UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA
INGENIERÍA ELÉCTRICA – SISTEMAS ELECTRÓNICOS

DISEÑO DE UN SISTEMA DE CONTROL DE ORIENTACIÓN UTILIZANDO RUEDAS DE REACCIÓN

TESIS
QUE PARA OPTAR POR EL GRADO DE:
MAESTRO EN INGENIERÍA

PRESENTA:
ANTONIO GARCÍA SANTIAGO

TUTOR PRINCIPAL
DR. JORGE PRADO MOLINA
INSTITUTO DE GEOGRAFÍA

CIUDAD UNIVERSITARIA, CD. MX., DICIEMBRE 2017

JURADO ASIGNADO:

Presidente:	Dr. Pérez Alcázar Pablo Roberto
Secretario:	Dra. Navarrete Montesinos Margarita
1 er. Vocal:	Dr. Prado Molina Jorge
2 do. Vocal:	Dr. De La Rosa Nieves Saúl
3 er. Vocal:	Dra. Moumtadi Fátima

Lugar o lugares donde se realizó la tesis: INSTITUTO DE GEOGRAFÍA, UNAM.

TUTOR DE TESIS:
DR. JORGE PRADO MOLINA

AGRADECIMIENTOS

A la Universidad Nacional Autónoma de México por permitirme realizar mis estudios de posgrado

y por ser mi alma mater.

Al Laboratorio de Análisis Geoespacial del Instituto de Geografía por permitirme desarrollar el presente proyecto.

Al Conacyt por la beca otorgada.

A mi tutor el Dr. Jorge Prado Molina, por sus consejos y

por su apoyo en la realización de este proyecto.

A mis padres: Asunción y Paula por su invaluable apoyo.

A mis hermanos: Carlos, Alejandro, Reyna y Mari.

A mis compañeros de laboratorio y de la maestría.

ÍNDICE

1. INTRODUCCIÓN 1
1.1. Subsistemas que componen a un satélite 1
1.2. Subsistema de control 2
1.3. Sensores 3
1.3.1. Magnetómetro 3
1.3.2. Giróscopo 4
1.4. Actuadores 4
1.4.1 Ruedas de reacción o inerciales 4
1.5. Computadora de a bordo 4
1.6. Algoritmo de control 5
1.7. Tipos de control de orientación 5
1.8. Factores que determinan la selección del tipo de sistema de control de orientación 7
1.9. Estado del arte del control de orientación en nanosatélites 7
1.10. Infraestructura 10
1.11. Justificación 10
1.12. Objetivo 10
2. SISTEMAS DE REFERENCIA Y REPRESENTACIÓN DE LA ORIENTACIÓN DE UN SATÉLITE 12
2.1. Sistema de referencia inercial con centro en la Tierra 12
2.2. Sistema de referencia fijo con centro en la Tierra 12
2.3. Sistema de referencia orbital 13
2.4. Sistema de referencia fijo al satélite 13
2.5. Matriz de rotación 14
2.6. Representación de la orientación 15
2.6.1 Ángulos de Euler 16
2.6.2 Cuaterniones 16
3. DINÁMICA DEL SATÉLITE 18
3.1. Modelo del satélite 18
3.2. Cinemática del satélite 20
3.3. Pares externos que afectan a los satélites 24
3.3.1. Par de gradiente gravitacional 24
3.3.2. Par por arrastre aerodinámico 25

3.3.3. Par magnético 25
3.4. Par producido por las ruedas de reacción 26
4. LEY DE CONTROL 27
4.1. Ley de control: posición más velocidad angular 27
5. REQUERIMIENTOS DE HARDWARE DEL SISTEMA DE CONTROL 29
5.1. Prototipo de satélite 29
5.1.1. Sensores 29
5.1.2. Actuadores 29
5.1.3. Computadora de a bordo 30
5.1.4. Fuente de alimentación 31
5.1.5. Motores 31
5.2. Plataforma de simulación 32
5.3. Diseño de las ruedas inerciales 34
5.3.1. Determinación de las dimensiones de las ruedas de reacción 36
5.4. Caracterización dinámica de la plataforma de simulación 38
5.4.1. Pruebas sobre el eje Z 40
5.4.2. Pruebas sobre el eje X 44
6. ALGORITMO DE CONTROL 48
6.1. Algoritmo de control proporcional en Matlab 48
6.2. Implementación del algoritmo en la computadora de a bordo 49
6.3. Algoritmo de control óptimo LQR (Regulador Lineal Cuadrático) 52
7. RESULTADOS Y CONCLUSIONES 55
7.1. Resultados 55
7.2. Conclusiones 57
BIBLIOGRAFÍA 59
ANEXOS 62
A 62
B 69
C 70
D 72
E 79
F 80

ÍNDICE DE TABLAS

Tabla 1-1 Clasificación de los satélites de acuerdo a su masa. 1
Tabla 1-2 Precisión alcanzada utilizando diferentes tipos de actuadores. 10
Tabla 5-1 Valores de caracterización de la plataforma de simulación sobre el eje Z. 41
Tabla 5-2 Constantes de la señal de control para el eje Y y Z. 43
Tabla 5-3 Valores de caracterización de la plataforma de simulación sobre el eje X. 45
Tabla 5-4 Constantes de la señal de control para el eje X. 47

