phi \end{bmatrix} \tag{2.5}$$

en donde C y S representan las funciones sin() y cos(), respectivamente.

La matriz de rotación compuesta está definida por

$$\mathbf{R} = \mathbf{R}_{x,\phi}\mathbf{R}_{z,\psi} = \begin{bmatrix} C_\psi & S\psi & 0 \\ -S_\psi C_\phi & C_\psi C\phi & S_\phi \\ S_\psi S_\phi & -C_\psi S_\phi & C_\phi \end{bmatrix} \tag{2.6}$$

La velocidad angular en el marco del cuerpo definida en la ecuación (2.2), se relaciona con el vector de velocidades en el marco fijo $\dot{\eta}$ mediante el Jacobiano de velocidad $\mathbf{W}$ de la siguiente forma

$$\omega = \mathbf{W}\dot{\eta} = \mathbf{W} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{2.7}$$

donde $\dot{\eta}$ denota el vector de velocidades angulares en el marco inercial y la matriz $\mathbf{W}$ está definida por

$$\mathbf{W} = \begin{bmatrix} 1 & 0 & -S_\theta \\ 0 & C_\phi & C_\theta S_\phi \\ 0 & -S_\phi & C_\theta C_\phi \end{bmatrix} \tag{2.8}$$

Como $\theta \equiv 0$ y $\dot{\theta} \equiv 0$, debido a que no hay movimiento de cabeceo, se tendrá

$$\mathbf{W} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & C_\phi & S_\phi \\ 0 & -S_\phi & C_\phi \end{bmatrix} \tag{2.9}$$

Debido a la misma restricción en movimiento, sólo se puede girar sobre los ejes x y z, se elimina el segundo renglón y la segunda columna de $\mathbf{W}$, lo cual reduce $\mathbf{W}$ a dos coordenadas, lo que resulta en

$$\mathbf{W} = \begin{bmatrix} 1 & 0 \\ 0 & C_\phi \end{bmatrix} \tag{2.10}$$

La matriz de inercias en el marco del cuerpo será

$$\mathbf{I} = \begin{bmatrix} I_x & 0 \\ 0 & I_z \end{bmatrix} \tag{2.11}$$

La matriz de inercias en el marco inercial corresponderá a

$$\mathbf{J} = \mathbf{W}^T \mathbf{I} \mathbf{W} = \begin{bmatrix} I_x & 0 \\ 0 & I_z C_\phi^2 \end{bmatrix} \tag{2.12}$$

La matriz de masas-inercias será entonces

$$\mathbf{M}(\mathbf{q}) = \begin{bmatrix} m & 0 & 0 \\ 0 & I_x & 0 \\ 0 & 0 & I_z C_\phi^2 \end{bmatrix} \tag{2.13}$$

La forma general del modelo dinámico de robots [22] es:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) + \mathbf{f}(\dot{\mathbf{q}}) = \tau \tag{2.14}$$

donde $\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}}$ es la matriz de masas e inercias, $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}}$ es el vector de fuerzas centrífugas y de Coriollis, $\mathbf{g}(\mathbf{q})$ es el vector de gravedad, $\mathbf{f}(\dot{\mathbf{q}})$ es el vector de fricción y τ corresponde al vector de torques de control.

Para este sistema el vector de Coriollis $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}}$ se obtiene a partir de la matriz de masas e inercias y está definido como

$$\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} = \dot{\mathbf{M}}(\mathbf{q})\dot{\mathbf{q}} - \frac{1}{2}\frac{\partial}{\partial \mathbf{q}}\left[\dot{\mathbf{q}}^T\mathbf{M}(\mathbf{q})\dot{\mathbf{q}}\right] \tag{2.15}$$

$$= \begin{bmatrix} 0 \\ I_z C_\phi S_\phi \dot{\psi}^2 \\ -2I_z C_\phi S_\phi \dot{\phi}\dot{\psi} \end{bmatrix} \tag{2.16}$$

El vector de gravedad es

$$\mathbf{g}(\mathbf{q}) = \begin{bmatrix} mg \\ 0 \\ 0 \end{bmatrix} \tag{2.17}$$

El vector de fricción $\mathbf{f}(\dot{\mathbf{q}})$ corresponde a

$$\mathbf{f}(\dot{\mathbf{q}}) = \begin{bmatrix} K_{f1}\dot{z} \\ K_{f2}\dot{\phi} \\ K_{f3}\dot{\psi} \end{bmatrix} = \begin{bmatrix} K_{f1} & 0 & 0 \\ 0 & K_{f2} & 0 \\ 0 & 0 & K_{f3} \end{bmatrix} \dot{\mathbf{q}} \tag{2.18}$$

El vector de entrada del sistema τ indica las fuerzas que ejercen los actuadores (motores) sobre el sistema bi-rotor, y está definido como

$$\tau = \begin{bmatrix} F_z \\ \tau_\phi \\ \tau_\psi \end{bmatrix} \tag{2.19}$$

donde la fuerza de elevación se expresa como

$$F_z = u\cos(\phi) \tag{2.20}$$

u es el empuje de los rotores

$$u = f_{m1} + f_{m2} \tag{2.21}$$

y los torques de entrada se definen como

$$\tau_\phi = (f_{m1} - f_{m2})l \tag{2.22}$$

$$\tau_\psi = \tau_{m1} - \tau_{m2} \tag{2.23}$$

En la ecuación anterior τ_{m1} y τ_{m2} son los torques rotacionales debidos a los motores M_1 y M_2 respectivamente. Por lo tanto, el modelo del sistema en ecuaciones de estado será

$$\frac{d}{dt}\begin{bmatrix}\mathbf{q}\\ \dot{\mathbf{q}}\end{bmatrix} = \begin{bmatrix}\dot{\mathbf{q}}\\ \mathbf{M}^{-1}[\tau - \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} - \mathbf{g} - \mathbf{f}(\dot{\mathbf{q}})]\end{bmatrix} \tag{2.24}$$

Esta propuesta bi-rotor corresponde a un sistema subactuado, debido a que para los 3 grados de libertad, se tendrán únicamente 2 actuadores [17].

2.4. Descripción del sistema motor (rotor-hélice)

Para el propósito requerido de este proyecto se utilizó el procedimiento de caracterización reportado en [8] para determinar los parámetros del sistema rotor.

El modelo matemático del motor brushless puede ser aproximado mediante una ecuación diferencial de primer orden mediante pruebas experimentales, donde la velocidad está relacionada con la entrada de control descrita por

$$t_m\dot{\omega} = k_m u - \omega \tag{2.25}$$

donde k_m indica la ganancia de proceso de cada motor, t_m es la constante de proceso dada por la respuesta al escalón del sistema motor-hélice y u representa la entrada de control.

Para la hélice, consideramos T y Q como el torque y empuje respectivamente, que pueden ser expresados mediante

$$T = k_T\omega^2 \tag{2.26}$$

$$Q = k_Q\omega^2$$

donde k_T es la constante de empuje y k_Q es la constante de torque.

Para el estudio de nuestra plataforma, es necesario cumplir con los requerimientos de empuje para poder asegurar un funcionamiento correcto. Por ello, el empuje nominal de los rotores debe de satisfacer

$$\sum_{i=1}^{2} T_i = mg \tag{2.27}$$

donde T_i es el empuje total producido por los i-rotores, m es la masa total de la parte móvil del bi-rotor y g es la gravedad, por lo cual, se puede reescribir la ecuación (2.25) como:

$$T_i = k_{T_i}\omega_i^2 \tag{2.28}$$

donde k_{T_i} es el coeficiente de empuje y depende básicamente del perfil de la hélice y ω_i es la velocidad angular del i-rotor.

Relacionando las ecuaciones (2.27) y (2.28), la velocidad angular para el sistema de dos hélices, en estado de sustentación se puede calcular como:

$$\omega_h = \sqrt{mg/2k_T} \tag{2.29}$$

Por lo tanto, el sistema motor-hélice dependerá de dos factores: el primero es la capacidad de cada motor de producir la velocidad angular ω_h, y el segundo es una hélice con una constante k_T suficiente para obtener el empuje requerido.

Observación: Los requerimientos de empuje para un sistema bien dimensionado corresponden a un par de rotores cuyo empuje total sea de 1.5 veces el peso del sistema a sustentar [18].

2.4.1. Caracterización de los motores

En esta subsección se muestran las pruebas realizadas para obtener los coeficientes de empuje y torque para el par de motores que se están utilizando en la plataforma bi-rotor. Estos experimentos de caracterización de torque y empuje de motores fueron llevados a cabo en el Laboratorio Nacional de Robótica del área centro y noreste de México, ubicado en el Instituto Tecnológico y de Estudios Superiores de Monterrey, en la ciudad de Monterrey, México.

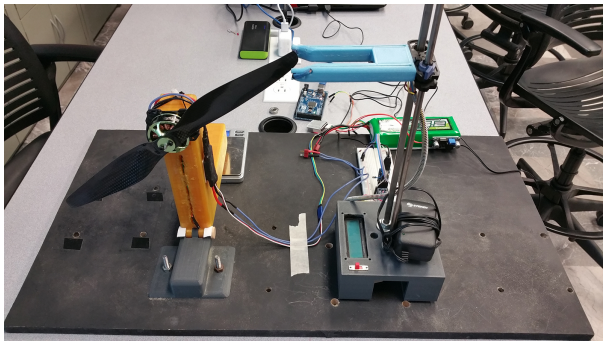

Figura 2.13 Sistema para medición de empuje

A. Empuje

El coeficiente de empuje k_T fue medido mediante una configuración palanca-báscula, la cual consiste en un sensor infrarrojo colocado a un costado de la hélice en rotación, midiendo así la velocidad angular. Al mismo tiempo, el motor está colocado en un extremo de la palanca, mientras que en el otro extremo equidistante al punto de apoyo, se coloca la báscula. Al rotar la hélice y generar el empuje, éste es transmitido en una relación 1:1 a la báscula. La configuración se observa en la Figura 2.13.

Se realizaron 4 pruebas distintas de 20 mediciones cada una(con voltajes de batería de 16.8v, 16.6v, 16.5v y 16.3v respectivamente) desde 0 a 100 % de velocidad, con aumentos

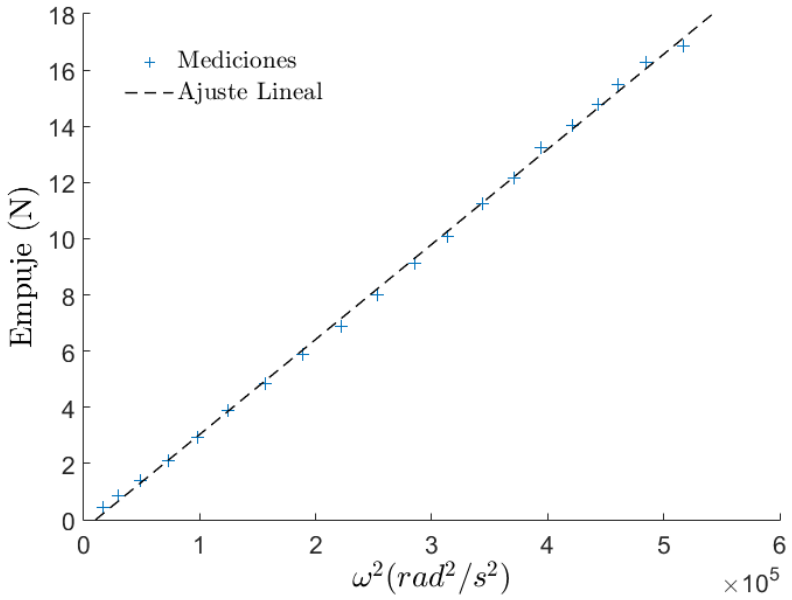

Figura 2.14 Banco de pruebas para el coeficiente de empuje (16.8 v)

de velocidad del 5 %. Con los resultados recabados para la prueba de 16.8v (en donde se presentó el empuje máximo con la batería cargada al 100 %), se obtuvo la gráfica de la Figura 2.14, en donde el coeficiente de empuje k_T se determina con la ecuación (2.26).

Se utilizó la herramienta de gráficos de Matlab para realizar un ajuste lineal de los resultados, y con la ecuación resultante del ajuste se obtuvo un coeficiente de empuje

$$k_T = 3.4 \times 10^{-5} \mathrm{N}/(\mathrm{rad}^2/\mathrm{s}^2) \tag{2.30}$$

Es pertinente resaltar que la aproximación lineal obtenida para el coeficiente de empuje k_T presenta una desviación en el eje (0,0), tal como se puede observar en la Figura 2.14. Esta desviación pudiera introducir un error al trabajar el sistema bi-rotor en lazo cerrado, sin embargo, por la magnitud del mismo y para los propósitos establecidos en este trabajo, este error se considera manejable, y por lo tanto se desprecia del análisis, sin embargo, la caracterización más precisa para el empuje es considerada un área de oportunidad como parte de un trabajo a futuro.

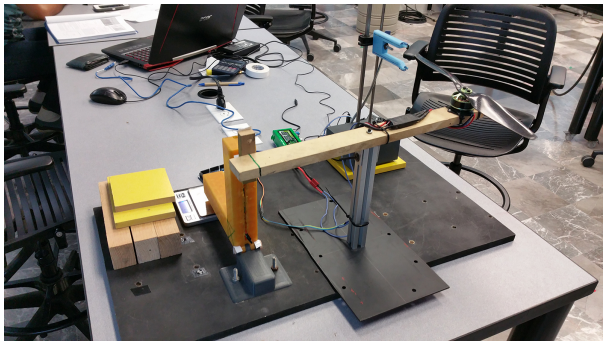

Figura 2.15 Sistema de medición de torque

B. Torque

Para obtener experimentalmente el torque, se adaptó el sistema palanca-báscula de la forma que se muestra en la Figura 2.15. El motor fue colocado en forma vertical sobre el extremo de un eje móvil de madera, con punto de apoyo equidistante a la palanca. Dado que el torque produce una rotación sobre el eje vertical, esta fuerza se ve reflejada sobre la báscula.

Siguiendo la misma metodología de pruebas que se utilizó para medir el empuje, se generó la gráfica de torque que se muestra en la Figura 2.16.

El ajuste lineal de los datos medidos dio como resultado un coeficiente de torque

$$k_Q = 3.4 \times 10^{-6} \mathrm{N \cdot m/(rad^2/s^2)} \tag{2.31}$$

La velocidad angular máxima para cada motor obtenida a través de los experimentos es de 719 rad/s, lo que significa que mediante la ecuación (2.29) se puede determinar que el par de rotores son capaces de sustentar una masa total de

$$m = 2k_T\omega_h^2/g = 3.59 \text{ Kg} \tag{2.32}$$

Dado que la masa del sistema es de 1.84 Kg, el par de rotores cumple con las condiciones de sustentación establecidas.

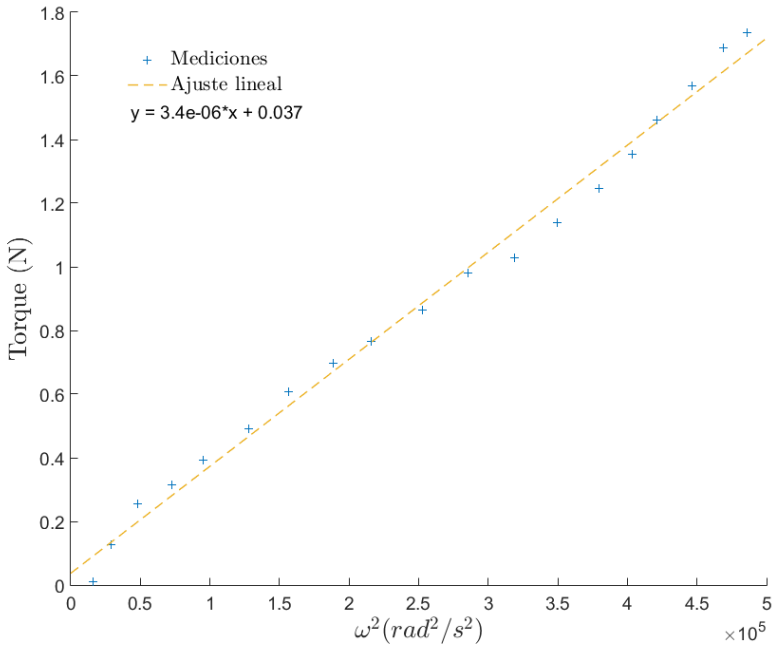

Figura 2.16 Banco de pruebas para el coeficiente de torque (16.8 v)

C. Respuesta al PWM de motor brushless

Para caracterizar el desempeño del sistema rotor, se realizó una prueba que consistió en la asignación secuencial de anchos de pulso de PWM al ESC del motor, con variaciones de $50\mu s$ entre sí, iniciando en $1000\mu s$ y terminando en $2100\mu s$. Los resultados se muestran en la Figura 2.17. El desempeño fue ajustado a una función lineal, cuya pendiente da como resultado una relación

$$\omega = 0.68D \tag{2.33}$$

siendo D el porcentaje del ciclo de trabajo.

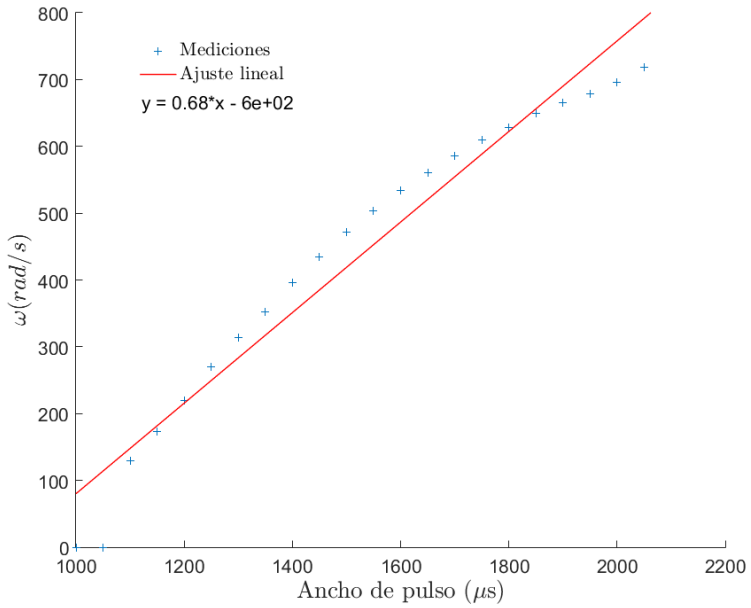

Figura 2.17 Desempeño del PWM

Así como se señaló en la caracterización del coeficiente de empuje, para determinar el desempeño de la señal PWM frente a la velocidad angular se utilizó una aproximación de primer orden. Observando la Figura 2.17 es posible notar que el gráfico del ajuste no cruza por el origen (1000,0) y que existe una desviación del ajuste respecto a los puntos de la medición, sin embargo, para los experimentos realizados en este trabajo de investigación estos errores fueron manejables. La caracterización más exacta del desempeño de la señal PWM se considera como área de oportunidad en trabajo a futuro.

Con las características del modelo matemático propuesto en este capítulo fue posible plantear cuatro algoritmos con cuatro distintos controladores. De cada uno de ellos se generó un reporte de simulación, los cuales se presentan en el capítulo 5.

A su vez, el estudio y la experimentación para obtener las características físicas del sistema rotor condujo a poder aplicar los controladores directamente sobre el sistema físico. La metodología y resultados se exponen en el capítulo 6.

Capítulo 3

Propuesta de Controladores

3.1. Control PD con compensación de gravedad

El control proporcional derivativo con compensación de gravedad es un aporte clásico de la teoría de control para la plataforma bi-rotor [12].

El controlador para este sistema está definido como

$$\tau = \mathbf{K_p q_e} + \begin{bmatrix} mg \\ 0 \\ 0 \end{bmatrix} - \mathbf{K_v \dot{q}} \tag{3.1}$$

en donde $\mathbf{K_p}$ es la matriz de ganancia proporcional, y está definida como

$$\mathbf{K_p} = \begin{bmatrix} K_{pz} & 0 & 0 \\ 0 & K_{p\phi} & 0 \\ 0 & 0 & K_{p\psi} \end{bmatrix} \tag{3.2}$$

El error de posición se denota por $\mathbf{q}_e$ y está definido como

$$\mathbf{q}_e = \mathbf{q}_d - \mathbf{q} \tag{3.3}$$

donde $\mathbf{q}_d$ es el vector con los valores de referencia. $\mathbf{K}_v$ corresponderá a la matriz de ganancia derivativa y se define como

$$\mathbf{K}_v = \begin{bmatrix} K_{vz} & 0 & 0 \\ 0 & K_{v\phi} & 0 \\ 0 & 0 & K_{v\psi} \end{bmatrix} \tag{3.4}$$

3.2. Control por modos deslizantes

Ën años recientes mucha de la investigación en el área de la teoría de control se ha enfocado en el diseño de retroalimentación discontinua, la cual alterna la estructura del sistema de acuerdo a la evolución del vector de estados". (Bartoszewicz, 2010, p.3772) [2].

La característica distintiva del control por modos deslizantes (SMC) es su robustez. En el control por modos deslizantes se diseña el control para forzar todas las trayectorias a que alcancen la variedad en tiempo finito y permanezcan en ella para todo tiempo futuro, a pesar de las perturbaciones que pudieran afectar el sistema [14]. El primer paso para desarrollar el SMC es definir la superficie deslizante.

Para el sistema bi-rotor, se retoma la ecuación (2.14) correspondiente al modelo general de robots. Se define la superficie deslizante para el modelo, como

$$\sigma = [\dot{\mathbf{q}} - \dot{\mathbf{q}}_\mathbf{d} + \lambda(\mathbf{q} - \mathbf{q}_\mathbf{d})] \tag{3.5}$$

Donde $\mathbf{q}$ es el vector de coordenadas generalizadas, $\mathbf{q_d}$ corresponde al vector de coordenadas deseadas, mientras que $\dot{\mathbf{q}}$ y $\dot{\mathbf{q}}_\mathbf{d}$ es la respectiva derivada de cada uno de ellos. El vector λ es un factor que controla la rapidez de convergencia a la superficie deslizante para cada grado de libertad [14] y [6].