ÍNDICE DE FIGURAS

Figura 1-1 Subsistemas que componen a un satélite. 2
Figura 1-2 Tipos de control de orientación [8]. 7
Figura 2-1 Sistema de referencia inercial. 12
Figura 2-2 Sistema de referencia fijo con centro en la Tierra. 13
Figura 2-3 Sistema de referencia orbital. 13
Figura 2-4 Sistema de referencia fijo en el cuerpo. 14
Figura 2-5 Representación de la orientación con cuaterniones. 17
Figura 3-1 Sistema de referencia orbital y fijo al cuerpo del satélite. 20
Figura 3-2 Sistema de referencia orbital y vector de velocidad angular. 23
Figura 5-1 Sensores instalados en el prototipo de satélite. Se muestran los sistemas de referencia asociados a los sensores. 29
Figura 5-2 Ruedas de reacción colocadas ortogonalmente en el prototipo de satélite. 30
Figura 5-3 Computadora de a bordo (Raspberry Pi) montada en el prototipo de satélite. 31
Figura 5-4 Diagrama de la fuente que alimenta a la computadora de a bordo del prototipo de satélite. 31
Figura 5-5 Fuente para alimentar a los puentes H. 32
Figura 5-6 Copa que a través de seis orificios provee el aire a presión para sustentar la esfera en donde se coloca la estructura completa del nanosatélite. 33
Figura 5-7 Detalle del simulador satelital, la esfera está montada sobre la copa. Es en esta unión dónde se genera un medio sin fricción. 33
Figura 5-8 Prototipo de un nanosatélite CubeSat 3U, montado sobre un simulador satelital. En esta estructura se encuentra el sistema de control de orientación. 34
Figura 5-9 Momento de inercia de una esfera hueca. 35
Figura 5-10 Ejes de momentos de inercia para un prisma rectangular sólido. 35
Figura 5-11 Geometría de las ruedas de reacción, dónde r1 es el radio interno, r2 el radio externo y h la altura. 37
Figura 5-12 Forma del impulso angular aplicado a la plataforma de simulación para su caracterización. 39
Figura 5-13 Impulso de 0.3 normalizado aplicado sobre el eje Z (guiñada) 40
Figura 5-14 Impulso de 0.5 normalizado aplicado sobre el eje Z (guiñada) 40
Figura 5-15 Impulso de 0.8 normalizado aplicado sobre el eje Z (guiñada) 41
Figura 5-16 Gráfica para obtener las constantes k2 y c2 para los ejes Y y Z. 42
Figura 5-17 Gráfica para obtener las constantes k1 y c1 para los ejes Y y Z. 43

Figura 5-18 Impulso de 0.3 normalizado aplicado sobre el eje X (cabeceo). 44
Figura 5-19 Impulso de 0.5 normalizado aplicado sobre el eje X (cabeceo). 44
Figura 5-20 Impulso de 0.9 normalizado aplicado sobre el eje X (cabeceo). 45
Figura 5-21 Gráfica para obtener las constantes k2 y c2 para el eje X. 46
Figura 5-22 Gráfica para obtener las constantes k1 y c1 para el eje X. 46
Figura 6-1 Algoritmo de control de orientación en Simulink de Matlab. 48
Figura 6-2 Respuesta del algoritmo de control proporcional. Eje x (azul). Eje y (verde). Eje z (rojo). 49
Figura 6-3 Diagrama de bloques del sistema de control de orientación del satélite. 50
Figura 6-4 Diagrama de flujo del algoritmo de control de orientación. 51
Figura 6-5 Respuesta del algoritmo de control LQR. Eje x (rojo). Eje y (verde). Eje z (azul). 54
Figura 7-1 Control de orientación sobre el eje X (cabeceo). 55
Figura 7-2 Control de orientación sobre el eje Y (alabeo). 56
Figura 7-3 Control de orientación sobre el eje Z (guiñada). 56

Resumen

Se presenta el diseño, la construcción y las pruebas de funcionamiento en el Laboratorio, de un subsistema de control de orientación para un prototipo de nanosatélite de baja órbita integrado por un algoritmo, sensores, computadora de a bordo y actuadores. El subsistema se fijó en una estructura de un nanosatélite 3U para llevar a cabo su validación. El control de orientación se realizó en tres ejes, haciendo uso de ruedas de reacción como actuadores, por lo que se calcularon las dimensiones necesarias, para tener una operación eficaz. En lo referente a los sensores, se tiene un magnetómetro en tres ejes y una unidad de medición inercial; la cual consta de un giróscopo de tres ejes para medir las velocidades angulares. La computadora de a bordo, tiene la característica de que ya ha sido utilizada en vuelo orbital en nanosatélites. El algoritmo de control proporcional desarrollado, permite alcanzar una precisión de ±2°, adicionalmente se da una revisión de un algoritmo de control LQR para realizar una comparación entre estos algoritmos.

El subsistema de control de orientación se prueba en el Laboratorio, en un simulador satelital con movimiento irrestricto en tres ejes y que reproduce un ambiente con mínima fricción. La estructura completa del prototipo de nanosatélite 3U, con el sistema de control de orientación desarrollado, fue montada en este simulador satelital para verificar su funcionamiento.

1. INTRODUCCIÓN

Los nanosatélites se lanzaron desde los primeros días de los vuelos espaciales (1957-1962) y los picosatélites a partir de 1997, hasta la actualidad. La razón de que los primeros satélites fueran de tamaño reducido se debió a la pequeña capacidad de carga útil de los vehículos de lanzamiento. Pero una vez que su capacidad aumentó, los satélites se hicieron cada vez más grandes, por lo que en las siguientes décadas no hubo necesidad de poner en órbita satélites pequeños como en los primeros años, además de que los componentes tecnológicos desarrollados eran para satélites grandes. Sin embargo, a finales de los años noventa esto cambió debido a la disponibilidad de electrónica de baja potencia, que también proporcionaba un alto rendimiento, una reducción significativa en el costo, así como el tiempo de desarrollo [1]. Otro avance sobresaliente fue la concepción del estándar CubeSat, que fue introducido en 1999 por la Universidad Estatal Politécnica de California y la Universidad de Stanford, lo que impulsó enormemente el número de pico y nanosatélites desarrollados, especialmente entre las universidades [1, 2].

En la tabla 1-1 se da una clasificación de los satélites de acuerdo a su masa, hecha por la Universidad de Surrey del Reino Unido.

Grande	> 400 kg
Pequeño	100 – 400 kg
Micro	10 - 100 kg
Nano	1 – 10 kg
Pico	0.1 – 1 kg

Tabla 1-1 Clasificación de los satélites de acuerdo a su masa.

1.1. Subsistemas que componen a un satélite

Un nanosatélite se compone de varios subsistemas [3] que son (ver figura 1-1):

Computadora de a bordo.
Comunicación; para establecer enlaces con la estación terrena.
Subsistema de potencia para alimentar a todos los subsistemas que lo requieran.
Carga útil (experimento para el que fue creada la misión espacial).
Control térmico.
Estructura exterior (mantiene a la carga útil y a los componentes del satélite protegidos)
Sistema de determinación y control de orientación.

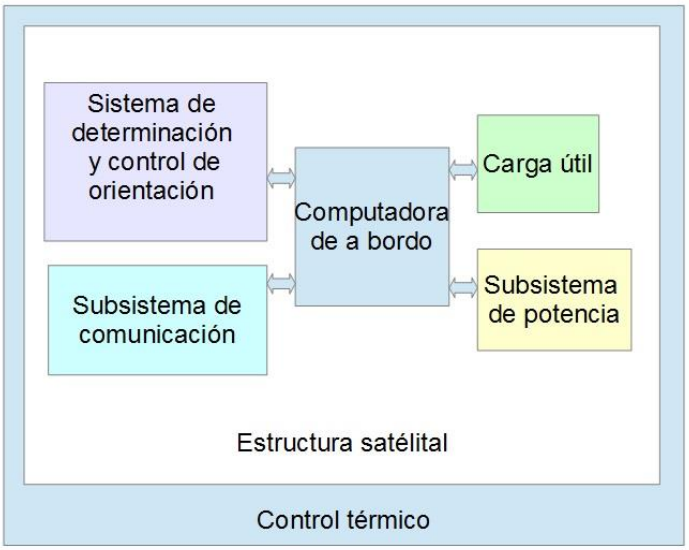

Figura 1-1 Subsistemas que componen a un satélite.

1.2. Subsistema de control

Cuando un satélite se desprende del cohete y se coloca en órbita, este comienza a rotar de manera inestable mientras se traslada en su órbita, debido al impulso que le imprime el sistema de expulsión del propio cohete y en etapas posteriores, debido a los pares externos que hay en el medio ambiente espacial. Estos pares externos se deben principalmente al arrastre atmosférico, el gradiente gravitacional, el par magnético y la radiación solar. En una órbita baja como en la que se colocan los nanosatélites, los pares externos que tienen mayor afectación son los primeros tres. El par producido por el viento solar afecta mayormente a los satélites geoestacionarios (36,000 km de altura). De manera tal que, si un satélite no cuenta con un mecanismo para cambiar o estabilizar su orientación, ésta tenderá a modificarse de manera importante.

El subsistema de determinación y control de orientación se compone de un conjunto de sensores, actuadores, un modelo dinámico y una computadora de a bordo; la cual ejecuta el algoritmo de control. El objetivo de este sistema es amortiguar la rotación inestable y el momento angular inicial al completarse la puesta en órbita del satélite. También cumple la tarea de determinar la orientación actual y mantenerla o cambiarla, basándose en los datos que van recopilando sus sensores [4].

Determinar la orientación de un cuerpo rígido (en este caso de un satélite) consiste en medir su rotación respecto de un marco de referencia fijo. Dicha rotación es medida mediante el uso de sensores que entregan información sobre el campo magnético, la dirección del sol, la dirección de las estrellas, etc. De manera que una combinación de la información proporcionada por los sensores, referida a un sistema de referencia fijo, nos proporciona la orientación del satélite [4].

Para mantener o cambiar la orientación del satélite es necesario un mecanismo que genere un

par correctivo, el cual comúnmente es proporcionado por conjuntos de bobinas magnéticas y ruedas inerciales.

1.3. Sensores

Hay dos clases de sensores de orientación. Los que obtienen mediciones absolutas y los que obtienen mediciones relativas. Las mediciones absolutas de los sensores se basan en el hecho de que, al conocerse la posición en órbita del satélite, se realizan cálculos para determinar los vectores de dirección, con respecto a un sistema de referencia, el cual está determinado por los objetos astronómicos o las líneas del campo magnético de la Tierra. Los sensores de medición absoluta miden estas direcciones con respecto al sistema de referencia fijo a la nave y mediante una matriz de rotación determinan la orientación del satélite, con respecto al sistema de referencia inercial. Las medidas absolutas son usadas en los algoritmos de determinación de orientación estáticos. [5]

Los sensores de medición relativa pertenecen a la clase de los instrumentos giroscópicos, tales como el girocompás y el giróscopo de integración. Los sensores de mediciones relativas son utilizados en algoritmos de determinación de orientación dinámica. [5]

1.3.1. Magnetómetro

El magnetómetro es un sensor de referencia inercial referido al sistema de coordenadas fijo a la Tierra con centro en la Tierra. Los magnetómetros miden la dirección del campo magnético Terrestre, que normalmente se mide con un vector de tres componentes XYZ, en el sistema coordenado en el que están configurados los sensores. Estas componentes son normalizadas para describir un vector unitario que está relacionado con la dirección del campo magnético en dicho sistema coordenado, por lo que se puede obtener de manera inmediata la orientación del satélite en tres ejes. La lectura del magnetómetro obtenida tiene la forma:

$$b_B = [b_{Bx} + b_{By} + b_{Bz}]^T \tag{1.1}$$

$$|b_B| = 1 \tag{1.2}$$

Para interpretar la lectura del magnetómetro, es necesario conocer el modelo del campo magnético de la Tierra, tal que la información es un vector referido al sistema de referencia fijo a la Tierra con centro en la Tierra. Mediante el uso de una matriz de rotación, la información del modelo de campo magnético terrestre es convertida al sistema de referencia fijo al cuerpo del satélite, es decir:

$$b_B = R_e^b(q_b^e)b_E \tag{1.3}$$

Donde q_b^e es el cuaternión unitario que relaciona el sistema de referencia fijo a la Tierra con el sistema de referencia fijo al cuerpo del satélite. [6]

1.3.2. Giróscopo

El giróscopo es un dispositivo que obtiene la velocidad angular absoluta del satélite respecto al sistema de referencia inercial, de tal manera que la medición tiene la siguiente forma:

$$\Omega_b^i = [\omega_x\ \omega_y\ \omega_z]^T + v(t) \tag{1.4}$$

Donde $v(t)$ es una función que describe el ruido y la deriva [6] y $\omega_x, \omega_{y,}\ y\ \omega_z$ las velocidades angulares en los ejes x, y y z.