Se tiene una condición fundamental para la superficie deslizante, la cual debe de cumplir con

$$\sigma = \dot{\sigma} = 0 \tag{3.6}$$

por lo tanto se requiere derivar σ

$$\dot{\sigma} = [\ddot{\mathbf{q}} - \ddot{\mathbf{q}}_\mathbf{d} + \lambda(\dot{\mathbf{q}} - \dot{\mathbf{q}}_\mathbf{d})] \tag{3.7}$$

Se despeja $\ddot{\mathbf{q}}$ de la ecuación (2.14)

$$\ddot{\mathbf{q}} = [\tau - \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} - \mathbf{g}(\mathbf{q}) - \mathbf{f}(\dot{\mathbf{q}})]\mathbf{M}(\mathbf{q})^{-1} \tag{3.8}$$

Se sustituye $\ddot{\mathbf{q}}$ de la ecuación (3.8) en la ecuación (3.7), resultando en

$$\dot{\sigma} = [\tau - \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} - \mathbf{g}(\mathbf{q}) - \mathbf{f}(\dot{\mathbf{q}})]\mathbf{M}(\mathbf{q})^{-1} - \ddot{\mathbf{q_d}} + \lambda(\dot{\mathbf{q}} - \dot{\mathbf{q_d}}) \tag{3.9}$$

En la ecuación (3.9) existe el vector τ, que para este sistema se considera la entrada. Se debe de proponer una entrada τ tal que $\ddot{\sigma} = 0$, lo que se logra con:

$$\tau = \mathbf{M}(\mathbf{q})[\ddot{\mathbf{q_d}} - \lambda(\dot{\mathbf{q}} - \dot{\mathbf{q_d}})] + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) + \mathbf{f}(\dot{\mathbf{q}}) + \mathbf{u_a} \tag{3.10}$$

donde $\mathbf{u_a}$ corresponde al control auxiliar y está definido por el control por modos deslizantes

$$\mathbf{u_a} = -\mathbf{K}\text{sign}(\sigma) \tag{3.11}$$

3.3. Control Super Twisting

Para el propósito de este trabajo de investigación es importante formular una ley de control que permita el correcto funcionamiento de los actuadores, que en este caso son los motores, ya que en las pruebas experimentales realizadas a los mismos, éstos llegan a presentar fallas ante cambios muy repentinos de velocidad.

La principal desventaja del controlador SMC clásico es el introducir vibraciones (*chattering*) en la señal de control. La técnica de los modos deslizantes de segundo orden incorpora derivadas de segundo orden en la variable deslizante. Esta inclusión conduce a tener las siguientes ventajas:

- Convergencia en tiempo finito.
- Reducción de vibraciones (*chattering*).
- Robustez ante perturbaciones [21] y [19].

Estas vibraciones (*chattering*) pueden llegar a ocasionar un mal funcionamiento en los motores, al conducir a variaciones muy marcadas en su velocidad para poder ejercer la acción de control.

Por tales motivos, se propone el controlador Super Twisting [19], en donde, a diferencia del controlador SMC clásico, la ley de control $\mathbf{u_a}$ corresponderá a

$$\mathbf{u_a} = -\mathbf{K_1}|\sigma|^{1/2}\text{sign}(\sigma) + \xi \tag{3.12}$$

$$\dot{\xi} = -\mathbf{K_2}\text{sign}(\sigma) \tag{3.13}$$

siendo las gananacias $\mathbf{K_1} > 0$ y $\mathbf{K_2} > 0$.

σ corresponde a la superficie deslizante planteada en la ecuación (3.5).

3.4. Control Super Twisting Adaptativo

El control supertwisting adaptativo propuesto para el bi-rotor sigue las condiciones de diseño descritas en [21] y [7].

Este algoritmo se presenta como una mejora del control super twisting y consiste en generar una función en la que las ganancias $\mathbf{K_1}$ y $\mathbf{K_2}$ de la ecuación (3.12) y de la ecuación (3.13) sean adaptables. Esto se logra con la siguiente propuesta de ley de control $\mathbf{u_a}$

$$\mathbf{u_a} = -\mathbf{K}(t)|\sigma|^{1/2}\text{sign}(\sigma) + \xi(t) \tag{3.14}$$

$$\dot{\xi} = -\mathbf{\Lambda}(t)\text{sign}(\sigma) \tag{3.15}$$

A diferencia del controlador super twisting donde se tiene $\mathbf{K_1}$ y $\mathbf{K_2}$, en este algoritmo se tendrán $\mathbf{K(t)}$ y $\mathbf{\Lambda}(t)$ como ganancias del controlador, y su evolución en el tiempo estará descrita mediante la ecuación (3.16) y la ecuación (3.17).

$$\dot{\mathbf{K}}(t) = \begin{cases} \mathbf{k}\text{sign}(|\sigma| - \mu) & \text{si} \quad \mathbf{K} > \mathbf{K_{min}} \\ \\ \mathbf{K_{min}} & \text{si} \quad \mathbf{K} \leq \mathbf{K_{min}} \end{cases} \tag{3.16}$$

$$\Lambda(t) = 2\epsilon\mathbf{K} \tag{3.17}$$

Este controlador ajusta las ganancias para reducir el esfuerzo de control, en donde $\mathbf{k}$ establece la tasa de adaptación, μ es el umbral para detección de la pérdida del

modo deslizante, que puede ser visto como: una vez que $|\sigma| \leq \mu$ se alcanza el dominio, las ganancias $\mathbf{K}(t)$, $\mathbf{\Lambda}(t)$ comienzan a reducir hasta que las trayectorias del sistema abandonen el dominio, entonces las ganancias comienzan a incrementar para forzar a las trayectorias a regresar al dominio. Finalmente, $\mathbf{K_{min}}$ es usado para asegurar valores diferentes a cero [5].

Esta técnica de control es robusta frente a perturbaciones e incertidumbres, con convergencia en tiempo finito. Además, sus ganancias permiten diseñar el controlador aun sin conocer los límites de las incertidumbres y las perturbaciones [21] [5].

Capítulo 4

Simulaciones

En este capítulo se exponen los parámetros de cada simulación, y se discuten lo resultados obtenidos de cada una de ellas.

En el cuadro 4.1 se muestran los parámetros físicos asociados al modelo bi-rotor.

Cuadro 4.1 Parámetros del modelo bi-rotor

Parámetro	Valor	Unidades
Masa	1.85	kg
Elevación máxima	.90	m
Inercia de roll	0.043	kgm^2
Inercia de yaw	0.041	kgm^2
kf1	0.01	
kf2	0.01	
kf3	0.01	

Cuadro 4.2 Valores de referencia

Grado de libertad	Valor	unidad
Elevación z	0.5	m
Alabeo ϕ	0	grados
Guiñada ψ	20	grados

En el cuadro 4.2 se muestran los valores de referencia para todas las simulaciones.

Para efectuar la simulación, se configuró Simulink con un método de resolución Runge-Kutta, con un tiempo fundamental de resolución de 10ms [5], [7] y [6]. La duración de la simulación se configuró a 10 segundos; tiempo apto para evaluar el comportamiento del controlador debido a que en 10 segundos, el sistema permite generar el estado transitorio y alcanzar el estado estable para la variable del proceso.

Se introdujo una perturbación en $t = 5s$, para poder evaluar la respuesta del controlador a esta misma perturbación.

4.1. Controlador PD con compensación de gravedad

Se sintonizó el controlador PD con compensación de gravedad con los parámetros del cuadro 4.3.

Cuadro 4.3 Parámetros del controlador PD con compensación de gravedad

Parámetro	Valor
Kp_z	50
Kd_z	18
Kp_ϕ	10
Kd_ϕ	1.8
Kp_ψ	10
Kd_ψ	1.8

En la Figura 4.1 se observa la evolución temporal de las variables de estado, mientras que en la Figura 4.2 se muestra la evolución temporal de las señales de control.

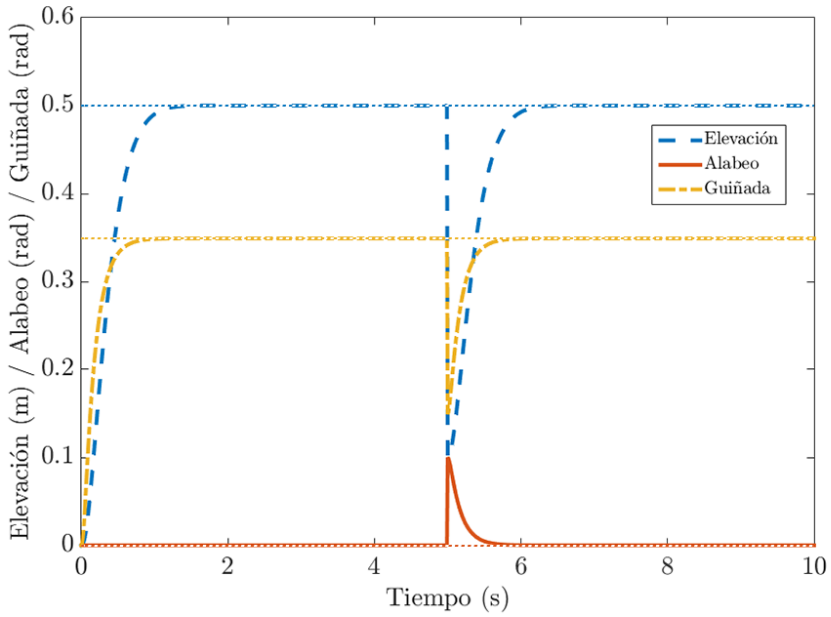

Figura 4.1 Evolución temporal de las variables de estado para el controlador PD con compensación de gravedad

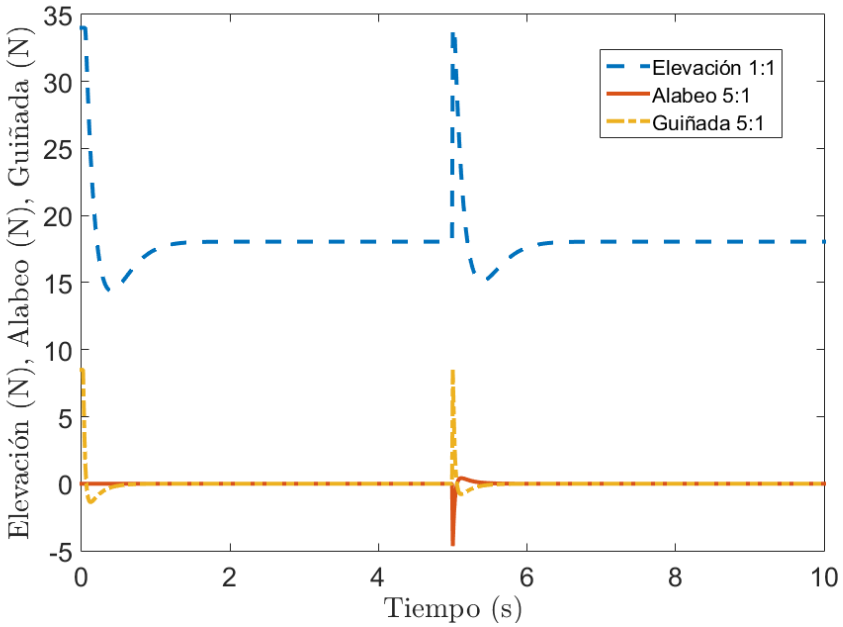

Figura 4.2 Evolución temporal de las señales de control para el controlador PD con compensación de gravedad

4.2. Controlador SMC

Se simuló el controlador SMC utilizando los parámetros del Cuadro 4.4

Cuadro 4.4 Parámetros del controlador SMC

Parámetro	Valor
K_z	10
λ_z	6
K_ϕ	1.7
λ_ϕ	10
K_ψ	1.2
λ_ψ	10

Las evolución temporal de las variables de estado, resultado de este controlador, se muestran en la Figura 4.3, mientras que en la Figura 4.4 se muestra la evolución temporal de las señales de control.

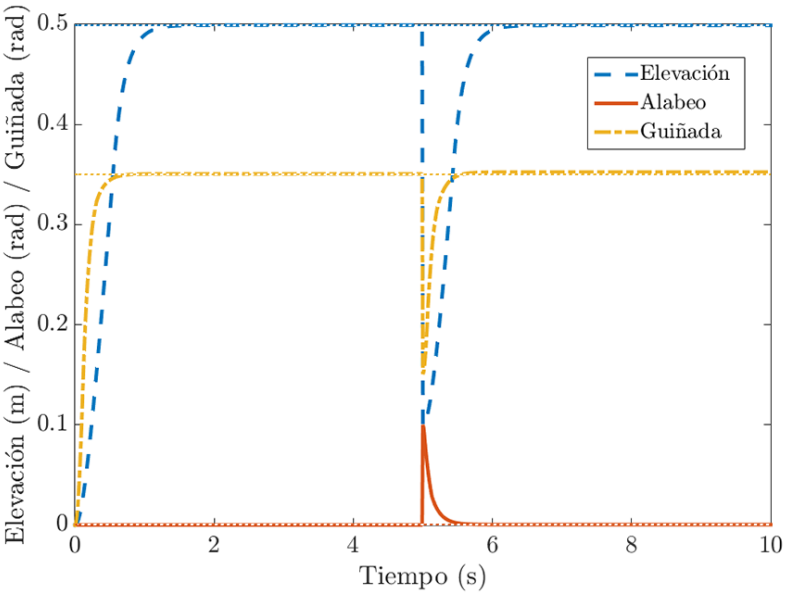

Figura 4.3 Evolución temporal de las variables de estado para el controlador SMC

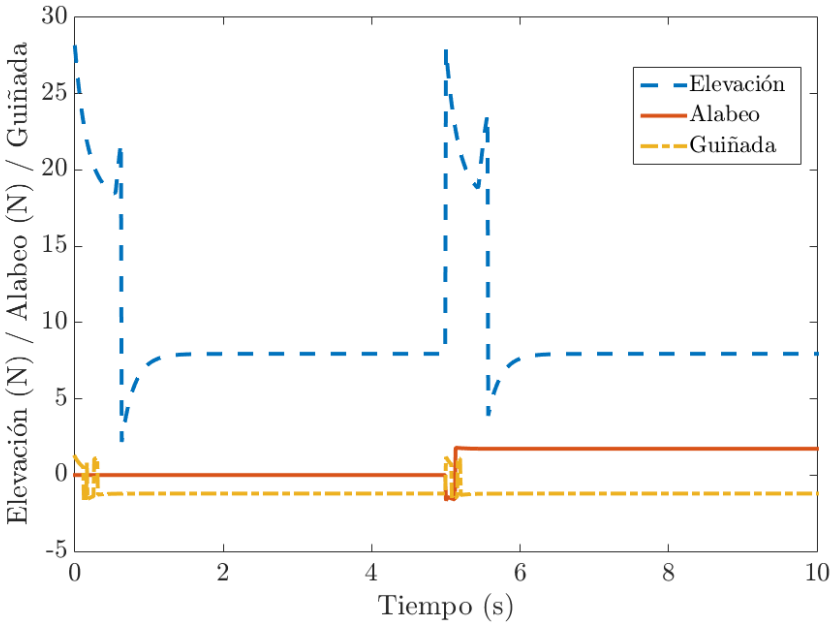

Figura 4.4 Evolución temporal de las señales de control para el controlador SMC

4.3. Controlador Super Twisting (ST)

Se simuló el controlador ST con los parámetros del cuadro 4.5. Los resultados de la simulación de este controlador se muestran en la Figura 4.5 y la Figura 4.6, siendo la Figura 4.5 la correspondiente a la evolución de las variables de estado, mientras que la Figura 4.6 muestra las señales de control para el controlador ST.

Cuadro 4.5 Parámetros del controlador ST

Parámetro	Valor
K_{1z}	15
K_{2z}	0.5
λ_z	5
$K_{1\phi}$	1.5
$K_{2\phi}$	0.006
λ_ϕ	5
$K_{1\psi}$	.5
$K_{2\psi}$	0.003
λ_ψ	5

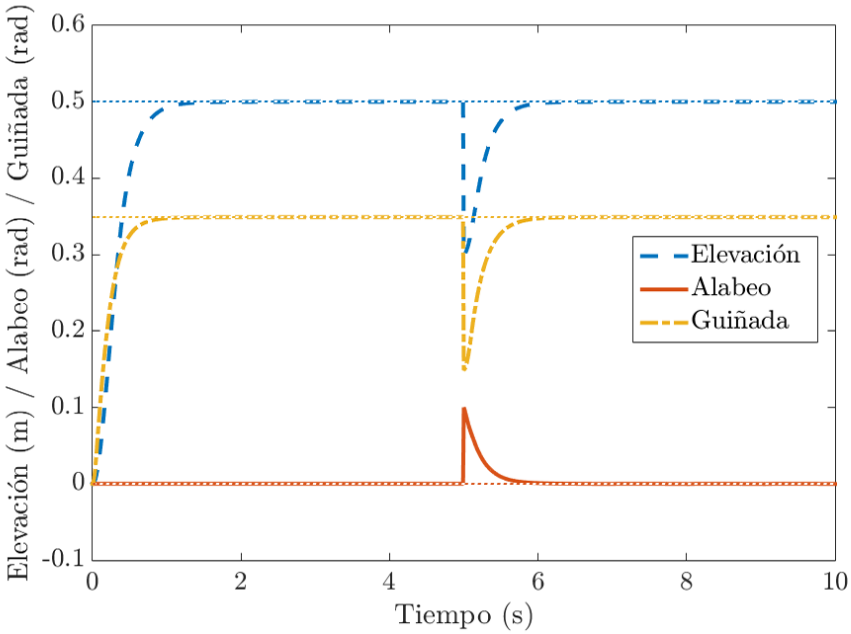

Figura 4.5 Evolución temporal de variables de estado de controlador ST

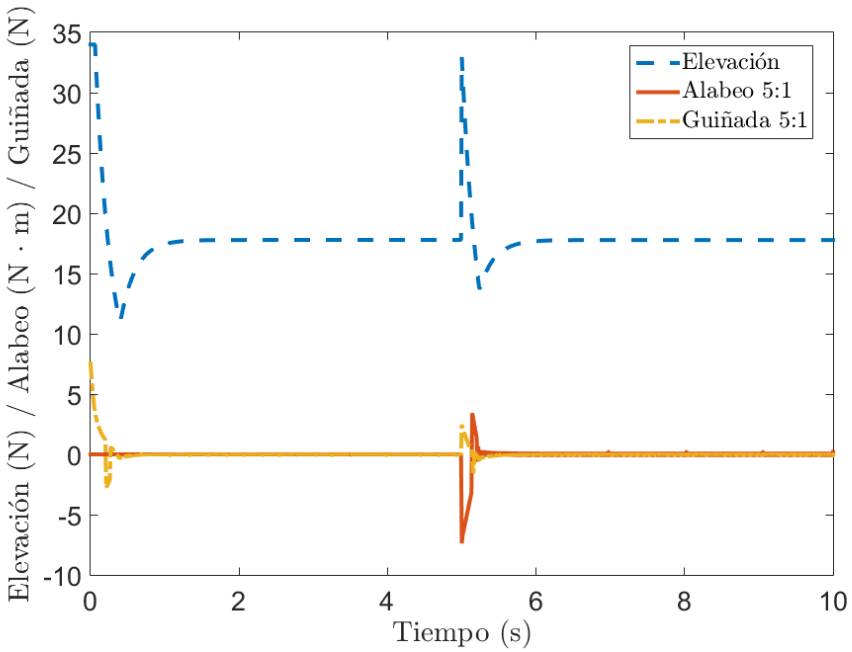

Figura 4.6 Evolución temporal de las señales de control para el controlador ST

4.4. Controlador Super Twisting Adaptativo (STA)

El controlador STA fue simulado con los parámetros del cuadro 4.6.

Cuadro 4.6 Parámetros del controlador STA

Parámetro	Valor
K_z	40
$Kmin_z$	8
λ_z	12
μ_z	0.01
ϵ_z	1
K_ϕ	1.2
$Kmin_\phi$	0.3
λ_ϕ	7
μ_ϕ	0.01
ϵ_ϕ	.1
K_ψ	.8
$Kmin_\psi$	0.15
λ_ψ	8
μ_ψ	0.01
ϵ_ψ	0.001

En la Figura 4.7 se muestra la evolución temporal de las variables de estado, mientras que en la Figura 4.8 se muestra la evolución en el tiempo de las señales de control para este controlador.