1.4. Actuadores

Para mantener o cambiar la orientación del satélite se necesita un mecanismo para generar un par correctivo, que comúnmente son bobinas magnéticas, propulsores, ruedas de reacción o inerciales o una combinación de estos actuadores.

1.4.1 Ruedas de reacción o inerciales

Cuando una rueda opera a velocidad cero y entonces reacciona para cambiar la orientación del satélite, se le conoce como rueda de reacción. Si la rueda mantiene una velocidad constante y aumenta o disminuye esa velocidad, entonces se le llama rueda inercial.

El principio bajo el cual operan las ruedas inerciales o las ruedas de reacción es el intercambio de momento angular. Un satélite equipado con ruedas inerciales o de reacción, puede cambiar su momento angular mediante un cambio en la velocidad de cada una de sus ruedas. Este cambio de velocidad genera un par, el cual causa cambios en la orientación del satélite, es decir, si las ruedas aceleran en una dirección el satélite rotará en la dirección contraria [6,7].

1.5. Computadora de a bordo

De los microcontroladores que se utilizan como computadora de a bordo en los satélites, se ha encontrado que estos son de 8, 16, y 32 bits. Como es bien sabido, el valor en bits denota la capacidad para efectuar cálculos complejos de valor flotante. Anteriormente los microcontroladores de 32 bits consumían mucha energía y no eran ideales para utilizarse en un micro o nano satélite, pero en la actualidad empresas como Atmel han sacado al mercado este tipo de microcontroladores con un bajo consumo de energía [4]. De manera que hoy en día se prefiere utilizar microcontroladores de 32 bits o más para la determinación de la orientación del satélite, la ejecución del algoritmo de control y los cálculos matemáticos. Recientemente los nuevos satélites CubeSat han incorporado microcomputadoras con alta capacidad de procesamiento y manejo de un sistema operativo, como es el caso de la Raspberry Pi.

Al elegir la computadora de a bordo que va a realizar el control de orientación, una de las cosas que se debe tomar en cuenta es que la capacidad de memoria sea suficiente para almacenar el algoritmo de control.

Otros requerimientos para seleccionar la computadora de a bordo [3] son:

-Suficiente número de convertidores analógico-digitales.

-Suficiente número de entradas y salidas.

-Puertos de comunicación serial (SPI, I2C, USART, USB).

-Que se programe en lenguajes de alto nivel (C, Java, Python).

-Disponibilidad de herramientas de desarrollo.

-Tolerancia a la temperatura -40 ºC a 80 ºC.

A continuación, se mencionan algunos microcontroladores que se han utilizado en misiones satelitales [3].

-MSP430 en el satélite Jugnu-IIT Kanpur`s.

-ATmega128 en el satélite PRATHAM.

-Intel 80C186 en el satélite coreano KITSAT.

-Atmel AT91M40800 para el satélite DTUsat.

-Hitachi SH7045 para el satélite TUBSAT.

-Raspberry Pi para el proyecto Astro Pi, y para el proyecto AAReST

1.6. Algoritmo de control

Los lenguajes de programación que se pueden implementar en un microcontrolador o microcomputadora están: C, Python y Java. En este trabajo para la elaboración del algoritmo de control se eligió el lenguaje Python ya que cuenta con librerías que manejan cálculos matriciales complejos, similares a los que se realizan con Matlab.

1.7. Tipos de control de orientación

Dependiendo de la aplicación u objetivo para el que fue diseñado el satélite se elige el tipo de control de orientación.

En lo que respecta al sistema de control de orientación éste puede clasificarse en cuatro tipos (los primeros satélites como el Sputnik, y el Echo 1 y 2 fueron enviados sin un sistema de control de orientación). Los tipos 1, 2 y 3 involucran el giro de la nave o de alguna parte de ella. La función de giro hace que la orientación del satélite sea inherentemente estable [8]; si la nave es afectada por un par perturbador, el cambio en la orientación resultante es pequeño. Esta es una característica útil, que implica que el control de orientación no realiza un gran esfuerzo para controlar la orientación del satélite, no obstante, se emplea energía para mantener la estabilidad giroscópica.

Los satélites con control del tipo 1 son llamados estabilizados por giro, en donde la nave es de forma cilíndrica y rota a determinada velocidad proporcionando estabilidad. El control de tipo 2 es conocido como de doble giro, en la cual se tiene una sección de la plataforma satelital que rota a determinada velocidad, pero adicionalmente se cuenta con un mecanismo montado en la parte superior que no gira, de tal manera que los sistemas del satélite que deban apuntar en una dirección fija deben estar montados en esta sección. El tercer tipo de control es el llamado híbrido, en el cual la nave cuenta con una rueda, que proporciona estabilidad giroscópica en un eje y adicionalmente cuenta con otros actuadores para corregir la orientación. El cuarto tipo es conocido como control tri-axial; en este caso no necesariamente se utilizan ruedas inerciales, por lo cual no se cuenta con la estabilidad inherente y el subsistema de control de orientación debe realizar un gran trabajo para lograr el apuntamiento requerido. Un ejemplo de lo anterior son los satélites de telecomunicaciones localizados en órbita geosíncrona (órbita cuyo movimiento alrededor del planeta está sincronizado con un punto sobre la superficie) que cambia su orientación con pequeños cohetes localizados en los extremos de los ejes de rotación de la nave.