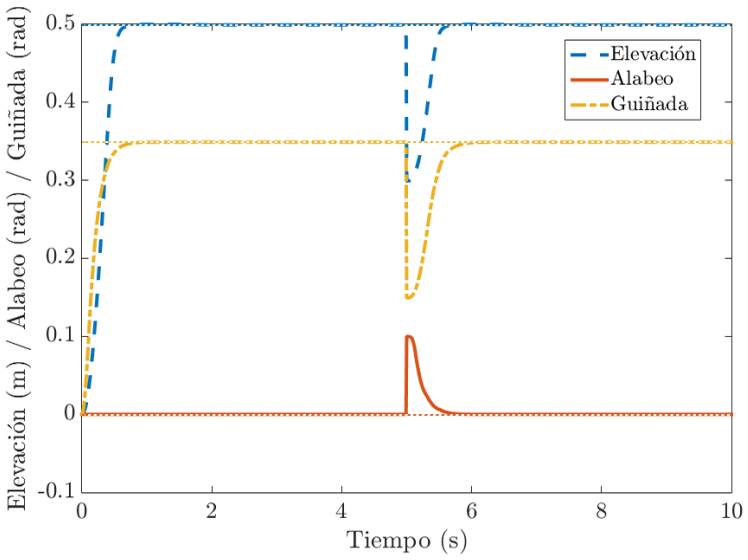

Figura 4.7 Evolución temporal de variables de estado de controlador STA

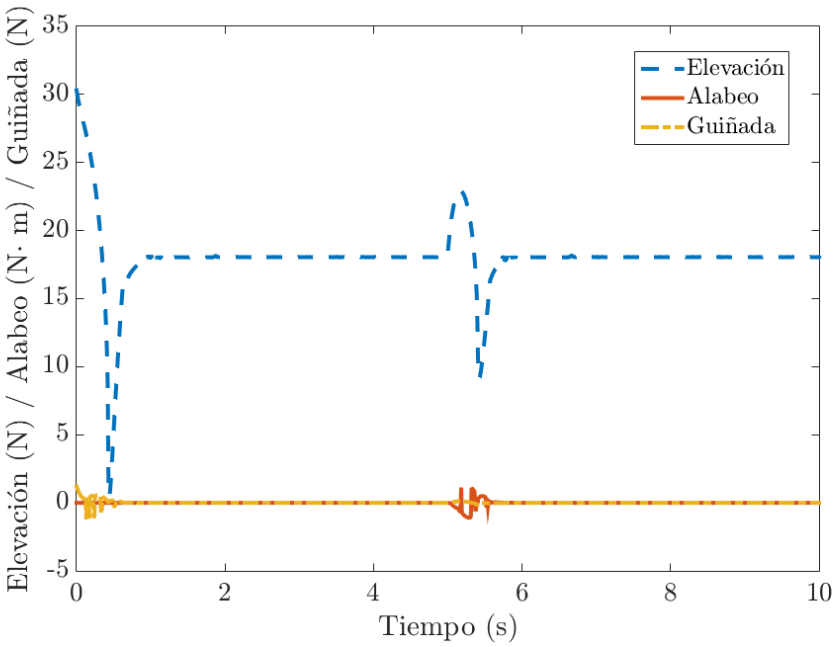

Figura 4.8 Evolución temporal de las señales de control para el controlador STA

4.5. Discusión de resultados

Para comparar el desempeño de los controladores simulados se analizaron las gráficas del vector de variables de estado, y las gráficas de las señales de control.

Para este análisis, se compararon los resultados de cada uno de los controladores SMC, ST y STA frente al controlador más utilizado en estos sistemas, el controlador PD con compensación de gravedad.

Se debe resaltar que en

4.5.1. SMC vs PD con compensación de gravedad

El controlador SMC con presenta un desempeño menos satisfactorio que el controlador PD con compensación de gravedad en este sistema. Si se compara la evolución de las variables de estado en el tiempo de ambas simulaciones, se puede notar que en el estado transitorio, ambos estabilizan en un tiempo similar, sin embargo, cuando se analizan las gráficas de la Figura 4.2 y de la Figura 4.4 se observa cómo las variaciones en las señales de control en el SMC son más abruptas, lo cual, en una aplicación real significa un gasto excesivo de energía de los motores. Además, el *chattering* producido por la naturaleza de este controlador, hace muy complicada la aplicación directa en la plataforma.

4.5.2. ST vs PD con compensación de gravedad

El desempeño mostrado en el controlador ST es muy superior al controlador SMC, dado que presenta un tiempo de estabilización aun más rápido que el controlador PD. Así mismo, se puede observar en la Figura 4.2 y Figura 4.6 que la señal de control para el controlador ST resulta ser muy conveniente al no saturarse ni tener variaciones repentinas en la misma. Una característica importante para obtener un buen desempeño en los motores al implementar este algoritmo físicamente.

Uno de los puntos fuertes de este controlador frente al controlador SMC es que al ser de segundo orden, se reduce significativamente el *chattering*.

4.5.3. STA vs PD con compensación de gravedad

El controlador Super Twisting Adaptativo fue el de mejor desempeño frente a los otros algoritmos simulados. Se observa en la Figura 4.7 que el tiempo para alcanzar el estado deseado es menor al tiempo tomado por los tres controladores anteriores.

Además, en la Figura 4.8 se muestra cómo las ganancias del controlador se adaptan más rápido frente a la perturbación que se presenta en el segundo 5. Con este controlador se reduce considerablemente el *chattering*, y además se logra una mejora en la señal de las ganancias, evitando las saturaciones en las señales de control y por ende el desgaste y mal funcionamiento de los motores. La desventaja de este controlador es que se requiere mayor capacidad de procesamiento por parte de la computadora en la que se implementa.

Con estas simulaciones realizadas, se procede a evaluar los controladores en la plataforma física.

Capítulo 5

Implementación práctica de controladores

En este capítulo se expone la metodología utilizada para la implementación de los controladores en la plataforma bi-rotor.

5.1. Esquema práctico

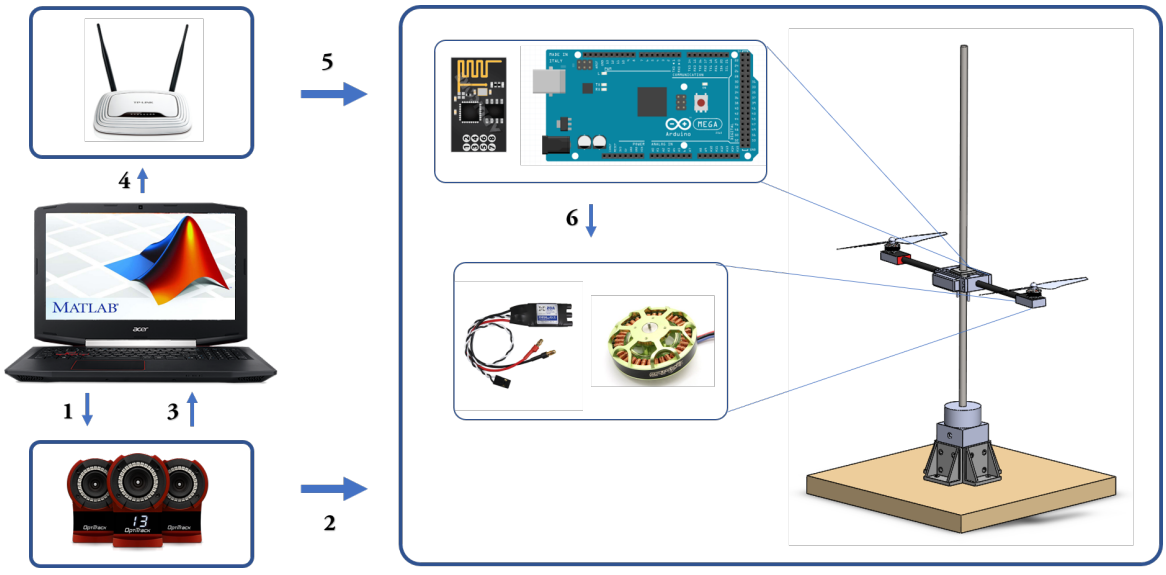

Figura 5.1 Diagrama de implementación práctica

Se presenta en la Figura 5.1 el esquema utilizado para poner en funcionamiento la plataforma y así la aplicación y evaluación de cada controlador abordado en la sección anterior. Cada flecha indica un proceso, y se detalla a continuación cada uno de ellos.

Se tiene un equipo de cómputo con Mathworks Matlab como procesador central. A su vez, el equipo de cómputo se encuentra conectado mediante WiFi a un enrutador, y enlazado al sistema de visión óptica con 6 cámaras Motive de Optitrack mediante una conexión USB.

En el paso 1 se envía un comando desde Matlab al sistema Motive para realizar la medición de la posición de la plataforma Bi Rotor. En el paso 2 se realiza la medición de los 3 grados de libertad de interés: Distancia de elevación, ángulo de alabeo y ángulo de guiñada. El paso 3 indica que la medición de cada grado de libertad es recibida en el equipo de cómputo, y capturada a través de un código de Matlab.

Una vez que Matlab recibe los datos medidos, se ejecuta una rutina: el controlador. Cuando éste calcula las señales de control, se lleva a cabo el paso 4, en donde se envía al enrutador una cadena de 10 caracteres con la siguiente estructura: 'a####b####'. Las letras 'a' y 'b' son utilizadas para indicar si la información pertenece al motor uno o al motor dos, mientras que los símbolos # indican los 4 dígitos que conforman la duración en milisegundos del ancho de pulso de la señal PWM que controla la velocidad de cada motor, siendo el número 1000 el 0 % de velocidad, mientras que el número 2100 indica el 100 % de velocidad.

Las señales de control son enviadas en conjunto con metadatos propios del protocolo TCP/IP. En el paso 5 esta cadena de información pasa del enrutador al módulo WiFi ESP8266 conectado al Arduino Mega. La función del Arduino Mega es interpretar la información que llega al ESP8266 y ejecutar una rutina que asigna a cada controlador ESC la velocidad indicada previamente por la rutina de control enviada desde Matlab. A su vez, el controlador ESC regula la velocidad de rotación de cada motor de la plataforma.

Físicamente, el sistema de cámaras, la computadora con Matlab y el enrutador se encuentran colocados en una posición fija, mientras que los demás elementos mostrados en el diagrama se encuentran integrados en la parte móvil del sistema bi-rotor.

Un solo código de Matlab es capaz de realizar las rutinas de obtención de mediciones, cálculo de señales de control, y envío de señales a motores. El proceso descrito desde el paso 1 al paso 6 se realiza 100 veces por segundo, es decir, se tiene un período de muestreo para este sistema de 10 ms.

Este código de Matlab es estructurado de forma muy similar en cada uno de los cuatro controladores evaluados. La diferencia fundamental entre un código y otro radica en el bloque destinado a hacer el cálculo de las señales de control.

La duración de la ejecución de cada experimento es ajustable. Para los propósitos de este trabajo, se hicieron pruebas de entre 10 y 15 segundos. Una vez que cada prueba era ejecutada, al terminar se generaban 6 gráficos por experimento: 3 correspondientes a la evolución temporal de las señales de control para cada grado de libertad, y 3 correspondientes a la evolución temporal de las variables de estado.

En la siguiente sección se muestran los gráficos de resultados obtenidos para cada controlador evaluado.

5.2. Resultados de controladores evaluados

El sistema bi-rotor presentó, al ser una plataforma experimental y por cuestiones de diseño, una variación significativa en las constantes de fricción para los 3 grados de libertad. Estas variaciones están directamente relacionadas con factores técnicos de lubricación de bujes, mala alineación de baleros y brazos, posición de la batería y desgaste de piezas por golpes.

Además se encontró una limitante en los motores utilizados. No fue posible obtener las aceleraciones estimadas en las simulaciones previas del controlador.

Por último, las baterías utilizadas en el bi-rotor, a pesar de ser baterías litio-polímero de alta capacidad de descarga, éstas iban reduciendo su desempeño conforme se ejecutaba un experimento tras otro.

Todos estos factores repercutían directamente en el desempeño de cada controlador, y a su vez, en la complejidad de sintonización de cada uno de ellos. Por ende, la sintonización de los parámetros de cada controlador fue realizada de forma experimental, sintonizando primero las ganancias de control de alabeo, y posteriormente, las ganancias de control de elevación, hasta obtener resultados que no pusieran en riesgo la integridad de la plataforma. Este procedimiento experimental fue realizado para cada uno de los cuatro controladores evaluados.

Como último punto a destacar, en este análisis se decidió no efectuar un control sobre el ángulo de guiñada, debido a la falta de consistencia en los resultados arrojados. Esta inconsistencia en los resultados puede ser atribuible en primer lugar a errores en la alineación y en el ensamble de piezas, así como en la fatiga producida por golpes

producto de la experimentación. Además, siendo el bi-rotor un sistema sub actuado es difícil producir las señales de control que compensan de manera efectiva cambios en la guiñada. Este problema se enfrentará en subsecuentes trabajos. Razón por la cual, se concentra el trabajo sobre el control de alabeo y elevación.

Cuadro 5.1 Valores de referencia PD+

Grado de libertad	Valor	unidad
Elevación z	0.75	m
Alabeo ϕ	0	grados

5.2.1. Controlador PD con compensación de gravedad

Para el controlador PD se utilizaron los valores de referencia del cuadro 5.1, los cuales fueron definidos así con el objetivo de hacer funcionar la plataforma bi-rotor entre los rangos de elevación y alabeo adecuados para someterlos a un análisis y también garantizar el mínimo impacto y daño al bi-rotor.

Cuadro 5.2 Parámetros del controlador PD

Parámetro	Valor
Kp_z	40
Kd_z	12
Kp_ϕ	0.50
Kd_ϕ	0.18

Para sintonizar el controlador, se fueron ajustando una a una las ganancias proporcional y derivativa, tanto en el control de elevación como en el control de alabeo. Estas ganancias para el controlador se indican en el Cuadro 5.2.

Los resultados mostrados en la Figura 5.2 exhiben la evolución temporal de la elevación y el ángulo de alabeo. Mientras que en la Figura 5.3 se puede observar la evolución temporal de las señales de control.

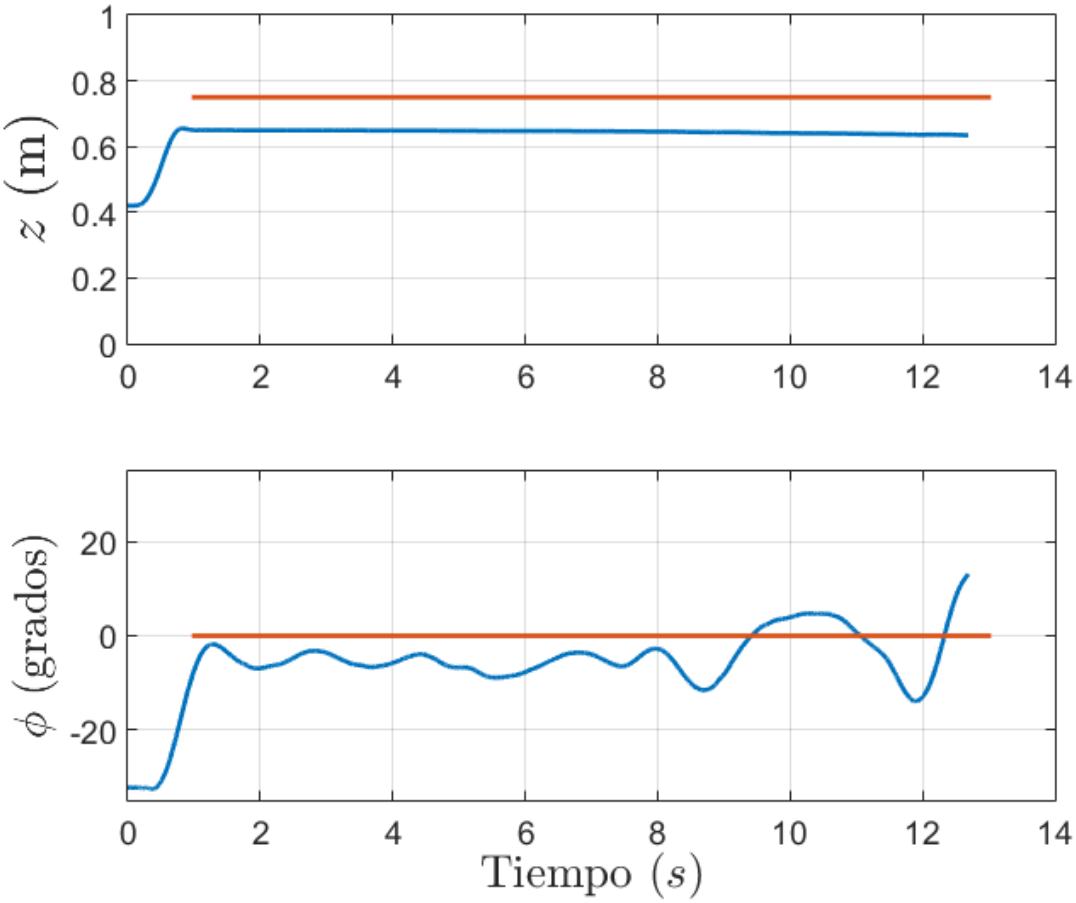

Figura 5.2 Evolución temporal de las variables de estado

Análisis de Resultados

Se observa en el primer gráfico de la Figura 5.2 que la elevación es controlada con una respuesta de menos de 1 segundo, y permanece estabilizada en 0.65 m del segundo 1 al segundo 10. Observando el primer gráfico de la Figura 5.3 el controlador satura la señal para poder elevar el bi-rotor. Una vez que esta alcanza los 0.65 m, el controlador establece una fuerza de 22.7 N para mantener el bi-rotor en la elevación establecida.

El error en elevación es de 0.1m. Este error se atribuye al desgaste en la carga de la batería y debido a que el controlador PD no es capaz de compensar estas perturbaciones.

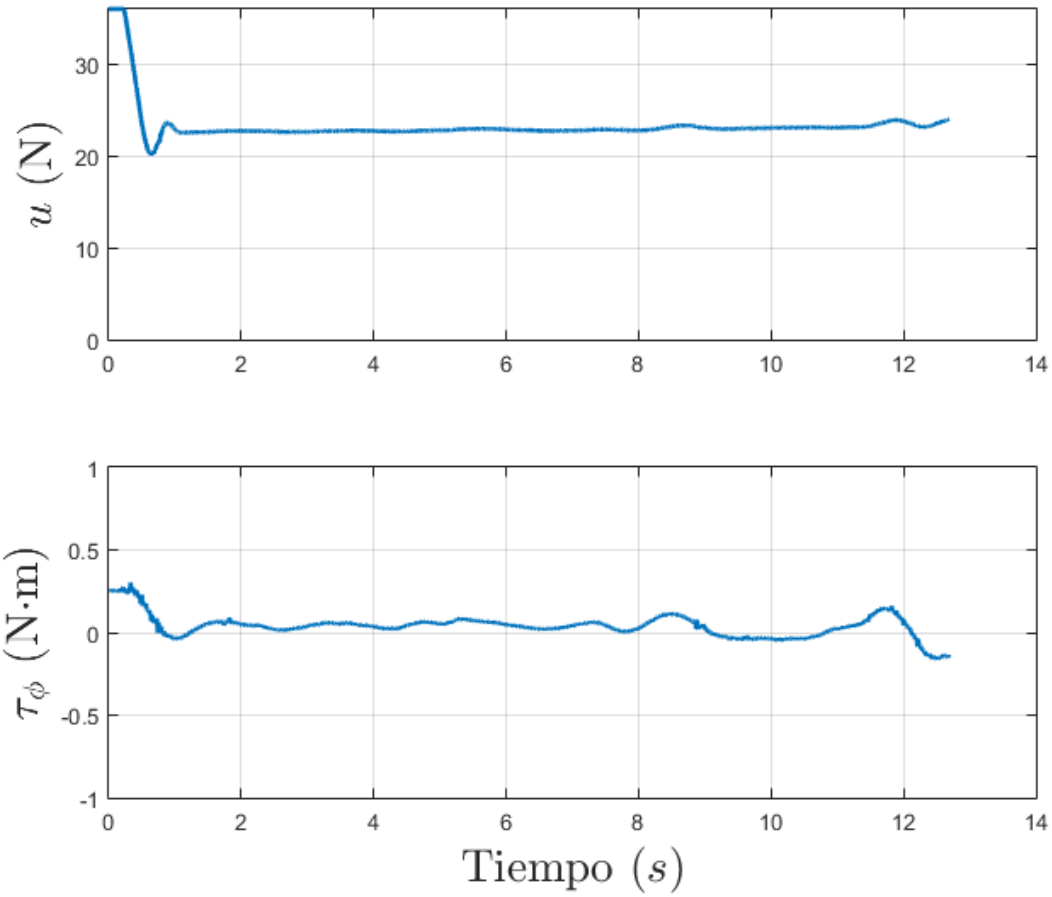

Figura 5.3 Evolución temporal de las señales de control

Para el ángulo de alabeo se observa en el segundo gráfico de la Figura 5.2 que el controlador lleva al bi-rotor a un valor cercano a los 0 grados establecidos como valor de referencia. Sin embargo, del segundo 1 al segundo 9 se observan oscilaciones entre los 0 y -10 grados, y del segundo 9 en adelante, una perturbación mayor. En la Figura 5.3 se observa que son variaciones de menos de 0.4 Nm en el torque rotacional las que se encargan de realizar el control de alabeo.

El desempeño de este controlador, tanto en elevación como en ángulo de alabeo pudiera verse mejorado integrando una batería con menos desgaste y mayor capacidad y con

una mejor sintonización en las ganancias proporcionales, derivativas, y en el factor de compensación de gravedad.

5.2.2. Controlador SMC

Para el controlador SMC se utilizaron los valores de referencia del cuadro 5.3, los cuales fueron definidos siguiendo la misma metodología del controlador PD+.

Cuadro 5.3 Valores de referencia controlador SMC

Grado de libertad	Valor	unidad
Elevación z	0.75	m
Alabeo ϕ	0	grados

Las ganancias del controlador SMC se fueron calibrando manualmente, sin seguir un procedimiento establecido. Con la finalidad de no infringir daño a la plataforma. en el cuadro 5.4 se presentan los valores de los factores $\mathbf{K}$ y λ.

Cuadro 5.4 Parámetros del controlador SMC

Parámetro	Valor
K_z	2
λ_z	5
K_ϕ	0.08
λ_ϕ	8

Posteriormente en las Figura 5.4 y Figura 5.5 se muestra la evolución temporal de las variables de estado y de las señales de control respectivamente.

Análisis de Resultados

Esta prueba se ejecutó en 9 segundos. Se observa en la Figura 5.4 el gráfico correspondiente a la elevación z. La elevación de referencia es de 0.75 m. El controlador SMC es capaz de llevar la plataforma bi-rotor a una elevación de 0.73 m en menos de 2.5 segundos con la señal de control u, observable en el gráfico superior de la Figura 5.5, y permanecer en ella hasta los 8 segundos. Del segundo 8 al segundo 9, se presenta un descenso que rápidamente es compensado por la señal de control.

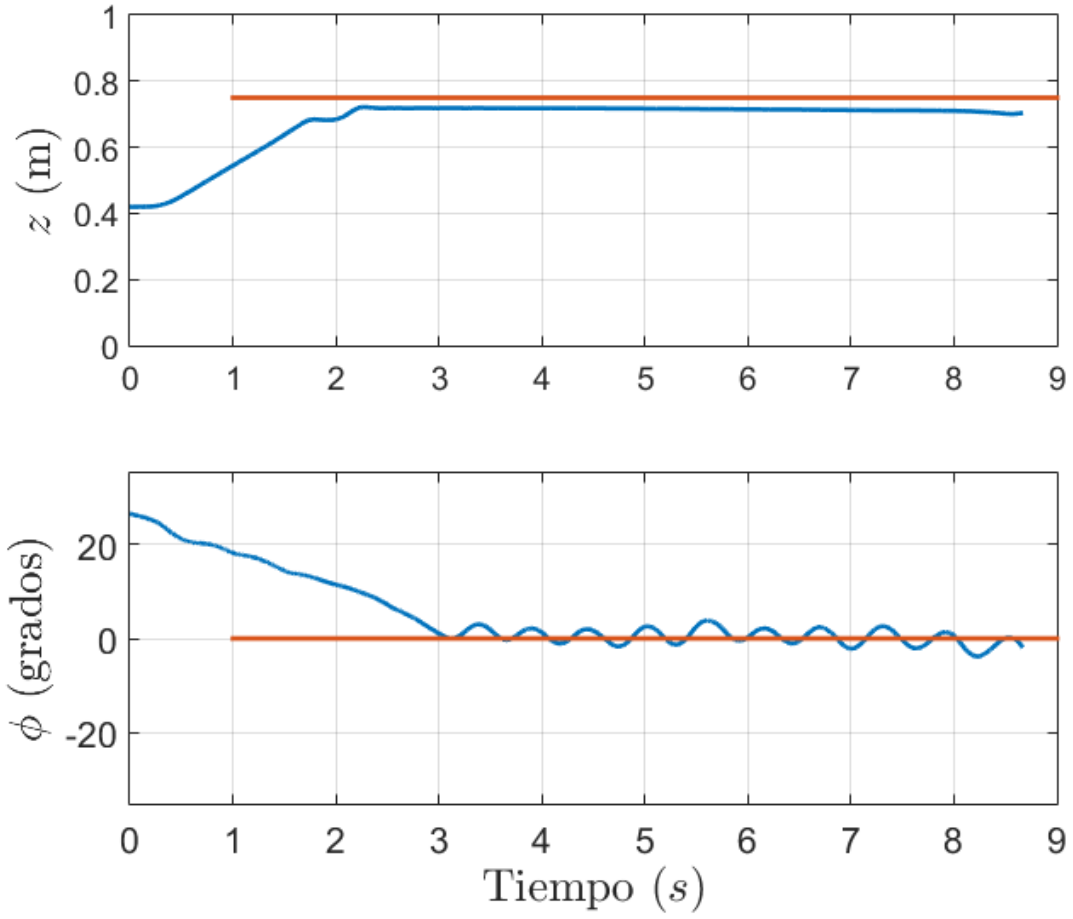

Figura 5.4 Evolución temporal de las variables de estado

Se logra observar en el grafico ϕ de la Figura 5.4 que la señal de control τ_ϕ de la Figura 5.5 logra llevar el alabeo de 30 grados hasta los 0 grados deseados en 3 segundos. Del segundo 3 hasta concluir el experimento, se observa como el alabeo oscila en valores entre -1 y 4 grados. Esta oscilación viene reflejada por el fenómeno de vibración (chattering) que es inherente a este tipo de controlador.

En un panorama general, el controlador SMC, a pesar de presentar una respuesta lenta, es capaz de llevar las variables de estado del bi-rotor a los niveles de referencia deseados, presentando un error en elevación de 0.02 m y en alabeo de ± 4 grados.

Esta respuesta podría verse mejorada mediante una sintonización más detallada de las ganancias para el controlador SMC.

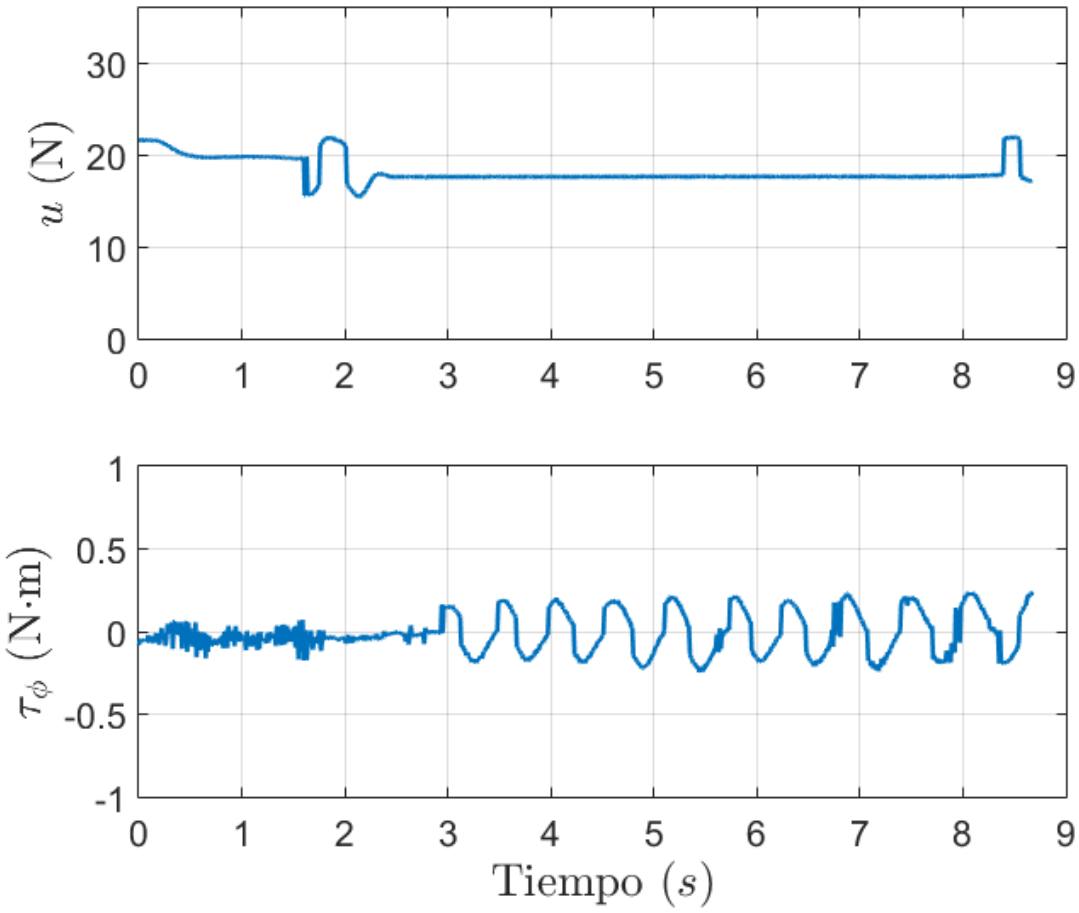

Figura 5.5 Evolución temporal de las señales de control

5.2.3. Controlador Super Twisting

Siguiendo la misma metodología que en los controladores anteriores, se designaron los valores de referencia del cuadro 5.5 para los experimentos del controlador ST.

La sintonización de este controlador se realizó con la misma metodología de los controladores anteriores. Las ganancias del controlador se muestran en el cuadro 5.6.

Cuadro 5.5 Valores de referencia controlador ST

Grado de libertad	Valor	unidad
Elevación z	0.70	m
Alabeo ϕ	0	grados

Cuadro 5.6 Parámetros del controlador ST

Parámetro	Valor
K_{1z}	1.4
K_{2z}	0.015
λ_z	1
$K_{1\phi}$	0.15
$K_{2\phi}$	0.0008
λ_ϕ	0.85

Con un tiempo de experimento de 9 segundos, en la Figura 5.6 y Figura 5.7 se muestra la evolución temporal de las variables de estado y de las señales de control respectivamente.

Análisis de Resultados

Observe el gráfico de elevación de la Figura 5.6. Se puede notar como el controlador lleva la elevación del bi-rotor desde 0.36m a 0.705 m en un tiempo menor de 2.5 segundos.

Para el alabeo, el controlador ST logra que esta variable se mantenga oscilando entre valores de ± 5 grados.

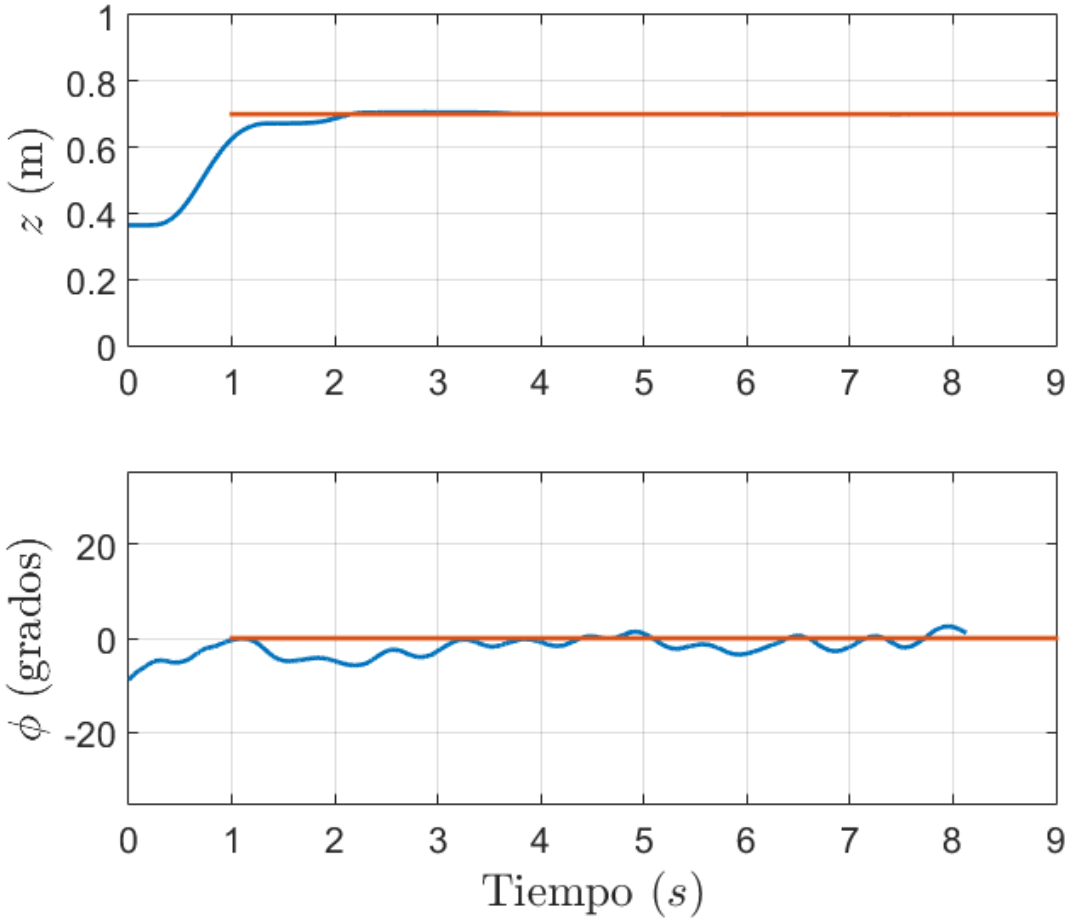

Figura 5.6 Evolución temporal de las variables de estado

En la Figura 5.7 se logra observar como las señales correspondientes al control de elevación y alabeo son similares a las observadas en el controlador SMC, con una diferencia fundamental: El fenómeno de *chattering* se ve muy reducido.

Para este controlador se observó una característica bastante interesante. Durante los experimentos, el controlador ST fue capaz de compensar la descarga de la batería, y aun así llevar la elevación al nivel de referencia deseado (0.70 m), resaltando respecto a los controladores anteriores.

Se considera además, que la respuesta de este controlador se pudiera ver mejorada con una sintonización más detallada para las ganancias $\mathbf{K_1}$ $\mathbf{K_2}$ y λ.

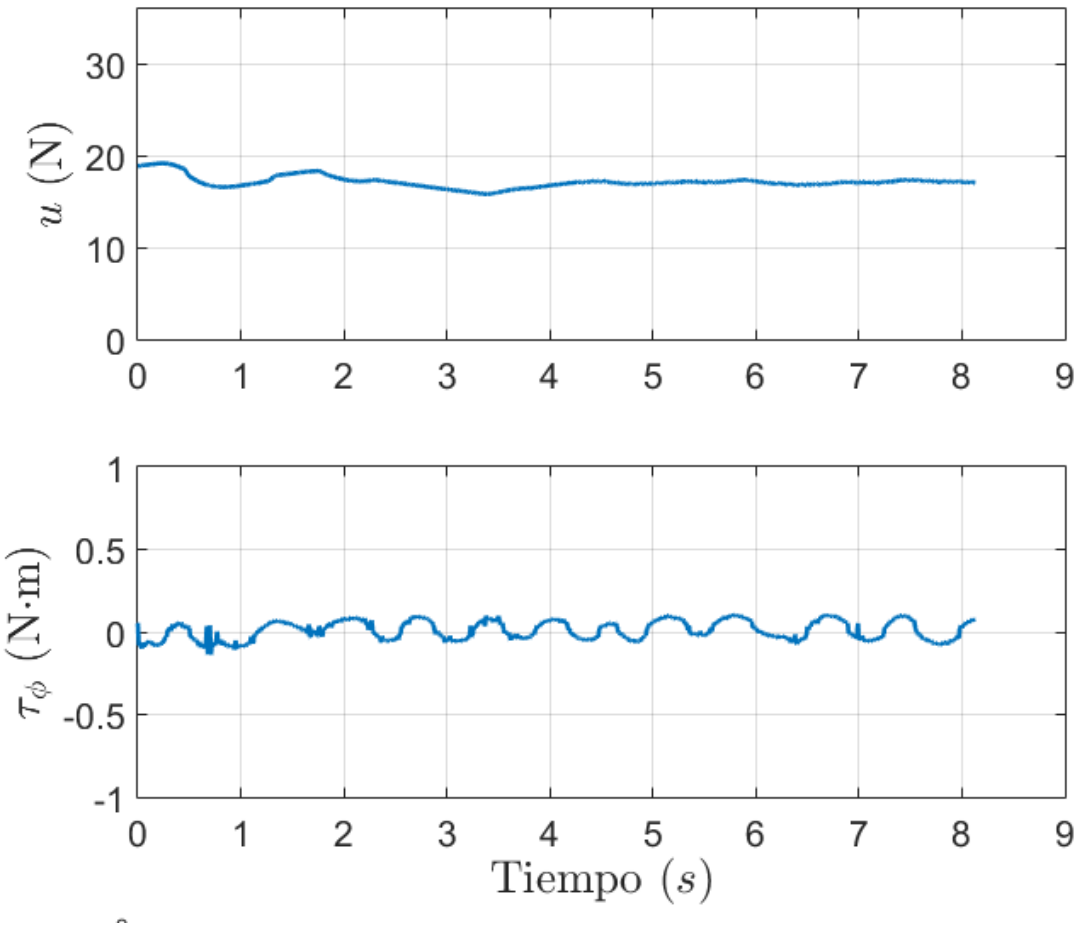

Figura 5.7 Evolución temporal de las señales de control

5.2.4. Controlador Super Twisting Adaptativo

Los valores de referencia para el controlador STA se enlistan en el cuadro 5.7, mientras que las ganancias para el controlador, y resultado de una laboriosa sintonización, se muestran en el cuadro 5.8.

En la Figura 5.8 se tiene la evolución temporal de las variables de elevación y alabeo. En la Figura 5.9 se observa la evolución temporal de las señales de control.

Este experimento se ejecutó con un tiempo de 12 s.

Cuadro 5.7 Valores de referencia controlador STA

Grado de libertad	Valor	unidad
Elevación z	0.70	m
Alabeo ϕ	0	grados

Cuadro 5.8 Parámetros del controlador STA

Parámetro	Valor
K_z	0.05
$Kmin_z$	0.003
λ_z	1.8
μ_z	0.15
ϵ_z	0.01
K_ϕ	0.0005
$Kmin_\phi$	0.00001
λ_ϕ	3.0
μ_ϕ	0.12
ϵ_ϕ	0.000001

Análisis de Resultados

Para la variable de elevación de este controlador, como se observa en la Figura 5.8, se alcanza el valor de referencia (0.70 m) en un tiempo de 2 segundos. Del segundo 2 en adelante, la elevación del bi-rotor se mantiene estabilizada, con un error de 0.02 m.

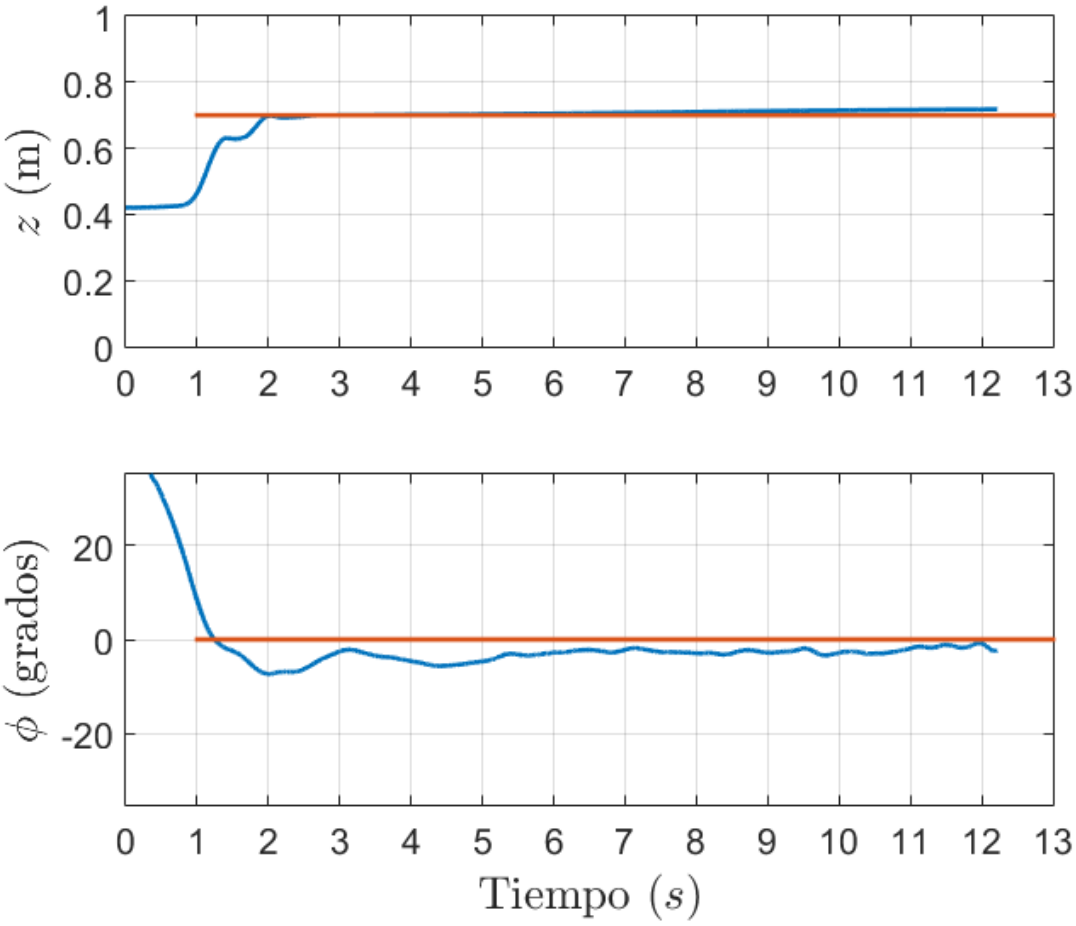

Figura 5.8 Evolución temporal de las variables de estado

Para el ángulo de alabeo, este controlador presenta una respuesta de estabilización al valor de referencia (0 grados) con un sobretiro observable en la segunda gráfica de la Figura 5.8. El error presente en estado estacionario es de -5 grados.

En la Figura 5.9 se puede observar cómo actuan las señales de control para obtener las respuestas para la elevación y el alabeo. Se observa una perturbación entre el segundo

5 y el segundo 6 en la señal de control de alabeo, debido a un error en el muestreo de los datos.

En este controlador no se aprecian saturaciones, ni fenómeno de *chattering*.

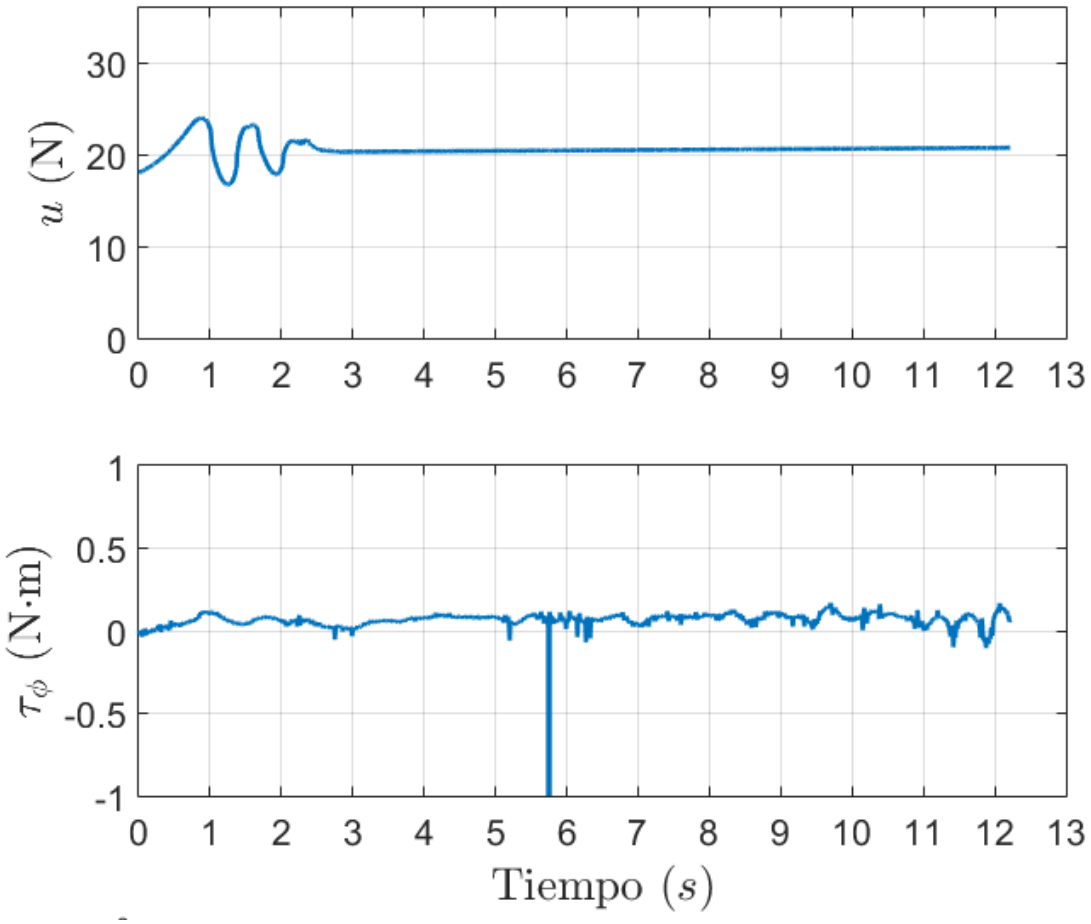

Figura 5.9 Evolución temporal de las señales de control

5.3. Resumen de controladores

Haciendo un análisis comparativo entre los controladores evaluados se concluye con lo siguiente:

El controlador PD+ se considera un buen punto de partida para poder aplicarse en este sistema. Sin embargo, este presenta una pobre respuesta ante la compensación de la elevación y el desgaste de la batería. La ventaja del controlador PD+ frente a los otros 3 controladores evaluados, es su rapidez de implementación y su bajo requerimiento de capacidad de proceso.