Figura 1-2 Tipos de control de orientación [8].

1.8. Factores que determinan la selección del tipo de sistema de control de orientación

Los parámetros que establecen los requerimientos y el intervalo de variabilidad para los componentes del sistema de control de orientación son: los objetivos de la misión, el presupuesto disponible, la instrumentación utilizada como carga útil, la selección de la órbita, los requerimientos de estabilización y de apuntamiento. Dichos requerimientos del satélite determinan el sistema de control de orientación a utilizar, que puede ser un simple control por giro o un control de orientación robusto en tres ejes. La estabilización por giro es generalmente menos compleja y menos cara que una estabilización en tres ejes y más adecuada para algunos experimentos científicos. La estabilización en tres ejes, sin embargo, se requiere para misiones que se enfoquen a comunicaciones, exploración, observación de la Tierra y mediciones astronómicas.

1.9. Estado del arte del control de orientación en nanosatélites

Casi 40% de los pico y nanosatélites lanzados tienen un sistema de control de orientación activo, mientras que el mismo porcentaje ha sido lanzado con un control pasivo, principalmente usando material magnético. Un poco más del 20% no tiene algún tipo de control. La función principal de muchos de los sistemas de control en pico y nanosatélites es simplemente limitar la rotación del satélite. Alrededor del 15% de los pico y nanosatélites usan el control de orientación para apuntar un instrumento. Principalmente se utiliza en

apuntamiento a nadir para una cámara o para la detección de radiación junto a las líneas de campo magnético [1].

Los conjuntos de sensores más comúnmente usados son los de Sol y los magnetómetros, seguidos de sensores de Sol y giróscopos, los menos usados son los sensores rastreadores de estrellas; no obstante, estos últimos son los más precisos a la fecha [1].

Las ruedas de reacción y los propulsores son los actuadores más adecuados para un control dinámico preciso, pero aún no se han implementado de manera extensiva en los pico y nanosatélites [1].

El primer nanosatélite que fue estabilizado en tres ejes fue el SNAP-1 (Surrey Nanosatellite Applications Platform) el cual fue desarrollado y construido por Surrey Satellite Technology Ltd. (SSTL). Esta nave fue lanzada el 28 de junio del año 2000 a bordo del cohete Russian Cosmos. La principal misión de este satélite de 6.5 kg fue mostrar el control de orientación en tres ejes, así como las maniobras orbitales. El SNAP-1 utilizó un magnetómetro de tres ejes, una cámara y un filtro de Kalman para estimar la orientación, además, para el control de estabilización utilizó tres bobinas magnéticas en conjunto con una rueda inercial [9].

El nano satélite STRaND-1 fue enviado al espacio con capacidad de manejar varios modos de control a lo largo de su operación. El primero fue usado para disminuir la inestabilidad que se presenta inmediatamente después de haber sido liberado desde el vehículo de lanzamiento. Este control se consigue con la combinación de las leyes de control –B punto y Y-Thomson. Un segundo control se empleó para orientar los paneles solares al sol. Y un tercero, utiliza una rueda de momentos para tener estabilidad en un eje y un cuarto realiza la implementación de un control en tres ejes con ruedas inerciales.

El control de orientación para el satélite AAUSAT-II utiliza como sensores un magnetómetro de tres ejes, seis giróscopos de un eje, seis fotodiodos y seis sensores de temperatura; y como actuadores, tres ruedas de momento y tres bobinas magnéticas [10].

El sistema de determinación y control de orientación para el satélite AAUSAT3 fue provisto para tener estabilización en tres ejes utilizando bobinas magnéticas como actuadores. La orientación y la velocidad angular se miden por magnetómetros y giróscopos, respetivamente. Su sistema de determinación y control de orientación se separa en dos subsistemas. El primero se encarga de controlar los giros inestables una vez que el satélite es puesto en órbita, de tal manera, que disminuye la velocidad de rotación del satélite utilizando solamente las medidas del campo magnético. El control utilizado para este caso fue el –B punto. El hardware para llevar a cabo esto fue un microcontrolador AT90CAN128, un magnetómetro Honeywell HMC6343 para medir el campo magnético y seis bobinas magnéticas como actuadores. El segundo subsistema da una precisión buena en la orientación del satélite. El hardware utilizado fue un microcontrolador AT91SAM7A1, seis sensores de sol colocados en cada una de las caras del satélite, un giróscopo de tres ejes y un magnetómetro de tres ejes [11].

En [12] se presenta una arquitectura para el control de orientación de un satélite utilizando cuatro ruedas inerciales como actuadores. El objetivo del control es seguir al Sol y a la estación terrena. El algoritmo se implementa y prueba en Matlab/Simulink.

En [13] se presenta, el diseño y evaluación del sistema de control de orientación del satélite CubeSat PHOENIX. Este satélite es de tipo 2U con un peso de 2 kg. Este satélite requirió de un sistema de control que le permitiera tener un eje paralelo al vector velocidad, de manera que los instrumentos de medición pudiesen medir correctamente la termosfera inferior.

Después de que el satélite es desplegado, éste entra en una etapa de rotación inestable de modo que para contrarrestar este efecto se debe de aplicar una ley de control diferente a la que se utiliza para un apuntamiento preciso; en este satélite (PHOENIX) se emplea la ley de control -B punto utilizando bobinas magnéticas. El modelo de perturbaciones tomó en cuenta el modelo dinámico, el modelo de par de gradiente gravitacional, y el modelo de pares magnéticos. La verificación del sistema de control fue hecha mediante simulación. Los sensores tales como el magnetómetro, sensores de sol, los giróscopos, así como los sensores de velocidad de las ruedas inerciales fueron agregados al modelo de simulación.