El controlador SMC basado en el algoritmo supertwisting, presenta una respuesta más lenta, pero con menor error que el controlador PD+. Además, presenta un comportamiento más robusto ante perturbaciones, como lo es la descarga de la batería.

La principal desventaja del controlador SMC es presentar chattering, que para el sistema bi-rotor supone un problema que genera vibraciones no deseadas.

El controlador ST presenta una respuesta mejorada del controlador SMC. Debido a que el controlador ST minimiza el efecto de *chattering* presente en el controlador SMC, se logra una robustez, con señales de control más suavizadas.

El punto a destacar al experimentar con el controlador ST fue la capacidad de compensar la elevación, aun teniendo poca carga en la batería.

Por último, con el controlador STA se tuvo una respuesta muy similar al controlador ST, no existiendo diferencia notable en el desempeño del bi-rotor entre ambos controladores.

Debido al bajo desempeño obtenido en el controlador PD, las oscilaciones no deseadas (producto del *chattering*) del controlador SMC, y a la dificultad de sintonización para el STA, se puede concluir que para el sistema bi-rotor, el controlador ST resulta ser el más conveniente por su desempeño frente a las perturbaciones y su relativa facilidad de sintonización (al poseer solo 3 ganancias de control por grado de libertad).

Capítulo 6

Conclusiones

Durante el desarrollo de este trabajo de investigación se presentaron algunos puntos importantes a resolver.

Con el análisis realizado acerca de las plataformas experimentales existentes y con un modelo matemático propuesto, el trabajo se ubicó en desarrollar exitosamente la plataforma física.

Una vez que fueron maquiladas las piezas, se comprobó el funcionamiento mecánico de la plataforma, encontrándose perturbaciones mínimas.

Se buscó implementar el sistema electrónico mediante el uso de la placa de desarrollo BeagleBone Black en conjunto con el software LabView, sin embargo, después de numerosas pruebas sin éxito, se descarta este esquema y se propone el uso del sistema Optitrak, el cual resulta ser mucho más útil, por su precisión y confiabilidad.

Con la plataforma funcional mecánicamente, se exploran las diferentes técnicas de control planteadas en capítulos anteriores.

El diseño y las simulaciones de los controladores sobre el modelo matemático propuesto mostraron la posibilidad de implementar los controladores de manera física.

Posterior al correspondiente estudio de cada uno de ellas, se hace una integración exitosa en todo el sistema electrónico-mecánico, por lo cual a partir de ese momento se comienza a experimentar con los algoritmos de control sobre la plataforma bi-rotor.

Por último, se demostró que fue posible implementar los cuatro controladores no lineales sobre la elevación y el alabeo. El controlador Super Twisting mostró una relativa facilidad de implementación, con resultados óptimos

Como trabajo a futuro se proponen modificaciones al diseño estructural, que permitan mejorar la resistencia de la plataforma, así como la disminución de perturbaciones asociadas al diseño.

Se sugiere además, la implementación más profunda y sistemática de los controladores ya trabajados, y el estudio de controladores adicionales sobre los tres grados de libertad.

Bibliografía

[1] Aldebrez, F. M., Darus, I. Z. M., and Tokhi, M. O. (2004). Dynamic modelling of a twin rotor system in hovering position. In *First International Symposium on Control, Communications and Signal Processing, 2004.*, pages 823–826.

[2] Bartoszewicz, A. and Żuk, J. (2010). Sliding mode control; basic concepts and current trends. In *2010 IEEE International Symposium on Industrial Electronics*, pages 3772–3777.

[3] Bertrand, S., Guénard, N., Hamel, T., Piet-Lahanier, H., and Eck, L. (2011). A hierarchical controller for miniature vtol uavs: Design and stability analysis using singular perturbation theory. *Control Engineering Practice*, 19(10):1099 – 1108.

[4] Bouabdallah, S., Murrieri, P., and Siegwart, R. (2005). Towards autonomous indoor micro vtol. *Autonomous Robots*, 18(2):171–183.

[5] Castañeda, H. and Gordillo, J. (2017). Spatial modeling, identification and adaptive second order sliding mode control of a micro air vehicle. *IEEE International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1083–1091.

[6] Castañeda, H., Plestan, F., Chriette, A., and de León-Morales, J. (2016a). Continuous differentiator based on adaptive seccond-order sliding-mode control for a 3-dof helicopter. *IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS*, 63:5786–5793.

[7] Castañeda, H., Salas-Peña, O. S., and de León-Morales, J. (2016b). Extended observer based on adaptive second order sliding mode control for a fixed wing uav. *ISA Transactions*, 66:226–232.

[8] Castañeda-Herman, L.A., C., Alejandro, L., and J.L., G. (2017). Guidelines for propulsion system design and implementation in a quadrotor mav. *IEEE International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1302–1308.

[9] Herisse, B., Russotto, F. X., Hamel, T., and Mahony, R. (2008). Hovering flight and vertical landing control of a vtol unmanned aerial vehicle using optical flow. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 801–806.

[10] Hua, M.-D., Hamel, T., Morin, P., and Samson, C. (2009). A control approach for thrust-propelled underactuated vehicles and its application to vtol drones. *IEEE TRANSACTIONS ON AUTOMATIC CONTROL*, 54(8):1837–1853.

[11] Islam, B. U., Ahmed, N., Bhatti, D. L., and Khan, S. (2003). Controller design using fuzzy logic for a twin rotor mimo system. In *7th International Multi Topic Conference, 2003. INMIC 2003.*, pages 264–268.

[12] Kelly, R. and Santibáñez, V. (2003). *Control de movimiento de robots manipuladores*. Automática y Robótica. Pearson Educación.

[13] Kendoul, F., Fantoni, I., and Lozano, R. (2006). Modeling and control of a small autonomous aircraft having two tilting rotors. *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference 2005*.

[14] Khalil, H. (2014). *Nonlinear Control*. Pearson Education.

[15] Lee, D., Ryan, T., and Kim, H. J. (2012). Autonomous landing of a vtol uav on a moving platform using image-based visual servoing. In *2012 IEEE International Conference on Robotics and Automation*, pages 971–976.

[16] Lippiello, V., Ruggiero, F., and Serra, D. (2014). Emergency landing for a quadrotor in case of a propeller failure: A backstepping approach. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4782–4788.

[17] Papachristos, C., Alexis, K., and Tzes, A. (2011). Design and experimental attitude control of an unmanned tilt-rotor aerial vehicle. In *2011 15th International Conference on Advanced Robotics (ICAR)*, pages 465–470.

[18] Phang, S. K., Li, K., Yu, K. H., Chen, B. M., and Lee, T. H. (2014). Systematic design and implementation of a micro unmanned quadrotor system. *Unmanned Systems*, 2:121–141.

[19] Rafiq, M., ur Rehman, S., ur Rehman, F., Butt, Q. R., and Awan, I. (2012). A second order sliding mode control design of a switched reluctance motor using super twisting algorithm. *Simulation Modelling Practice and Theory*, 25:106–117.

[20] Saeki, M. and Sakaue, Y. (2001). Flight control design for a nonlinear non-minimum phase vtol aircraft via two-step linearization. In *Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No.01CH37228)*, volume 1, pages 217–222 vol.1.

[21] Shtessel, Y., Taleb, M., and Plestan, F. (2012). A novel adaptive-gain supertwisting sliding mode controller: Methodology and application. 48:759–769.

[22] Spong, M., Hutchinson, S., and Vidyasagar, M. (2005). *Robot Modeling and Control*. Wiley.

[23] Tao, C., Taur, J., and Chen, Y. (2010). Design of a parallel distributed fuzzy lqr controller for the twin rotor multi-input multi-output system. *Fuzzy Sets and Systems*, 161(15):2081 – 2103. Theme: Non-Linear Control.

Apéndice A

Códigos de simulación de controladores

A.1. Código en Matlab de modelo bi-rotor

El modelo matemático del bi-rotor fue codificado en un bloque de Matlab y se muestra a continuación

MATLAB Script A.1 birotor.m

```
1  ********************************************************************
2  function [out] = birrotor (in)
3
4  q=in(1:3);       %Vector de posiciones
5  qp=in(4:6);      %Vector de velocidades
6  tau=in(7:9);     %Vector de entradas
7
8  kf1=0.01;         %Constante de friccion de elevacion.
9  kf2=0.01;         %Constante de friccion del alabeo.
10 kf3=0.01;         %Constante de friccion de guinada.
11
12 m=1.84;           %Masa del sistema(kg)
13 l=0.34;           %Longitud
14
15 Ix=0.043;         %Inercia de roll (Obtenidas experimentalmente)
16 Iz=0.041;         %Inercia del yaw (Obtenidas experimentalmente)
17 Cphi=cos(q(2));   %Coseno de phi
18 Sphi=sin(q(2));   %Seno de phi
```

```
19
20 M=[m 0 0; 0 Ix 0; 0 0 Iz*Cphi^2];    %Matriz masas e inercias
21 Cqp=[0;Iz*Cphi*Sphi*qp(3)^2;-2*Iz*Cphi*Sphi*qp(2)*qp(3)];
22 %Vector de coriolis
23 f=[kf1 0 0; 0 kf2 0; 0 0 kf3]*qp;
24 %Vector de fricciones
25 g=[9.81*m; 0; 0];  %Vector de gravedad
26
27 out=inv(M)*(tau-Cqp-g-f);
28 ****************************************************************
```

Para propósitos de simulación de los controladores fue necesario realizar para cada uno de ellos un código, y en Simulink un diagrama de bloques. Dado que cada código presenta características específicas para cada tipo de control. Éstos se detallan en las secciones siguientes.

A.2. Controlador PD con compensación de gravedad

En esta sección se muestra el código y el diagrama conceptual utilizado para el controlador PD. Consulte en el Apéndice A.1 el diagrama en Simulink para este controlador.

A.2.1. Código Matlab Controlador PD con compensación de gravedad

MATLAB Script A.2 ControladorPD.m

```
1  ****************************************************************
2  function [tau] = ControlPDt(in)
3
4  q=in(1:3);
5  qd=in(4:6);
6  qp=in(7:9);
7  qe=qd-q;
8
9  Kp=[50 0 0; 0 10 0; 0 0 10]
10 Kv=[18 0 0;0 1.8 0;0 0 1.8];
11 tau=Kp*qe - Kv*qp + [9.81*1.84;0;0];
12 ****************************************************************
```

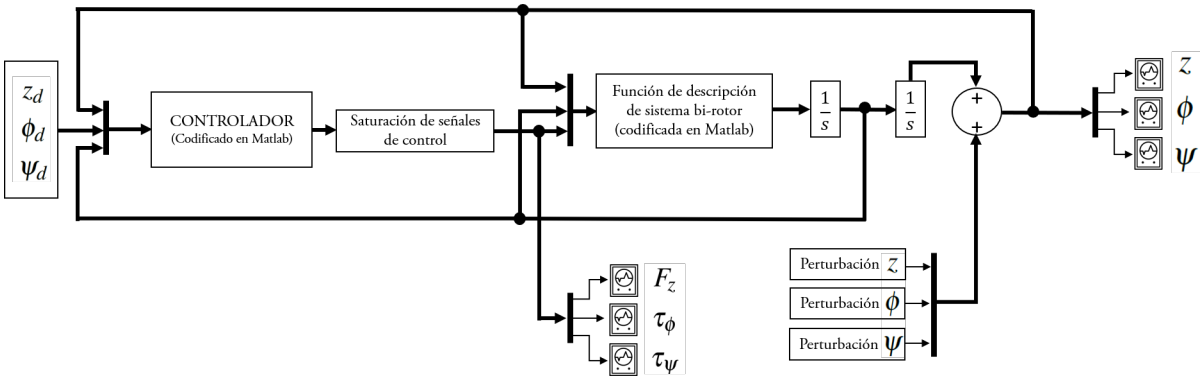

Figura A.1 Diagrama de bloques conceptual utilizado para el controlador PD con compensación de gravedad y SMC

A.2.2. Diagrama de bloques controlador PD con compensación de gravedad

Se muestra en la Figura. B.1.

A.3. Control SMC

A.3.1. Código Matlab Controlador SMC

MATLAB Script A.3 ControladorSMC.m

```
1  ************************************************************************
2  function [tau] = ControlSMC(in)
3
4  q=in(1:3);
5  qd=in(4:6);
6  qp=in(7:9);
7
8  kf1=0.1;          %Constante de friccion de altitud.
9  kf2=0.1;          %Constante de friccion del alabeo.
10 kf3=0.1;          %Constante de friccion de guinada.
11
12 m=1.84;            %Masa del sistema.
13
14 Ix=0.043;           %Inercia de roll (Obtenidas experimentalmente)
15 Iz=0.041;           %Inercia del yaw (Obtenidas experimentalmente)
```

```
16 Cphi=cos(q(2));    %Coseno de phi
17 Sphi=sin(q(2));    %Seno de phi
18
19 %Matriz de masas e inercias
20 M=[m 0 0; 0 Ix 0; 0 0 Iz*Cphi^2];
21 %Vector de coriolis
22 Cqp=[0;Iz*Cphi*Sphi*qp(3)^2;-2*Iz*Cphi*Sphi*qp(2)*qp(3)];
23 %Vector de fricciones
24 f=[kf1 0 0; 0 kf2 0; 0 0 kf3]*qp;
25 %Vector de gravedad
26 g=[9.81*m; 0; 0];
27
28 %%%%%%%%%%Parametros de control por modos deslizantes %%%%%%%%%%%%%
29
30 %Vector de velocidades en SetPoints
31 qdp=[0;0;0];
32 %Vector de aceleraciones en SetPoints
33 qdpp=[0;0;0];
34
35 k=[15 0 0;0 2 0;0 0 2];                    %Ganancia de Control
36 lambda=[10 0 0;0 10 0;0 0 10];             %Ganancia de q-qd
37
38 s=qp-qdp+lambda*(q-qd);                    %Fase deslizante
39
40 %Control Auxilizar -> Modos deslizantes <-
41 ua=-k*sign(s);
42 tau=M*(qdpp-lambda*(qp-qdp))+Cqp+g+f+ua;   %Salida tau
43 ********************************************************************
```

A.3.2. Diagrama de bloques controlador SMC

El diagrama para simular el Controlador SMC es el mismo diagrama utilizado para la simulación del controlador PD con compesación de gravedad, y éste se aprecia en la Figura B.1.

A.4. Controlador Super Twisting y Super Twisting Adaptativo

Se conceptualiza el diagrama de bloques utilizado para estos dos controladores en la Figura A.2.

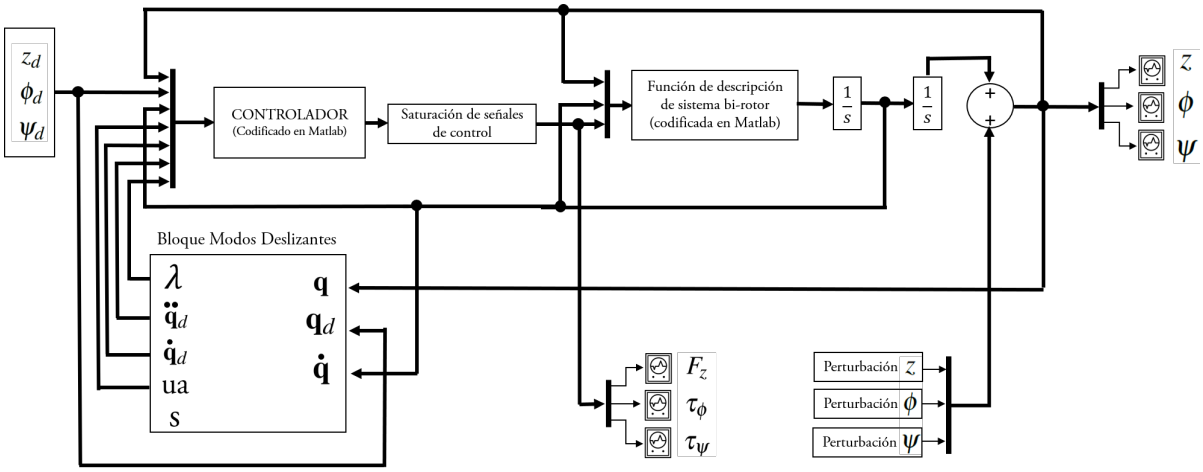

Figura A.2 Diagrama conceptual para controladores ST y STA

A.4.1. Código Matlab Controladores ST y STA

MATLAB Script A.4 ControladorSTySTA.m

```
1  ********************************************************************************
2  function [tau] = ControlSMC_SupTwi_y_AdSupTwi(in)
3  q=in(1:3);
4  %qd=in(4:6);
5  qp=in(7:9);
6  ua=in(10:12);
7  qdp=in(13:15);
8  qdpp=in(16:18);
9  lambda=in(19);
10
11 kf1=0;            %Constante de friccion de altitud.
12 kf2=0;            %Constante de friccion del alabeo.
13 kf3=0;            %Constante de friccion de guinada.
14
15 m=1.84;             %Masa del sistema.
```

```
16
17 Ix=0.043;            %Inercia de roll (Obtenidas experimentalmente)
18 Iz=0.041;            %Inercia del yaw (Obtenidas experimentalmente)
19 Cphi=cos(q(2));      %Coseno de phi
20 Sphi=sin(q(2));      %Seno de phi
21
22 %Matriz de masas e inercias
23 M=[m 0 0; 0 Ix 0; 0 0 Iz*Cphi^2];
24 %Vector de coriolis
25 Cqp=[0;Iz*Cphi*Sphi*qp(3)^2;-2*Iz*Cphi*Sphi*qp(2)*qp(3)];
26 %Vector de fricciones
27 f=[kf1 0 0; 0 kf2 0; 0 0 kf3]*qp;
28 %Vector de gravedad
29 g=[9.81*m; 0; 0];
30
31
32 %Ecuacion de control por modos deslizantes
33
34
35 %El control Auxiliar es procesado mediante Simulink
36
37 tau=M*(qdpp-lambda*(qp-qdp))+Cqp+g+f+ua;         %Salida tau
38 ********************************************************************
```

A.4.2. Diagrama Simulink Controladores ST y STA

El diagrama conceptual mostrado en la Figura A.2 corresponde al utilizado tanto en el controlador Super Twisting de Segundo Orden, como en el Super Twisting de Segundo Orden Adaptativo. La diferencia para uno y otro controlador radica en el subsistema presente en el bloque "Modo Deslizante S". Para más detalles del diagrama de Simulink para estos controladores, consulte la Figura B.2.

A.4.3. Bloque de Control Super Twisting

En la Figura B.3 se observa la programación del control auxiliar propuesto, cuyas ganancias se detallan en la sección de simulaciones.

A.4.4. Bloque de Control Super Twisting Adaptativo

En la Figura B.4 se observa la programación del control auxiliar propuesto, cuyas ganancias se detallan en la sección de simulaciones.

Apéndice B

Diagramas Simulink

Figura B.1 Diagrama de bloques en Simulink para controladores PD y SMC

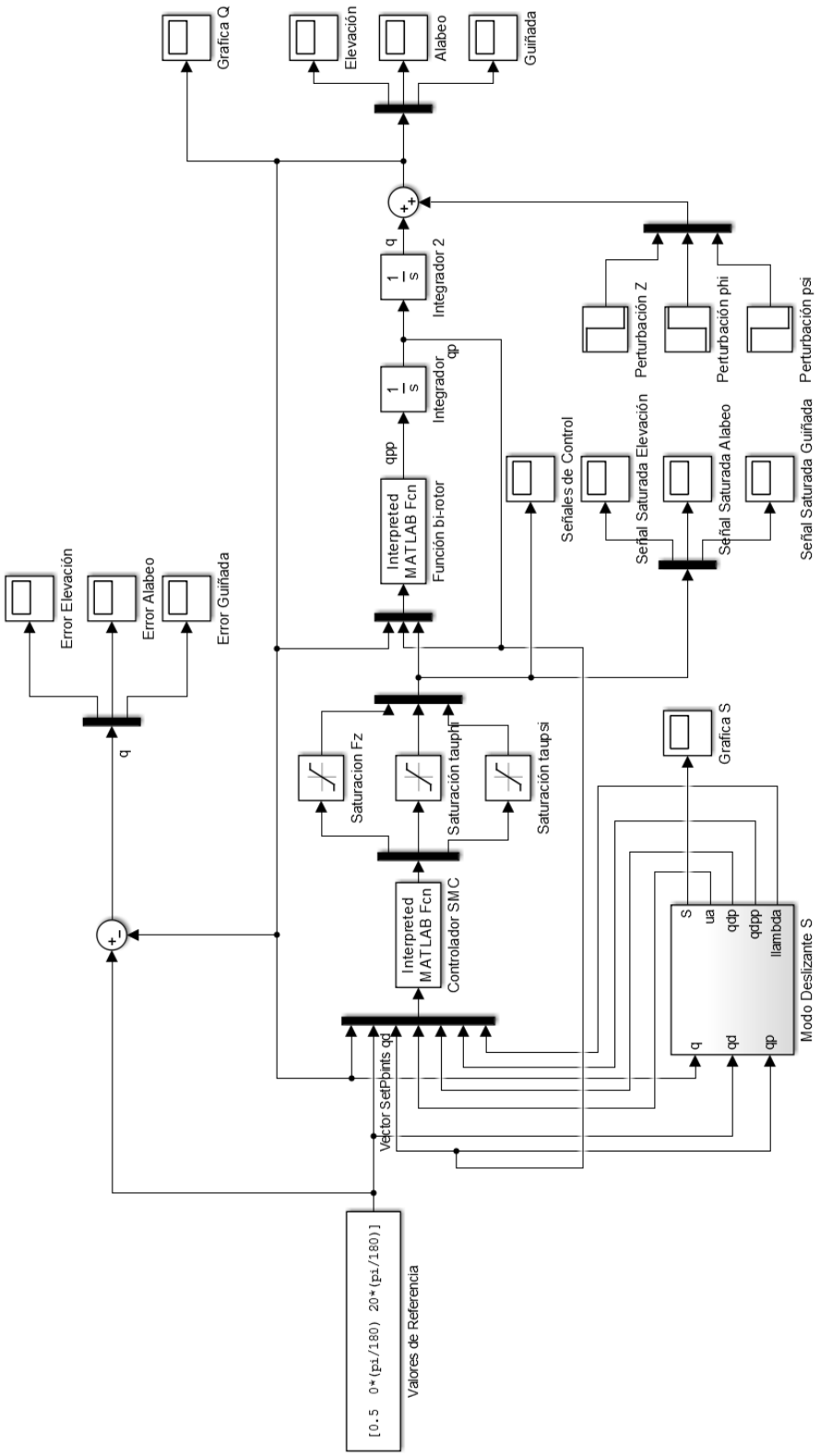

Figura B.2 Diagrama de bloques en Simulink para controladores ST y STA

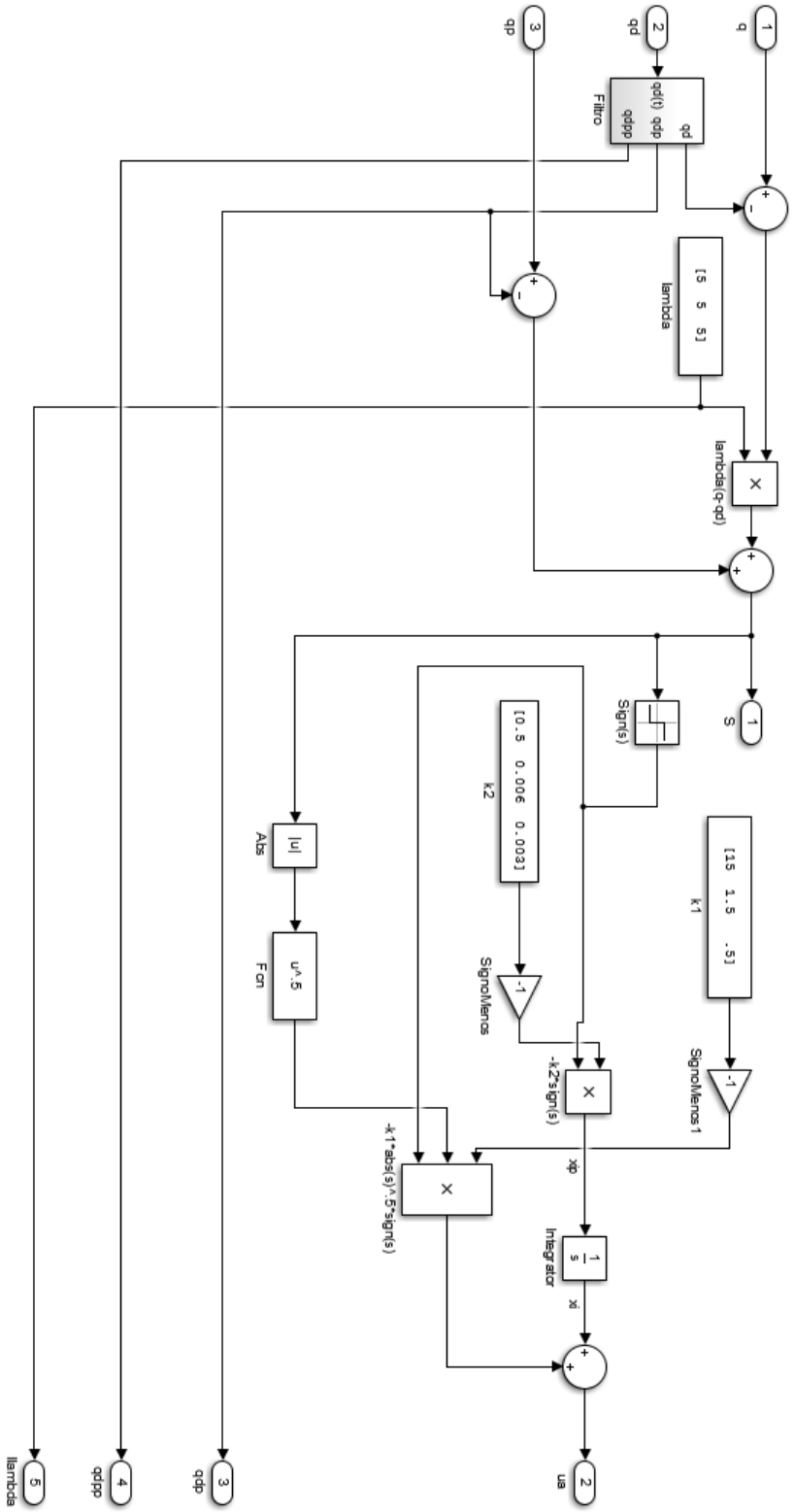

Figura B.3 Bloque de Superficie Deslizante para el controlador ST

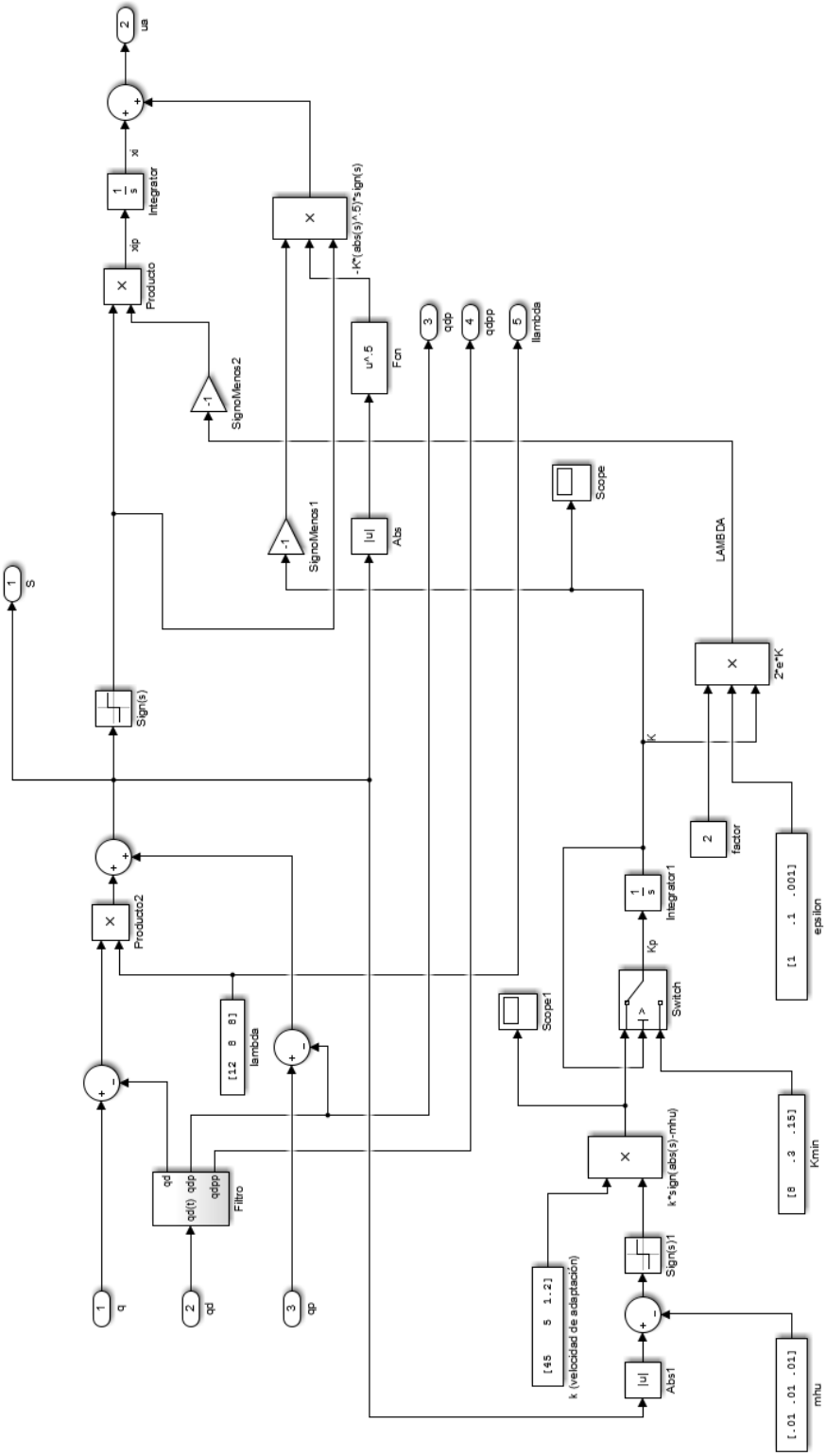

Figura B.4 Bloque de Superficie Deslizante para controlador STA

Apéndice C

Código para Arduino Mega

MATLAB Script C.1 Bi-Rotor Arduino ESP8266

```
#include <Servo.h>

String inputString = "";  // Dato de Entrada ESP8266
String PWM1 = "";         // PWM Motor 1
String PWM2 = "";         // PWM Motor 2
String tamano="";         // Tamano de cadena recibida
int tam=0;
int nPWM1=0;
int nPWM2=0;
boolean stringComplete = false;  // Indica la transmision completa del
    String
int Motor1 = 9;
int Motor2 = 10;

Servo ESC1;
Servo ESC2;

void setup()
{
  Serial1.begin(115200);
  inputString.reserve(200);
  tamano.reserve(20);
  PWM2.reserve(20);

  // INICIO BLOQUE DE MOTORES 1 y 2 //

```

```
26    ESC1.attach(9);
27    ESC2.attach(10);
28
29    ESC1.writeMicroseconds(2100);
30    ESC2.writeMicroseconds(2100);
31    delay(2000);
32    ESC1.writeMicroseconds(1000);
33    ESC2.writeMicroseconds(1000);
34    delay(2000);
35
36    // FIN DE BLOQUE DE MOTORES 1 y 2 //
37
38    //INICIO BLOQUE DE INICIO DE ESP8266 //
39    Serial1.println("AT+CIFSR");
40    delay(2000);
41    Serial1.println("AT+CWMODE=3");
42    delay(2000);
43    Serial1.println("AT+CIPMUX=1");
44    delay(2000);
45    Serial1.println("AT+CIPSERVER=1,80");
46    delay(2000);
47    //FIN BLOQUE DE INICIO DE ESP8266 //
48  }
49
50  void loop()
51  {
52    if (stringComplete)
53    {
54      int pos=inputString.indexOf(":a");
55      if (pos==9 || pos==10)
56      {
57        PWM1= inputString.substring(pos+2,pos+6);
58        PWM2= inputString.substring(pos+8,pos+12);
59      }
60      else
61      {
62        PWM1= inputString.substring(1,5);
63        PWM2= inputString.substring(7,11);
64      }
65
66      nPWM1=PWM1.toInt();
```

```
67      nPWM2=PWM2.toInt();
68
69      if (nPWM1>= 1000 && nPWM1<=2100)
70      {
71        ESC1.writeMicroseconds(nPWM1);
72      }
73      if (nPWM2>= 1000 && nPWM2<=2100)
74      {
75        ESC2.writeMicroseconds(nPWM2);
76      }
77      inputString = "";
78      stringComplete = false;
79    }
80  }
81
82  void serialEvent1()
83  {
84    while (Serial1.available())
85    {
86      char inChar = (char)Serial1.read();
87      inputString += inChar;
88      if (inChar == '\n')
89      {
90        stringComplete = true;
91      }
92    }
93  }
```

Apéndice D

Códigos en Matlab para controladores

D.1. Código de Controlador PD con compensación de gravedad

MATLAB Script D.1 MotiveBirotorPDt.m

```
function MotiveBiRotorPDt
    % ------------------------------------------------------------------ %
                            % ABRIR PUERTO A ESP8266 %

    % Create TCP/IP object 't'. Specify server machine and port number.
    t = tcpip('192.168.0.150', 80);

    % Open connection to the server.
    fopen(t);

                           % ABRIR PUERTO A ESP8266 %
    % ------------------------------------------------------------------ %

    fprintf( 'Streaming Camaras Bi-Rotor\n' )

    % create an instance of the natnet client class
    fprintf( 'Creando objeto en clase NatNet\n' )
    natnetclient = natnet;

```

```
20      % connect the client to the server (multicast over local loopback) -
21      % modify for your network
22      fprintf( 'Conectado al servidor (laptop)\n' )
23      natnetclient.HostIP = '127.0.0.1';
24      natnetclient.ClientIP = '127.0.0.1';
25      natnetclient.ConnectionType = 'Multicast';
26      natnetclient.connect;
27
28
29      % get the asset descriptions for the asset names
30      model = natnetclient.getModelDescription;
31      if ( model.RigidBodyCount < 1 )
32          return
33      end
34
35      % Codigo para obtener el vector de estados del Bi-Rotor mediante las
36      % camaras
37      %fprintf( '\nMostrando vector de estados del Bi-Rotor\n\n' )
38
39      % ------------------------------------------------------------------ %
40                  % Validacion de variables para obtener q punto %
41      frame_anterior=0;
42      frame_actual=0;
43
44      t_anterior=0;
45      t_actual=0;
46
47      elev_anterior=0;
48      elev_actual=0;
49
50      roll_anterior=0;
51      roll_actual=0;
52
53      pitch_anterior=0;
54      pitch_actual=0;
55
56      yaw_anterior=0;
57      yaw_actual=0;
58                % Validacion de variables para obtener q punto %
59      % ------------------------------------------------------------------ %
60      vector_t=[];
```

```
61      vector_elev=[];
62      vector_roll=[];
63      vector_yaw=[];
64      vector_tau1=[];
65      vector_tau2=[];
66      vector_tau3=[];
67
68      disp ('Iniciando Motores...')
69      for indini = 1100:2:1250
70          indexstr1 = int2str(indini+15);
71          indexstr2 = int2str(indini);
72          inicio = strcat ('a',indexstr1,'-b',indexstr2);
73          fprintf(t,inicio);
74          pause(.01);
75      end
76
77      for idx = 1 : 5000
78
79      % ------------------------------------------------------------- %
80           % Asignacion de diferencias entre q actual y anterior %
81          frame_anterior = frame_actual;
82          t_anterior=t_actual;
83          elev_anterior=elev_actual;
84          roll_anterior=roll_actual;
85          pitch_anterior=pitch_actual;
86          yaw_anterior=yaw_actual;
87           % Asignacion de diferencias entre q actual y anterior %
88      % ------------------------------------------------------------- %
89
90          data = natnetclient.getFrame; % OBTENCION DE CUADRO ACTUAL
91
92          frame_actual = data.Frame;
93          t_actual=data.Timestamp;
94
95          if (idx == 1)
96              t_inicial=t_actual;
97          end
98
99          if (frame_anterior ~= frame_actual)
100
101     % ------------------------------------------------------------- %
```

```
102             % Obtener componentes de quaternion y elevacion %
103
104         elev_actual = data.RigidBody(1).z;
105
106         q0 = data.RigidBody(1).qw;
107         q1 = data.RigidBody(1).qx;
108         q2 = data.RigidBody(1).qy;
109         q3 = data.RigidBody(1).qz;
110
111     % Obtener componentes de quaternion y elevacion %
112     % ------------------------------------------------------------- %
113
114     % ------------------------------------------------------------- %
115             % Transforacion de quaternion a ongulos en Rad %
116
117     roll_actual = asin (2 * ((q0*q2) - (q3*q1)));
118     pitch_actual = atan2( (2 * ( (q0*q1)+ (q2*q3) ) ),
119     1 - (2 * (q1^2 + q2^2)) );
120     yaw_actual = atan2( (2 * ( (q0*q3)+ (q1*q2) ) ),
121     1 - (2 * (q2^2 + q3^2)));
122
123             % Transformacion de quaternion a ongulos en Rad %
124     % ------------------------------------------------------------- %
125
126         dif_tiempo = (t_actual - t_anterior);
127         dif_elev = (elev_actual - elev_anterior);
128         dif_roll = (roll_actual - roll_anterior);
129         dif_yaw = (yaw_actual - yaw_anterior);
130
131         vel_z = dif_elev / dif_tiempo;
132         vel_roll = dif_roll / dif_tiempo;
133         vel_yaw = dif_yaw / dif_tiempo;
134
135         vector_q=[elev_actual; roll_actual; yaw_actual];
136         vector_qp =[vel_z; vel_roll; vel_yaw];
137         c_roll = cos (roll_actual);
138
139     % ------------------------------------------------------------- %
140                 % C O N T R O L A D O R  PD+%
141     % ------------------------------------------------------------- %
142
```

```
143                 q= vector_q;
144                 qd=[0.80; 0 * (pi / 180) ; 0 * (pi / 180)];
145
146
147                 qp= vector_qp;
148                 qe=qd-q;
149
150                 Kp=[40 0 0; 0 0.5 0; 0 0 .050];
151                 Kv=[12 0 0;0 0.18 0;0 0 .01];
152                 tau= (Kp*qe - Kv*qp + [9.81*1; 0 ; 0 ])
153                 .* [1/c_roll;1;1] ;
154
155                 % SATURACION DE GANANCIAS DE CONTROL %
156
157                 if (tau(1)>36)
158                      tau(1)=36;
159                 end
160
161                 if (tau(2)>6.12)
162                      tau(2)=6.12;
163                 end
164
165                 if (tau(3)>1)
166                      tau(3)=1;
167                 end
168
169                 if (tau(3)<-1)
170                      tau(3)=-1;
171                 end
172       % ------------------------------------------------------------- %
173                       % C O N T R O L A D O R  PD+%
174       % ------------------------------------------------------------- %
175
176       % ------------------------------------------------------------- %
177                  % AJUSTE PARA ENVIO DE DATOS A MOTORES %
178
179                  elevPWM= (tau(1)/0.018)/2;
180                  rollPWM= (tau(2)/0.00580);
181
182                  yawPWM= (tau(3)/0.0170);
183
```

```
184                 pwm1=int16(1015 + elevPWM + (rollPWM/2) + (yawPWM));
185                 pwm2=int16(1000 + elevPWM - (rollPWM/2) - (yawPWM));
186
187                 if (pwm1>=2000)
188                     pwm1 = 2000;
189                 end
190                 if (pwm1<=1000)
191                     pwm1 = 1000;
192                 end
193
194                 if (pwm2>=2000)
195                     pwm2=2000;
196                 end
197                 if (pwm2<=1000)
198                     pwm2=1000;
199                 end
200
201               % AJUSTE PARA ENVIO DE DATOS A MOTORES %
202     % ------------------------------------------------------------- %
203
204     % ------------------------------------------------------------- %
205                       % ENVIO DE DATOS A MOTORES %
206
207               chr1 = int2str(pwm1);
208               chr2 = int2str(pwm2);
209               cadena = strcat ('a',chr1,'-b',chr2);
210               fprintf(t,cadena);
211               %fprintf(cadena);
212               %fprintf('\n');
213
214                       % ENVIO DE DATOS A MOTORES %
215     % ------------------------------------------------------------- %
216
217               vector_t(end+1)= data.Timestamp;
218               vector_roll(end+1)= vector_q(2) * (180/pi);
219               vector_elev(end+1)= vector_q(1);
220               vector_yaw(end+1)= vector_q(3) * (180/pi);
221               vector_tau1(end+1)= tau(1);
222               vector_tau2(end+1)= tau(2);
223               vector_tau3(end+1)= tau(3);
224
```

```
225                 fprintf( 'Frame:%6d  ' , data.Frame )
226                 fprintf( 'Time:%0.3f\n' , data.Timestamp )
227                 fprintf( 'Vector q= %.5f \n', vector_q );
228                 fprintf( 'Vector qp= %.5f \n', vector_qp );
229                 fprintf( 'Control tau: %.2f \n', tau);
230                 fprintf( 'PWM1: %i \n', pwm1);
231                 fprintf( 'PWM2: %i \n', pwm2);
232                 fprintf( '\n')
233             end
234     end
235     disp ('Descenso')
236
237 for ind = 1650:-1:1100
238     indexstr = int2str(ind);
239     cadena = strcat ('a',indexstr,'-b',indexstr);
240     fprintf(t,cadena);
241     pause(.025);
242 end
243
244 fprintf(t,'a1000-b1000');
245 delete(t);
246 clear t
247 clear pwm1
248 clear pwm2
249
250 fecha = datetime('now','Format','dd-MM-yy-HH_mm_ss');
251 fecha = char (fecha);
252 nombre=['Prueba_PDt_' fecha '.mat'];
253
254 titulo1 = strcat ('Controlador PDt Vector Q _',fecha);
255 titulo2 = strcat ('Controlador PDt Vector TAU _',fecha);
256
257 set(gcf,'Name',titulo1);
258 subplot(3,1,1);
259 plot (vector_t - t_inicial, vector_elev, 'LineWidth', 1)
260 %xlabel('Tiempo (seg)','Interpreter','latex','FontSize',16);
261 ylabel('z (m)','Interpreter','latex','FontSize',16);
262 %title('Evolucion temporal variables de estado','Interpreter','latex',
263 'FontSize',20);
264 ylim([0 1.2])
265 grid on;
```

```
266
267 subplot(3,1,2);
268 plot (vector_t - t_inicial, vector_roll, 'LineWidth', 1)
269  %xlabel('Tiempo (seg)','Interpreter','latex','FontSize',16);
270 ylabel('$\phi$ (grados)','Interpreter','latex','FontSize',16);
271  %title('$\phi$','Interpreter','latex','FontSize',20);
272 ylim([-50 50])
273 grid on;
274
275 subplot(3,1,3);
276 plot (vector_t - t_inicial, vector_yaw, 'LineWidth', 1)
277 xlabel('Tiempo (seg)','Interpreter','latex','FontSize',16);
278 ylabel('$\psi$ (grados)','Interpreter','latex','FontSize',16);
279  %title('$\psi$','Interpreter','latex','FontSize',20);
280 ylim([-180 180])
281 grid on;
282
283 figure();
284 set(gcf,'Name',titulo2);
285 subplot(3,1,1);
286 plot(vector_t - t_inicial, vector_tau1, 'LineWidth', 1)
287  %xlabel('Tiempo (seg)','Interpreter','latex','FontSize',16);
288 ylabel('$u$ (N)','Interpreter','latex','FontSize',16);
289  %title('$u$','Interpreter','latex','FontSize',20);
290 ylim([0 36])
291 grid on;
292
293 subplot(3,1,2);
294 plot(vector_t - t_inicial, vector_tau2, 'LineWidth', 1)
295  %xlabel('Tiempo (seg)','Interpreter','latex','FontSize',16);
296 ylabel('$\tau_\phi$ (N·m)','Interpreter','latex','FontSize',16);
297  %title('$\tau_\phi$','Interpreter','latex','FontSize',20);
298 ylim([-7 7])
299 grid on;
300
301 subplot(3,1,3);
302 plot(vector_t - t_inicial, vector_tau3, 'LineWidth', 1)
303 xlabel('Tiempo (seg)','Interpreter','latex','FontSize',16);
304 ylabel('$\tau_\psi$(N·m)','Interpreter','latex','FontSize',16);
305  %title('$\tau_\psi$','Interpreter','latex','FontSize',20);
306 ylim([-2 2])
```

```
grid on;


save(nombre,'Kp','Kv', 'vector_t','vector_elev','vector_roll',
'vector_yaw',
'vector_tau1','vector_tau2','vector_tau3')
disp('Programa terminado' )
end
```