El modelo de control de orientación para este satélite estuvo compuesto de 4 partes: control -B punto, control de rotación, desaturación y control sobre el eje y. Las bobinas magnéticas fueron utilizadas para la realización de los tres primeros controles, en tanto que para la realización del control sobre el eje y, se emplearon las ruedas inerciales.

Los resultados muestran que el estado de frenado de la rotación inestable se llevó a cabo en 11600 segundos, teniendo velocidades angulares iniciales de 10 grados/segundo. El control con ruedas inerciales fue llevado a cabo después de este tiempo.

La estrategia de control de orientación para el satélite ANTELSAT CubeSat 2U se describe en [14] y consiste de dos fases. En la primera se propone una ley de control B-punto para reducir la velocidad del satélite cuando se encuentra en estado de rotación inestable (tumbling). La segunda fase, después de que la energía cinética ha disminuido lo suficiente, propone un control en tres ejes.

Los sensores y actuadores utilizados proveen una exactitud de apuntamiento de 10 grados en cada eje. Los sensores utilizados para el control son: un magnetómetro de tres ejes, tres giróscopos de un eje y seis sensores de Sol. Todos los sensores son conectados al microcontrolador encargado de la determinación y control de orientación. El control es llevado a cabo por tres bobinas magnéticas, colocadas sobre los tres ejes principales de inercia. El control de energía en las bobinas se realiza mediante modulación de ancho de pulso PWM (por sus siglas en inglés Pulse Width Modulation).

Para la simulación se ha empleado Matlab/Simulink. El simulador incluye modelos del ambiente espacial, las perturbaciones, la cinemática y dinámica del satélite, modelos de los sensores y actuadores, así como el algoritmo de determinación y control de orientación.

En [15] se presenta el control de orientación utilizando una rueda de reacción para dos motores ultrasónicos lineales mediante una estructura doble. El concepto central de ese diseño es permitir el ajuste del eje principal de la rueda de reacción en todas direcciones. Este

sistema puede utilizarse también para disminuir la inestabilidad y para la desaturación de las ruedas de reacción.

En la tabla 1-2 se muestra la precisión alcanzada en la orientación de un nanosatélite utilizando diversos métodos de control de orientación [15].

Actuador	Precisión (grados)	Método
Estabilización por spin	0.1-1	Pasivo
Gradiente gravitacional	1-5	Pasivo
Control con propulsores	0.01	Consumible
Bobinas magnéticas	1-2	Lento solamente órbitas bajas
Ruedas de reacción	0.001-1	Alta precisión

Tabla 1-2 Precisión alcanzada utilizando diferentes tipos de actuadores.

Por lo que se puede concluir que las ruedas de reacción son la mejor opción para un control preciso de la orientación de un satélite.

Por lo menos tres ruedas de reacción deben de ser instaladas en un CubeSat para proveer estabilización en tres ejes.

1.10. Infraestructura

En el Laboratorio de Percepción Remota Alternativa y Tecnología Avanzada del Instituto de Geografía de la UNAM, se han llevado a cabo diferentes proyectos encaminados principalmente al desarrollo de los subsistemas para la determinación de la orientación y el control de la estabilización de nano y microsatélites. Para ello se ha desarrollado una plataforma para probar satélites CubeSat 3U que emula la falta de fricción que hay en el espacio, de manera que se puedan probar algoritmos de control en la Tierra. Este es un paso indispensable para lograr el control satelital en órbita.

1.11. Justificación

En nuestro país actualmente se están llevando a cabo algunos esfuerzos de desarrollo de nano y microsatélites, en los que se está considerando la utilización de sistemas de control de orientación, por lo que es deseable contar con esta tecnología y no depender de su adquisición en el extranjero. Este trabajo de tesis ayudará a cumplir este propósito.

1.12. Objetivo

Diseñar un algoritmo de control de orientación satelital en tres ejes que utilice ruedas de reacción como actuadores, determinar los requerimientos necesarios para su implementación y llevar a cabo pruebas de funcionamiento en un simulador físico.

Se tienen las siguientes metas:

- Seleccionar los sensores y la computadora de a bordo a utilizar.
- Seleccionar los motores a utilizar, así como los puentes h necesarios para su control.
- Diseñar las ruedas de reacción para el simulador físico.
- Desarrollar la plataforma de simulación y el prototipo de satélite.
- Elaborar el algoritmo de control.
- Probar el algoritmo de control en un simulador físico en el Laboratorio.
- Efectuar los ajustes y pruebas finales del programa de control de orientación en Tierra.

2. SISTEMAS DE REFERENCIA Y REPRESENTACIÓN DE LA ORIENTACIÓN DE UN SATÉLITE

La idea fundamental en el estudio de la orientación de un cuerpo rígido es el sistema de referencia [5].

2.1. Sistema de referencia inercial con centro en la Tierra

El sistema de referencia inercial R_i se asume fijo en el espacio, lo cual implica que se trata de un marco no acelerado en el cual las leyes de Newton son válidas [16]. Su origen se establece en el centro de la Tierra con el eje Z_i apuntando al polo norte geográfico, el eje X_i apunta al equinoccio vernal, también conocido como el primer punto de Aries, y el eje Y_i completa el sistema de referencia con base a la regla de la mano derecha (figura 2-1).

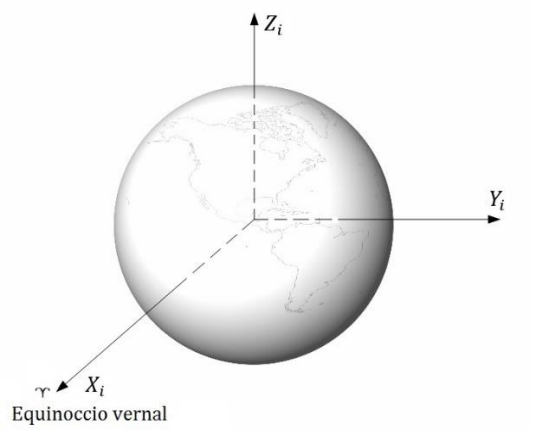

Figura 2-1 Sistema de referencia inercial.