D.2. Código de Controlador SMC

MATLAB Script D.2 MotiveBirotorSMC.m

```

function MotiveBiRotorSMCBueno
    % ------------------------------------------------------------- %
                          % ABRIR PUERTO A ESP8266 %

% Create TCP/IP object 't'. Specify server m\'achine and port number.
t = tcpip('192.168.0.150', 80);

% Open connection to the server.
fopen(t);

                          % ABRIR PUERTO A ESP8266 %
    % ------------------------------------------------------------- %

fprintf( 'Streaming Camaras Bi-Rotor\n' )

% create an instance of the natnet client class
fprintf( 'Creando objeto en clase NatNet\n' )
natnetclient = natnet;

% connect the client to the server (multicast over local loopback) -
% modify for your network
fprintf( 'Conectado al servidor (laptop)\n' )
natnetclient.HostIP = '127.0.0.1';
natnetclient.ClientIP = '127.0.0.1';
natnetclient.ConnectionType = 'Multicast';
natnetclient.connect;
```

```
28
29
30 % get the asset descriptions for the asset names
31 model = natnetclient.getModelDescription;
32 if ( model.RigidBodyCount < 1 )
33 return
34 end
35
36 % C\'odigo para obtener el vector de estados del Bi-Rotor mediante las
37 % camaras
38 %fprintf( '\nMostrando vector de estados del Bi-Rotor\n\n' )
39
40     % ------------------------------------------------------------- %
41               % Validacion de variables para obtener q punto %
42 frame_anterior=0;
43 frame_actual=0;
44
45 t_anterior=0;
46 t_actual=0;
47
48 elev_anterior=0;
49 elev_actual=0;
50
51 roll_anterior=0;
52 roll_actual=0;
53
54 pitch_anterior=0;
55 pitch_actual=0;
56
57 yaw_anterior=0;
58 yaw_actual=0;
59               % Validacion de variables para obtener q punto %
60     % ------------------------------------------------------------- %
61 vector_t=[];
62 vector_elev=[];
63 vector_roll=[];
64 vector_yaw=[];
65 vector_tau1=[];
66 vector_tau2=[];
67 vector_tau3=[];
68
```

```
69 disp ('Iniciando Motores...')
70 for indini = 1100:1:1600
71 indexstr1 = int2str(indini+15);
72 indexstr2 = int2str(indini);
73 inicio = strcat ('a',indexstr1,'-b',indexstr2);
74 fprintf(t,inicio);
75 pause(.01);
76 end
77
78 for idx = 1 : 5000
79
80     % ------------------------------------------------------------- %
81          % Asignacion de diferencias entre q actual y anterior %
82 frame_anterior = frame_actual;
83 t_anterior=t_actual;
84 elev_anterior=elev_actual;
85 roll_anterior=roll_actual;
86 pitch_anterior=pitch_actual;
87 yaw_anterior=yaw_actual;
88          % Asignacion de diferencias entre q actual y anterior %
89     % ------------------------------------------------------------- %
90
91
92 data = natnetclient.getFrame; % OBTENCION DE CUADRO ACTUAL
93
94 frame_actual = data.Frame;
95 t_actual=data.Timestamp;
96
97 if (idx == 1)
98     t_inicial=t_actual;
99 end
100
101
102 if (frame_anterior ~= frame_actual)
103
104     % ------------------------------------------------------------- %
105              % Obtener componentes de quaternion y elevacion %
106
107 elev_actual = data.RigidBody(1).z;
108
109 q0 = data.RigidBody(1).qw;
```

```
110 q1 = data.RigidBody(1).qx;
111 q2 = data.RigidBody(1).qy;
112 q3 = data.RigidBody(1).qz;
113
114             % Obtener componentes de quaternion y elevacion %
115     % ------------------------------------------------------------- %
116
117     % ------------------------------------------------------------- %
118             % Transformacion de quaternion a angulos en Rad %
119
120 roll_actual = asin (2 * ((q0*q2) - (q3*q1)));
121 pitch_actual = atan2( (2 * ( (q0*q1)+ (q2*q3) ) ),
122 1 - (2 * (q1^2 + q2^2)) );
123 yaw_actual = atan2( (2 * ( (q0*q3)+ (q1*q2) ) ),
124 1 - (2 * (q2^2 + q3^2)));
125
126             % Transformacion de quaternion a angulos en Rad %
127     % ------------------------------------------------------------- %
128
129 dif_tiempo = (t_actual - t_anterior);
130 dif_elev = (elev_actual - elev_anterior);
131 dif_roll = (roll_actual - roll_anterior);
132 dif_yaw = (yaw_actual - yaw_anterior);
133
134 vel_z = dif_elev / dif_tiempo;
135 vel_roll = dif_roll / dif_tiempo;
136 vel_yaw = dif_yaw / dif_tiempo;
137
138 vector_q=[elev_actual; roll_actual; yaw_actual];
139 vector_qp =[vel_z; vel_roll; vel_yaw];
140 c_roll = cos (roll_actual);
141 s_roll = sin (roll_actual);
142
143     % ------------------------------------------------------------- %
144                         % C O N T R O L A D O R   S M C %
145     % ------------------------------------------------------------- %
146
147 q= vector_q;
148 qd=[0.70; 0 * (pi / 180) ; 0 * (pi / 180)];
149 qp= vector_qp;
150 qe=qd-q;
```

```
151 qdp=[0;0;0];                    %Vector de velocidades en SetPoints
152 qdpp=[0;0;0];                   %Vector de aceleraciones en SetPoints
153
154 k=[2 0 0;0 .08 0;0 0 .1];      %Ganancia de Control
155 lambda=[5 0 0;0 8 0;0 0 1];     % Ganancia de q-qd (Rapidez de
156 Convergencia a S)
157
158 s=qp-qdp+lambda*(q-qd);         %Fase deslizante
159 ua=-k*sign(s);                  %Control Auxilizar -> Modos deslizantes <-
160
161 kf1=0.01;         %Constante de friccion de altitud.
162 kf2=0.03;         %Constante de friccion del alabeo.
163 kf3=0.0065;         %Constante de friccion de guinada.
164
165 m=2;            %Masa del sistema.
166 Ix=0.043;          %Inercia de roll (Obtenidas experimentalmente)
167 Iz=0.041;          %Inercia del yaw (Obtenidas experimentalmente)
168
169 c_roll;    %Coseno de phi
170 s_roll;    %Seno de phi
171
172 M=[m 0 0; 0 Ix 0; 0 0 Iz*c_roll^2];
173 %Matriz de masas e inercias
174 Cqp=[0;Iz*c_roll*s_roll*qp(3)^2;-2*Iz*c_roll*s_roll*qp(2)*qp(3)];
175 %Vector de coriolis
176 f=[kf1 0 0; 0 kf2 0; 0 0 kf3]*qp;
177 %Vector de fricciones
178 g=[9.81*m; 0; 0];
179 %Vector de gravedad
180
181 tau= M *(qdpp-lambda*(qp-qdp)) + Cqp + g + f + ua;  %Salida tau
182
183 % SATURACION DE GANANCIAS DE CONTROL %
184
185 if (tau(1)>36)
186      tau(1)=36;
187 end
188
189 if (tau(2)>6.12)
190      tau(2)=6.12;
191 end
```

```
192
193 if (tau(3)>1.73)
194      tau(3)=1.73;
195 end
196 % ------------------------------------------------------------- %
197                       % C O N T R O L A D O R  S M C%
198 % ------------------------------------------------------------- %
199
200 % ------------------------------------------------------------- %
201                % AJUSTE PARA ENVIO DE DATOS A MOTORES %
202
203  elevPWM= (tau(1)/0.013)/2;
204  rollPWM= (tau(2)/0.00612);
205
206  pwm1=int16(1015 + elevPWM + (rollPWM/2));
207  pwm2=int16(1000 + elevPWM - (rollPWM/2));
208
209  if (pwm1>=2000)
210      pwm1 = 2000;
211  end
212  if (pwm1<=1000)
213      pwm1 = 1000;
214  end
215
216  if (pwm2>=2000)
217      pwm2=2000;
218  end
219  if (pwm2<=1000)
220      pwm2=1000;
221  end
222
223                % AJUSTE PARA ENVIO DE DATOS A MOTORES %
224 % ------------------------------------------------------------- %
225
226 % ------------------------------------------------------------- %
227                       % ENVIO DE DATOS A MOTORES %
228 chr1 = int2str(pwm1);
229 chr2 = int2str(pwm2);
230 cadena = strcat ('a',chr1,'-b',chr2);
231 fprintf(t,cadena);
232 %fprintf(cadena);
```

```
233 %fprintf('\n');
234                         % ENVIO DE DATOS A MOTORES %
235 % ------------------------------------------------------------- %
236
237 vector_t(end+1)= data.Timestamp;
238 vector_roll(end+1)= vector_q(2) * (180/pi);
239 vector_elev(end+1)= vector_q(1);
240 vector_yaw(end+1)= vector_q(3) * (180/pi);
241 vector_tau1(end+1)= tau(1);
242 vector_tau2(end+1)= tau(2);
243 vector_tau3(end+1)= tau(3);
244
245 fprintf( 'Frame:%6d  ' , data.Frame )
246 fprintf( 'Time:%0.3f\n' , data.Timestamp )
247 fprintf( 'Vector q= %.5f \n', vector_q );
248 fprintf( 'Vector qp= %.5f \n', vector_qp );
249 fprintf( 'Control tau: %.2f \n', tau);
250 fprintf( 'Cos(phi): %.2f \n', c_roll);
251 fprintf( 'PWM1: %i \n', pwm1);
252 fprintf( 'PWM2: %i \n', pwm2);
253 fprintf( '\n')
254 end
255 end
256
257 disp ('Descenso')
258
259
260
261 for ind = 1650:-1:1100
262 indexstr = int2str(ind);
263 cadena = strcat ('a',indexstr,'-b',indexstr);
264 fprintf(t,cadena);
265 pause(.025);
266 end
267
268 fprintf(t,'a1000-b1000');
269 delete(t);
270 clear t
271 clear pwm1
272 clear pwm2
273
```

```
274 fecha = datetime('now','Format','dd-MM-yy-HH_mm_ss');
275 fecha = char (fecha);
276 nombre=['Prueba_ST_' fecha '.mat'];
277
278 titulo1 = strcat ('Controlador ST Vector Q _',fecha);
279 titulo2 = strcat ('Controlador ST Vector TAU _',fecha);
280
281 set(gcf,'Name',titulo1);
282 subplot(3,1,1);
283 plot (vector_t - t_inicial, vector_elev, 'LineWidth', 1)
284 %xlabel('Tiempo (seg)','Interpreter','latex','FontSize',16);
285 ylabel('z (m)','Interpreter','latex','FontSize',16);
286 %title('Evolucion temporal variables de estado','Interpreter','latex',
287 'FontSize',20);
288 ylim([0 1.2])
289 grid on;
290
291 subplot(3,1,2);
292 plot (vector_t - t_inicial, vector_roll, 'LineWidth', 1)
293 %xlabel('Tiempo (seg)','Interpreter','latex','FontSize',16);
294 ylabel('phi (grados)','Interpreter','latex','FontSize',16);
295 %title('phi','Interpreter','latex','FontSize',20);
296 ylim([-50 50])
297 grid on;
298
299 subplot(3,1,3);
300 plot (vector_t - t_inicial, vector_yaw, 'LineWidth', 1)
301 xlabel('Tiempo (seg)','Interpreter','latex','FontSize',16);
302 ylabel('psi (grados)','Interpreter','latex','FontSize',16);
303 %title('psi','Interpreter','latex','FontSize',20);
304 ylim([-180 180])
305 grid on;
306
307
308 figure();
309 set(gcf,'Name',titulo2);
310 subplot(3,1,1);
311 plot(vector_t - t_inicial, vector_tau1, 'LineWidth', 1)
312 %xlabel('Tiempo (seg)','Interpreter','latex','FontSize',16);
313 ylabel('u (N)','Interpreter','latex','FontSize',16);
314 %title('u','Interpreter','latex','FontSize',20);
```

```
315 ylim([0 36])
316 grid on;
317
318 subplot(3,1,2);
319 plot(vector_t - t_inicial, vector_tau2, 'LineWidth', 1)
320 %xlabel('Tiempo (seg)','Interpreter','latex','FontSize',16);
321 ylabel('τϕ (N·m)','Interpreter','latex','FontSize',16);
322 %title('τϕ','Interpreter','latex','FontSize',20);
323 ylim([-7 7])
324 grid on;
325
326 subplot(3,1,3);
327 plot(vector_t - t_inicial, vector_tau3, 'LineWidth', 1)
328 xlabel('Tiempo (seg)','Interpreter','latex','FontSize',16);
329 ylabel('τψ (N·m)','Interpreter','latex','FontSize',16);
330 %title('τψ','Interpreter','latex','FontSize',20);
331 ylim([-2 2])
332 grid on;
333
334
335 save(nombre,'k','lambda', 'vector_t','vector_elev','vector_roll',
336 'vector_yaw','vector_tau1',
337 'vector_tau2','vector_tau3')
338 disp('Programa terminado' )
339 end
```