2.2. Sistema de referencia fijo con centro en la Tierra

El origen de este sistema de referencia está localizado en el centro de la Tierra. Los ejes X_e y Y_e rotan alrededor del eje Z_e , con respecto al sistema de referencia inercial con centro en la Tierra, a una velocidad angular $\omega = 7.2921x10^{-5}[rad/s]$ que es la velocidad a la que rota nuestro planeta. El eje Z_e apunta hacia el polo norte, el eje X_e hacia la intersección del meridiano de Greenwich con el Ecuador y el eje Y_e es perpendicular a X_e y Z_e . La importancia de este sistema de referencia es que un punto en la superficie de la Tierra tal como la estación terrena o los objetivos de observación, son fijos en este sistema. Sin embargo, como este está rotando es un sistema de referencia no inercial [5,17] (figura 2-2).

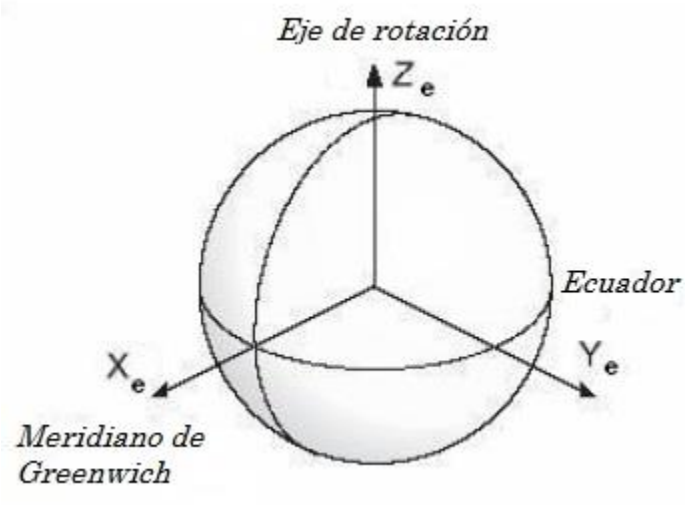

Figura 2-2 Sistema de referencia fijo con centro en la Tierra.

2.3. Sistema de referencia orbital

El sistema de referencia orbital R_o coincide con el centro de masa del satélite. La parte positiva del eje X_o apunta en el mismo sentido que la velocidad lineal del satélite al encontrarse en órbita. La dirección de este eje cambia conforme varía la dirección del vector velocidad. El eje Z_o apunta hacia el centro (nadir) de la tierra y el eje Y_o se obtiene complementando al sistema ortogonal utilizando la regla de la mano derecha (figura 2-3).

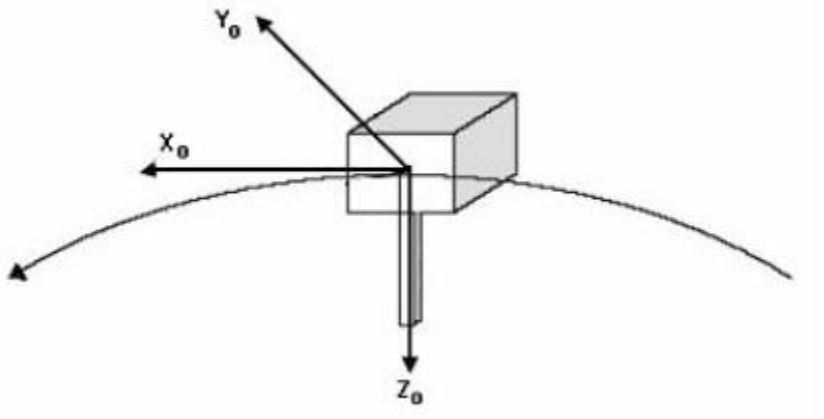

Figura 2-3 Sistema de referencia orbital.

2.4. Sistema de referencia fijo al satélite

El sistema de referencia fijo al cuerpo del satélite R_b, tiene a sus ejes X_b, Y_b y Z_b coincidentes con los ejes principales de inercia del satélite, su origen se encuentra en el centro de masa del satélite. Se trata de un sistema de referencia no inercial ya que se encuentra fijo al cuerpo del satélite. La orientación de la nave se determina, con relación al sistema de referencia orbital R_o, "mientras que la velocidad angular se expresa en el sistema fijo al

satélite.” El objetivo del sistema de control de orientación es que el sistema de referencia fijo al satélite R_b y el sistema de referencia orbital coincidan. La rotación que presentan los ejes de este sistema con respecto al sistema de referencia orbital R_o se conocen como los ángulos de cabeceo (roll), alabeo (pitch) y guiñada (yaw) que se representan por θ, φ y ψ, respectivamente. De igual manera, a los ejes de este sistema se les conoce como X_b de cabeceo, Y_b de alabeo, Z_b de guiñada (figura 2-4).

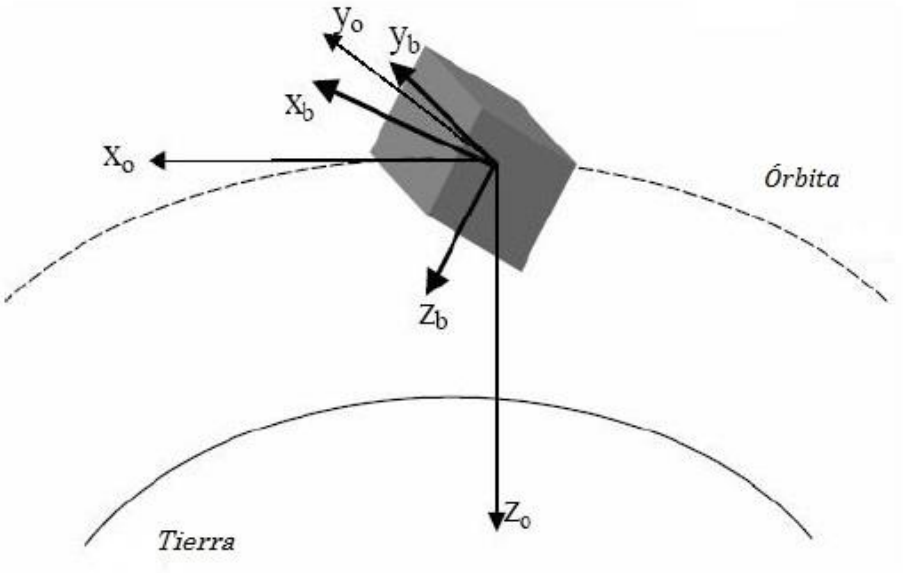

Figura 2-4 Sistema de referencia fijo en el cuerpo.