D.3. Código de Controlador ST

MATLAB Script D.3 MotiveBirotorST.m

```
1
2 function MotiveBiRotorSOST
3 % ------------------------------------------------------------- %
4                     % ABRIR PUERTO A ESP8266 %
5
6 % Create TCP/IP object 't'. Specify server machine and port number.
7 t = tcpip('192.168.0.150', 80);
8
9 % Open connection to the server.
10 fopen(t);
```

```
11
12                      % ABRIR PUERTO A ESP8266 %
13 % ------------------------------------------------------------- %
14
15 fprintf( 'Streaming Camaras Bi-Rotor\n' )
16
17 % create an instance of the natnet client class
18 fprintf( 'Creando objeto en clase NatNet\n' )
19 natnetclient = natnet;
20
21 % connect the client to the server (multicast over local loopback) -
22 % modify for your network
23 fprintf( 'Conectado al servidor (laptop)\n' )
24 natnetclient.HostIP = '127.0.0.1';
25 natnetclient.ClientIP = '127.0.0.1';
26 natnetclient.ConnectionType = 'Multicast';
27 natnetclient.connect;
28
29
30 % get the asset descriptions for the asset names
31 model = natnetclient.getModelDescription;
32 if ( model.RigidBodyCount < 1 )
33 return
34 end
35
36 % Codigo para obtener el vector de estados del Bi-Rotor mediante las
37 % camaras
38 %fprintf( '\nMostrando vector de estados del Bi-Rotor\n\n' )
39
40 % ------------------------------------------------------------- %
41           % Validacion de variables para obtener q punto %
42 frame_anterior=0;
43 frame_actual=0;
44
45 t_anterior=0;
46 t_actual=0;
47
48 elev_anterior=0;
49 elev_actual=0;
50
51 roll_anterior=0;
```

```
52 roll_actual=0;
53
54 pitch_anterior=0;
55 pitch_actual=0;
56
57 yaw_anterior=0;
58 yaw_actual=0;
59
60 xip_actual = [0; 0; 0];
61 xi_actual = [0; 0; 0];
62          % Validacion de variables para obtener q punto %
63 % ------------------------------------------------------------- %
64 vector_t=[];
65 vector_elev=[];
66 vector_roll=[];
67 vector_yaw=[];
68 vector_tau1=[];
69 vector_tau2=[];
70 vector_tau3=[];
71
72 disp ('Iniciando Motores...')
73 for indini = 1100:1:1600
74 indexstr1 = int2str(indini+15);
75 indexstr2 = int2str(indini);
76 inicio = strcat ('a',indexstr1,'-b',indexstr2);
77 fprintf(t,inicio);
78 pause(.01);
79 end
80
81 for idx = 1 : 12000
82
83 % ------------------------------------------------------------- %
84      % Asignacion de diferencias entre q actual y anterior %
85 frame_anterior = frame_actual;
86 t_anterior=t_actual;
87 elev_anterior=elev_actual;
88 roll_anterior=roll_actual;
89 pitch_anterior=pitch_actual;
90 yaw_anterior=yaw_actual;
91      % Asignacion de diferencias entre q actual y anterior %
92 % ------------------------------------------------------------- %
```

```
93
94
95  data = natnetclient.getFrame;  % OBTENCION DE CUADRO ACTUAL
96
97  frame_actual = data.Frame;
98  t_actual=data.Timestamp;
99
100 if (idx == 1)
101 t_inicial=t_actual;
102 end
103
104
105 if (frame_anterior ~= frame_actual)
106
107     % ------------------------------------------------------------- %
108             % Obtener componentes de quaternion y elevacion %
109
110 elev_actual = data.RigidBody(1).z;
111
112 q0 = data.RigidBody(1).qw;
113 q1 = data.RigidBody(1).qx;
114 q2 = data.RigidBody(1).qy;
115 q3 = data.RigidBody(1).qz;
116
117             % Obtener componentes de quaternion y elevacion %
118     % ------------------------------------------------------------- %
119
120     % ------------------------------------------------------------- %
121             % Transformacion de quaternion a angulos en Rad %
122
123 roll_actual = asin (2 * ((q0*q2) - (q3*q1)));
124 pitch_actual = atan2( (2 * ( (q0*q1)+ (q2*q3) ) ),
125 1 - (2 * (q1^2 + q2^2)) );
126 yaw_actual = atan2( (2 * ( (q0*q3)+ (q1*q2) ) ),
127 1 - (2 * (q2^2 + q3^2)));
128
129             % Transformacion de quaternion a angulos en Rad %
130     % ------------------------------------------------------------- %
131
132 dif_tiempo = (t_actual - t_anterior);
133 dif_elev = (elev_actual - elev_anterior);
```

```
134 dif_roll = (roll_actual - roll_anterior);
135 dif_yaw = (yaw_actual - yaw_anterior);
136
137 vel_z = dif_elev / dif_tiempo;
138 vel_roll = dif_roll / dif_tiempo;
139 vel_yaw = dif_yaw / dif_tiempo;
140
141 vector_q=[elev_actual; roll_actual; yaw_actual];
142 vector_qp =[vel_z; vel_roll; vel_yaw];
143 c_roll = cos (roll_actual);
144 s_roll = sin (roll_actual);
145
146 % ------------------------------------------------------------- %
147                     % C O N T R O L A D O R   S O S M C %
148 % ------------------------------------------------------------- %
149
150     q= vector_q;
151     qd=[0.70; 0 * (pi / 180) ; 0 * (pi / 180)];
152     qp= vector_qp;
153     qe=qd-q;
154     qdp=[0;0;0];
155     %Vector de velocidades en SetPoints
156     qdpp=[0;0;0];
157     %Vector de aceleraciones en SetPoints
158
159     lambda=[1.15; .90;.1];
160     % Ganancia de q-qd (Rapidez de Convergencia a S)
161
162     s=qp-qdp+lambda.*(q-qd);                    %Fase deslizante
163
164     % ELEMENTOS DE CONTROLADOR DE SEGUNDO ORDEN
165     % ----------------------------------------
166
167     K_1=[1.4 ; .150 ; .001];                %Ganancia de Control
168     K_2=[.015 ; .0008 ; .001];                %Ganancia de Control
169
170
171     xip_actual= -K_2 .* sign(s);
172     xi_actual = xi_actual + xip_actual;
173
174     abs_s=abs(s);
```

```
175     ua= -K_1 .*((abs_s).^(1/2)) .* sign(s) + xi_actual;
176     %Control Auxilizar -> Modos deslizantes <-
177
178     % ---------------------------------------
179     % ELEMENTOS DE CONTROLADOR DE SEGUNDO ORDEN
180
181
182
183     kf1=0.01;         %Constante de friccion de altitud.
184     kf2=0.03;         %Constante de friccion del alabeo.
185     kf3=0.0065;         %Constante de friccion de guinada.
186
187     m=1.84;            %Masa del sistema.
188     Ix=0.043;          %Inercia de roll (Obtenidas experimentalmente)
189     Iz=0.041;          %Inercia del yaw (Obtenidas experimentalmente)
190
191     c_roll;    %Coseno de phi
192     s_roll;    %Seno de phi
193
194     M=[m 0 0; 0 Ix 0; 0 0 Iz*c_roll^2];
195     %Matriz de masas e inercias
196     Cqp=[0;Iz*c_roll*s_roll*qp(3)^2;-2*Iz*c_roll*s_roll*qp(2)*qp(3)];
197     %Vector de coriolis
198     f=[kf1 0 0; 0 kf2 0; 0 0 kf3]*qp;
199     %Vector de fricciones
200     g=[9.81*m; 0; 0];
201     %Vector de gravedad
202
203     tau= M *(qdpp-lambda.*(qp-qdp)) + Cqp + g + f + ua;   %Salida tau
204
205     % SATURACION DE GANANCIAS DE CONTROL %
206
207     if (tau(1)>36)
208          tau(1)=36;
209     end
210
211     if (tau(2)>6.12)
212          tau(2)=6.12;
213     end
214
215     if (tau(3)>1.73)
```

```
216         tau(3)=1.73;
217     end
218     % ------------------------------------------------------------- %
219                         % C O N T R O L A D O R  S O S M C%
220     % ------------------------------------------------------------- %
221
222     % ------------------------------------------------------------- %
223                     % AJUSTE PARA ENVIO DE DATOS A MOTORES %
224
225      elevPWM= (tau(1)/0.012)/2;
226      rollPWM= (tau(2)/0.00612);
227
228      pwm1=int16(1015 + elevPWM + (rollPWM/2));
229      pwm2=int16(1000 + elevPWM - (rollPWM/2));
230
231      if (pwm1>=2000)
232          pwm1 = 2000;
233      end
234      if (pwm1<=1000)
235          pwm1 = 1000;
236      end
237
238      if (pwm2>=2000)
239          pwm2=2000;
240      end
241      if (pwm2<=1000)
242          pwm2=1000;
243      end
244
245                     % AJUSTE PARA ENVIO DE DATOS A MOTORES %
246     % ------------------------------------------------------------- %
247
248
249
250     % ------------------------------------------------------------- %
251                         % ENVIO DE DATOS A MOTORES %
252     chr1 = int2str(pwm1);
253     chr2 = int2str(pwm2);
254     cadena = strcat ('a',chr1,'-b',chr2);
255     fprintf(t,cadena);
256     %fprintf(cadena);
```

```
257      %fprintf('\n');
258                              % ENVIO DE DATOS A MOTORES %
259      % ------------------------------------------------------------- %
260
261      vector_t(end+1)= data.Timestamp;
262      vector_roll(end+1)= vector_q(2) * (180/pi);
263      vector_elev(end+1)= vector_q(1);
264      vector_yaw(end+1)= vector_q(3) * (180/pi);
265      vector_tau1(end+1)= tau(1);
266      vector_tau2(end+1)= tau(2);
267      vector_tau3(end+1)= tau(3);
268
269      fprintf( 'Frame:%6d  ' , data.Frame )
270      fprintf( 'Time:%0.3f\n' , data.Timestamp )
271      fprintf( 'Vector q= %.5f \n', vector_q );
272      fprintf( 'Vector qp= %.5f \n', vector_qp );
273      fprintf( 'Control tau: %.2f \n', tau);
274      fprintf( 'Cos(phi): %.2f \n', c_roll);
275      fprintf( 'PWM1: %i \n', pwm1);
276      fprintf( 'PWM2: %i \n', pwm2);
277      fprintf( '\n')
278  end
279  end
280
281  disp ('Descenso')
282
283
284
285  for ind = 1650:-1:1100
286  indexstr = int2str(ind);
287  cadena = strcat ('a',indexstr,'-b',indexstr);
288  fprintf(t,cadena);
289  pause(.025);
290  end
291
292  fprintf(t,'a1000-b1000');
293  delete(t);
294  clear t
295  clear pwm1
296  clear pwm2
297
```

```
298 fecha = datetime('now','Format','dd-MM-yy-HH_mm_ss');
299 fecha = char (fecha);
300 nombre=['Prueba_SOST_' fecha '.mat'];
301
302 titulo1 = strcat ('Controlador SOST Vector Q _',fecha);
303 titulo2 = strcat ('Controlador SOST Vector TAU _',fecha);
304
305 set(gcf,'Name',titulo1);
306 subplot(3,1,1);
307 plot (vector_t - t_inicial, vector_elev, 'LineWidth', 1)
308 %xlabel('Tiempo (seg)','Interpreter','latex','FontSize',16);
309 ylabel('z (m)','Interpreter','latex','FontSize',16);
310 %title('Evolucion temporal variables de estado',
311 'Interpreter','latex','FontSize',20);
312 ylim([0 1.2])
313 grid on;
314
315 subplot(3,1,2);
316 plot (vector_t - t_inicial, vector_roll, 'LineWidth', 1)
317 %xlabel('Tiempo (seg)','Interpreter','latex','FontSize',16);
318 ylabel('\phi (grados)','Interpreter','latex','FontSize',16);
319 %title('\phi','Interpreter','latex','FontSize',20);
320 ylim([-50 50])
321 grid on;
322
323 subplot(3,1,3);
324 plot (vector_t - t_inicial, vector_yaw, 'LineWidth', 1)
325 xlabel('Tiempo (seg)','Interpreter','latex','FontSize',16);
326 ylabel('\psi (grados)','Interpreter','latex','FontSize',16);
327 %title('\psi','Interpreter','latex','FontSize',20);
328 ylim([-180 180])
329 grid on;
330
331
332 figure();
333 set(gcf,'Name',titulo2);
334 subplot(3,1,1);
335 plot(vector_t - t_inicial, vector_tau1, 'LineWidth', 1)
336 %xlabel('Tiempo (seg)','Interpreter','latex','FontSize',16);
337 ylabel('u (N)','Interpreter','latex','FontSize',16);
338 %title('u','Interpreter','latex','FontSize',20);
```

```
ylim([0 36])
grid on;

subplot(3,1,2);
plot(vector_t - t_inicial, vector_tau2, 'LineWidth', 1)
%xlabel('Tiempo $(seg)$','Interpreter','latex','FontSize',16);
ylabel('$\tau_\phi$ (N·m)','Interpreter','latex','FontSize',16);
%title('$\tau_\phi$','Interpreter','latex','FontSize',20);
ylim([-7 7])
grid on;

subplot(3,1,3);
plot(vector_t - t_inicial, vector_tau3, 'LineWidth', 1)
xlabel('Tiempo $(seg)$','Interpreter','latex','FontSize',16);
ylabel('$\tau_\psi$ (N·m)','Interpreter','latex','FontSize',16);
%title('$\tau_\psi$','Interpreter','latex','FontSize',20);
ylim([-2 2])
grid on;


save(nombre,'K_1','K_2','lambda', 'vector_t','vector_elev','vector_roll',
'vector_yaw','vector_tau1','vector_tau2','vector_tau3')
disp('Programa terminado' )
end
```

D.4. Código de Controlador STA

MATLAB Script D.4 MotiveBirotorSTA.m

```

function MotiveBiRotorADSOST
% ------------------------------------------------------------- %
                    % ABRIR PUERTO A ESP8266 %

% Create TCP/IP object 't'. Specify server machine and port number.
t = tcpip('192.168.0.150', 80);

% Open connection to the server.
fopen(t);

```

```
12                      % ABRIR PUERTO A ESP8266 %
13 % ------------------------------------------------------------- %
14
15 fprintf( 'Control de Bi-Rotor: Programa iniciado\n' )
16
17 % create an instance of the natnet client class
18 fprintf( 'Creando objeto en clase NatNet\n' )
19 natnetclient = natnet;
20
21 % connect the client to the server (multicast over local loopback) -
22 % modify for your network
23 fprintf( 'Conectado al servidor (laptop)\n' )
24 natnetclient.HostIP = '127.0.0.1';
25 natnetclient.ClientIP = '127.0.0.1';
26 natnetclient.ConnectionType = 'Multicast';
27 natnetclient.connect;
28
29
30 % get the asset descriptions for the asset names
31 model = natnetclient.getModelDescription;
32 if ( model.RigidBodyCount < 1 )
33 return
34 end
35
36 % Codigo para obtener el vector de estados del Bi-Rotor mediante las
37 % camaras
38 %fprintf( '\nMostrando vector de estados del Bi-Rotor\n\n' )
39
40 % ------------------------------------------------------------- %
41          % Validacion de variables para obtener q punto %
42 frame_anterior=0;
43 frame_actual=0;
44
45 t_anterior=0;
46 t_actual=0;
47
48 elev_anterior=0;
49 elev_actual=0;
50
51 roll_anterior=0;
52 roll_actual=0;
```

```
53
54 pitch_anterior=0;
55 pitch_actual=0;
56
57 yaw_anterior=0;
58 yaw_actual=0;
59
60 xip = [0; 0; 0];                % Elemento xipunto para controlador SMC
61 xi = [0; 0; 0];                 % Elemento xi para controlador SMC
62
63 Kp = [0; 0; 0];                 % Factor adaptivo variable Kpunto
64 K = [0; 0; 0];                  % Integral
65          % Validacion de variables para obtener q punto %
66 % ------------------------------------------------------------------ %
67 vector_t=[];
68 vector_elev=[];
69 vector_roll=[];
70 vector_yaw=[];
71 vector_tau1=[];
72 vector_tau2=[];
73 vector_tau3=[];
74
75 disp ('Iniciando Motores...')
76 for indini = 1100:2:1500
77 indexstr1 = int2str(indini+15);
78 indexstr2 = int2str(indini);
79 inicio = strcat ('a',indexstr1,'-b',indexstr2);
80 fprintf(t,inicio);
81 pause(.01);
82 end
83
84 for idx = 1 : 15000
85
86 % ------------------------------------------------------------------ %
87      % Asignacion de diferencias entre q actual y anterior %
88 frame_anterior = frame_actual;
89 t_anterior=t_actual;
90 elev_anterior=elev_actual;
91 roll_anterior=roll_actual;
92 pitch_anterior=pitch_actual;
93 yaw_anterior=yaw_actual;
```

```
94       % Asignacion de diferencias entre q actual y anterior %
95  % ------------------------------------------------------------- %
96
97
98  data = natnetclient.getFrame; % OBTENCION DE CUADRO ACTUAL
99
100 frame_actual = data.Frame;
101 t_actual=data.Timestamp;
102
103 if (idx == 1)
104 t_inicial=t_actual;
105 end
106
107
108 if (frame_anterior ~= frame_actual)
109
110     % ------------------------------------------------------------- %
111             % Obtener componentes de quaternion y elevacion %
112
113 elev_actual = data.RigidBody(1).z;
114
115 q0 = data.RigidBody(1).qw;
116 q1 = data.RigidBody(1).qx;
117 q2 = data.RigidBody(1).qy;
118 q3 = data.RigidBody(1).qz;
119
120             % Obtener componentes de quaternion y elevacion %
121     % ------------------------------------------------------------- %
122
123     % ------------------------------------------------------------- %
124             % Transformacion de quaternion a angulos en Rad %
125
126 roll_actual = asin (2 * ((q0*q2) - (q3*q1)));
127 pitch_actual = atan2( (2 * ( (q0*q1)+ (q2*q3) ) ),
128 1 - (2 * (q1^2 + q2^2)) );
129 yaw_actual = atan2( (2 * ( (q0*q3)+ (q1*q2) ) ),
130 1 - (2 * (q2^2 + q3^2)));
131
132             % Transformacion de quaternion a angulos en Rad %
133     % ------------------------------------------------------------- %
134
```

```
135 dif_tiempo = (t_actual - t_anterior);
136 dif_elev = (elev_actual - elev_anterior);
137 dif_roll = (roll_actual - roll_anterior);
138 dif_yaw = (yaw_actual - yaw_anterior);
139
140 vel_z = dif_elev / dif_tiempo;
141 vel_roll = dif_roll / dif_tiempo;
142 vel_yaw = dif_yaw / dif_tiempo;
143
144 vector_q=[elev_actual; roll_actual; yaw_actual];
145 vector_qp =[vel_z; vel_roll; vel_yaw];
146 c_roll = cos (roll_actual);
147 s_roll = sin (roll_actual);
148
149 % ------------------------------------------------------------- %
150                     % C O N T R O L A D O R   A D S O S M C %
151 % ------------------------------------------------------------- %
152
153     q= vector_q;
154     qd=[0.70; 0 * (pi / 180); 0];
155     %qd=[0.36338; 1.264 * (pi / 180) ; 0 * (pi / 180)];
156     qp= vector_qp;
157     qe=qd-q;
158     qdp=[0;0;0];
159     % Vector de velocidades en SetPoints
160     qdpp=[0;0;0];
161     % Vector de aceleraciones en SetPoints
162
163     lambda=[2; 2.75; 1];
164     % Ganancia de q-qd (Rapidez de Convergencia a S)
165
166     s=qp-qdp+lambda.*(q-qd);
167     % Fase deslizante
168     abs_s=abs(s);
169
170     % ELEMENTOS DE CONTROLADOR ADAPTABLE
171     % ---------------------------------------
172
173
174     K_min=[.001; .00001; 0];    % Asegurar valores distintos a 0
175
```

```
176     eps = [.002; 0.000001; 0];  % Factor necesario para obtener Lambda
177     mhu = [.15; 0.12; 0];     % Umbral de perdida modo deslizante
178
179     k = [.03; .00035; 0];    % Tasa de adaptacion
180
181     if (K(1)>K_min(1))
182         Kp(1)= k(1)*sign((abs_s(1))-mhu(1));
183     end
184     if (K(1)<=K_min(1))
185         Kp(1)= K_min(1);
186     end
187
188     if (K(2)>K_min(2))
189         Kp(2)= k(2)*sign((abs_s(2))-mhu(2));
190     end
191     if (K(2)<=K_min(2))
192         Kp(2)= K_min(2);
193     end
194
195     K=(K+Kp); %*(0.01/2);    %Sumatoria en aproximacion trapezoidal
196     %(acumulador) de K = Integral tiempo discreto
197
198     Lambda = 2 * eps .* K;
199
200     xip= -Lambda .* sign(s);
201     xi = xi + xip;
202
203
204     ua= -K .*((abs_s).^(1/2)) .* sign(s) + xi;
205     %Control Auxilizar -> Modos deslizantes <-
206     if (ua(1)>18)
207         ua(1)=18;
208     end
209     if (ua(1)<-18)
210         ua(1)=-18;
211     end
212
213     % ----------------------------------------
214     % ELEMENTOS DE CONTROLADOR ADAPTABLE
215
216
```

```
217
218     kf1=0.01;         %Constante de friccion de elevacion.
219     kf2=0.03;         %Constante de friccion del alabeo.
220     kf3=0.0065;         %Constante de friccion de guinada.
221
222     m=1.84;            %Masa del sistema.
223     Ix=0.043;          %Inercia de roll (Obtenidas experimentalmente)
224     Iz=0.041;          %Inercia del yaw (Obtenidas experimentalmente)
225
226     c_roll;    %Coseno de phi
227     s_roll;    %Seno de phi
228
229     M=[m 0 0; 0 Ix 0; 0 0 Iz*c_roll^2];
230     %Matriz de masas e inercias
231     Cqp=[0;Iz*c_roll*s_roll*qp(3)^2;-2*Iz*c_roll*s_roll*qp(2)*qp(3)];
232     %Vector de coriolis
233     f=[kf1 0 0; 0 kf2 0; 0 0 kf3]*qp;
234     %Vector de fricciones
235     g=[9.81*m; 0; 0];
236     %Vector de gravedad
237
238
239     tau= M *(qdpp-lambda.*(qp-qdp)) + Cqp + g + f + ua;  %Salida tau
240
241     % SATURACION DE GANANCIAS DE CONTROL %
242
243     if (tau(1)>36)
244          tau(1)=36;
245     end
246
247     if (tau(2)>6.12)
248          tau(2)=6.12;
249     end
250
251     if (tau(3)>1.73)
252          tau(3)=1.73;
253     end
254     % ------------------------------------------------------------- %
255                          % C O N T R O L A D O R  ADSOSMC%
256     % ------------------------------------------------------------- %
257
```

```
258     % ------------------------------------------------------------- %
259                   % AJUSTE PARA ENVIO DE DATOS A MOTORES %
260
261      elevPWM= (tau(1)/0.015)/2;
262      rollPWM= (tau(2)/0.00612);
263
264      pwm1=int16(1010 + elevPWM + (rollPWM/2));
265      pwm2=int16(1000 + elevPWM - (rollPWM/2));
266
267      if (pwm1>=2000)
268          pwm1 = 2000;
269      end
270      if (pwm1<=1000)
271          pwm1 = 1000;
272      end
273
274      if (pwm2>=2000)
275          pwm2=2000;
276      end
277      if (pwm2<=1000)
278          pwm2=1000;
279      end
280
281                   % AJUSTE PARA ENVIO DE DATOS A MOTORES %
282     % ------------------------------------------------------------- %
283
284
285
286     % ------------------------------------------------------------- %
287                         % ENVIO DE DATOS A MOTORES %
288     chr1 = int2str(pwm1);
289     chr2 = int2str(pwm2);
290     cadena = strcat ('a',chr1,'-b',chr2);
291     fprintf(t,cadena);
292     %fprintf(cadena);
293     %fprintf('\n');
294                          % ENVIO DE DATOS A MOTORES %
295     % ------------------------------------------------------------- %
296
297     vector_t(end+1)= data.Timestamp;
298     vector_roll(end+1)= vector_q(2) * (180/pi);
```

```
299      vector_elev(end+1)= vector_q(1);
300      vector_yaw(end+1)= vector_q(3) * (180/pi);
301      vector_tau1(end+1)= tau(1);
302      vector_tau2(end+1)= tau(2);
303      vector_tau3(end+1)= tau(3);
304
305      fprintf( 'Frame:%6d  ' , data.Frame )
306      fprintf( 'Time:%0.3f\n' , data.Timestamp - t_inicial)
307      fprintf( 'Vector q= %.5f \n', vector_q );
308      fprintf( 'Vector qp= %.5f \n', vector_qp );
309      fprintf( 'Kp: %.7f \n', Kp);
310      fprintf( '|sigma|: %.5f \n', abs_s);
311      fprintf( 'K: %.5f \n', K);
312      fprintf( 'ua: %.5f \n', ua);
313      fprintf( 'Control tau: %.5f \n', tau);
314      fprintf( 'PWM1: %i \n', pwm1);
315      fprintf( 'PWM2: %i \n', pwm2);
316      fprintf( '\n')
317  end
318  end
319
320  disp ('Descenso')
321
322
323
324  for ind = 1500:-1:1100
325  indexstr = int2str(ind);
326  cadena = strcat ('a',indexstr,'-b',indexstr);
327  fprintf(t,cadena);
328  pause(.025);
329  end
330
331  fprintf(t,'a1000-b1000');
332  delete(t);
333  clear t
334  clear pwm1
335  clear pwm2
336
337  fecha = datetime('now','Format','dd-MM-yy-HH_mm_ss');
338  fecha = char (fecha);
339  nombre=['Prueba_ADSOST_' fecha '.mat'];
```

```
340
341 titulo1 = strcat ('Controlador ADSOST Vector Q _',fecha);
342 titulo2 = strcat ('Controlador ADSOST Vector TAU _',fecha);
343
344 set(gcf,'Name',titulo1);
345 subplot(3,1,1);
346 plot (vector_t - t_inicial, vector_elev, 'LineWidth', 1)
347 ylabel('z (m)','Interpreter','latex','FontSize',16);
348 %title('Evolucion temporal variables de estado',
349 'Interpreter','latex','FontSize',20);
350 ylim([0 1.2])
351 grid on;
352
353 subplot(3,1,2);
354 plot (vector_t - t_inicial, vector_roll, 'LineWidth', 1)
355 ylabel('\phi (grados)','Interpreter','latex','FontSize',16);
356 ylim([-50 50])
357 grid on;
358
359 subplot(3,1,3);
360 plot (vector_t - t_inicial, vector_yaw, 'LineWidth', 1)
361 xlabel('Tiempo (seg)','Interpreter','latex','FontSize',16);
362 ylabel('\psi (grados)','Interpreter','latex','FontSize',16);
363 ylim([-180 180])
364 grid on;
365
366
367 figure();
368 set(gcf,'Name',titulo2);
369 subplot(3,1,1);
370 plot(vector_t - t_inicial, vector_tau1, 'LineWidth', 1)
371 ylabel('u (N)','Interpreter','latex','FontSize',16);
372 ylim([0 36])
373 grid on;
374
375 subplot(3,1,2);
376 plot(vector_t - t_inicial, vector_tau2, 'LineWidth', 1)
377 ylabel('\tau_\phi (N·m)','Interpreter','latex','FontSize',16);
378 ylim([-2 2])
379 grid on;
380
```

```
subplot(3,1,3);
plot(vector_t - t_inicial, vector_tau3, 'LineWidth', 1)
xlabel('Tiempo $(seg)$','Interpreter','latex','FontSize',16);
ylabel('$\tau_\psi$ (N·m)','Interpreter','latex','FontSize',16);
ylim([-2 2])
grid on;


save(nombre,'lambda','K_min','eps', 'mhu','k', 'vector_t','vector_elev',
'vector_roll','vector_yaw','vector_tau1','vector_tau2','vector_tau3')
disp('Programa terminado' )
end
```