2.5. Matriz de rotación

La matriz de rotación se interpreta de tres formas diferentes: como una matriz que transforma un vector de un sistema de referencia a otro, como una rotación de un vector dentro del mismo sistema de referencia y también como una descripción de la orientación mutua entre dos sistemas de referencia. La matriz de rotación R de un sistema de referencia a hacia un sistema de referencia b es denotada por R_a^b, la cual es un miembro del grupo ortogonal especial de orden 3 y se define como: [18,19]

$$SO(3) = \{R|R \in R^{3x3}, R^T R = I, \det R = 1\} \tag{2.1}$$

donde R^{3x3} es el conjunto de todas las matrices de $3x3$ con elementos reales e I es la matriz identidad de $3x3$. En general, para denotar la rotación de un vector v^a de un sistema de referencia a a un vector v^b en un sistema de referencia b, se utiliza la siguiente expresión:

$$v^b = R_a^b \, v^a \tag{2.2}$$

En el caso de que se tengan tres o más sistemas coordenados y se desee una rotación de un primer sistema a a un tercer sistema c se requieren dos rotaciones. Primeramente, una rotación del sistema a al b, seguido de una rotación del sistema b al sistema c, es decir se ha llevado a cabo una rotación compuesta dada por: [19]

$$R_a^c = R_a^b R_b^c \tag{2.3}$$

Una parametrización útil de la matriz de rotación es aquella representada por un eje-ángulo, $R_{\lambda,\beta}$, que corresponde a una rotación β sobre el eje λ. [18]

$$R_{\lambda,\beta} = I + S(\lambda)\sin\beta + (1-\cos\beta)S^2(\lambda) \tag{2.4}$$

Donde S es el operador anti-simétrico, que está definido por la siguiente ecuación:

$$S(\lambda) = -S(\lambda) = \begin{bmatrix} 0 & -\lambda_3 & \lambda_2 \\ \lambda_3 & 0 & -\lambda_1 \\ -\lambda_1 & \lambda_1 & 0 \end{bmatrix} \tag{2.5}$$

Las rotaciones simples utilizando ángulos de Euler como parámetros, son definidas como:

$$R_{x,\psi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & \sin\psi \\ 0 & -\sin\psi & \cos\psi \end{bmatrix} \tag{2.6}$$

$$R_{y,\theta} = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \tag{2.7}$$

$$R_{z,\phi} = \begin{bmatrix} \cos\phi & \sin\phi & 0 \\ -sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.8}$$

Donde los subíndices x, y, y z denotan a los ejes y ψ, θ, y ϕ a los ángulos de rotación alrededor de cada uno de los ejes, respectivamente.

2.6. Representación de la orientación

La orientación de un cuerpo en el espacio tridimensional se representa utilizando diversas herramientas matemáticas, empleando tres o cuatro parámetros para este propósito. En la literatura, se encuentran diferentes métodos para representar la orientación de un satélite, tales como: eje-ángulo de Euler, rotación de vectores, cuaterniones, parámetros de Rodrigues, parámetros de Rodrigues modificados y los ángulos de Euler [16].

2.6.1 Ángulos de Euler

Se necesitan tres ángulos para especificar la orientación de un cuerpo rígido en relación a un marco inercial. La elección no es única, hay dos conjuntos de uso común: los ángulos de Euler y los ángulos guiñada, alabeo y cabeceo [20].

Los ángulos de Euler nos dan la orientación de un cuerpo rígido de un sistema de referencia relativo xyz ortogonal a un sistema de referencia inercial XYZ ortogonal.

2.6.2 Cuaterniones

Los cuaterniones son un conjunto numérico introducido por Sir William Rowan Hamilton a mediados del siglo XIX. Este conjunto, con las operaciones de suma y multiplicación definidas entre ellos, forman un sistema matemático conocido como anillo de división no conmutativo. Este título enfatiza el hecho de que el producto de cuaterniones, en general, no es conmutativo y también que para cada elemento diferente de cero existe un inverso multiplicativo [21].

Los cuaterniones han probado ser útiles en la representación de rotaciones en espacios tridimensionales, tarea en la que resultan ser más eficientes que las matrices de rotación, utilizando solamente cuatro elementos en vez de los nueve de la matriz y obedeciendo únicamente a una restricción: el que la norma del cuaternión sea unitaria [16]. Por esta razón han sido utilizados en aplicaciones relacionadas con el control de orientación de vehículos tales como aeronaves, submarinos, naves espaciales, animación tridimensional y realidad virtual [21].

El hecho de que a los cuaterniones se les use ampliamente para representar la orientación de un cuerpo rígido se basa en el teorema rotacional de Euler, el cual establece que “el desplazamiento general de un cuerpo rígido con un punto fijo es una rotación alrededor de cierto eje” [20]. De tal manera que la rotación de un cuerpo en el espacio tridimensional puede cuantificarse describiendo en un sistema de referencia inercial el eje ε alrededor del cual se lleva a cabo la rotación y un ángulo de rotación θ, tal como se muestra en la figura 2.5. Cabe mencionar que el eje ε es conocido como el eje de Euler y θ como el ángulo de Euler [19].

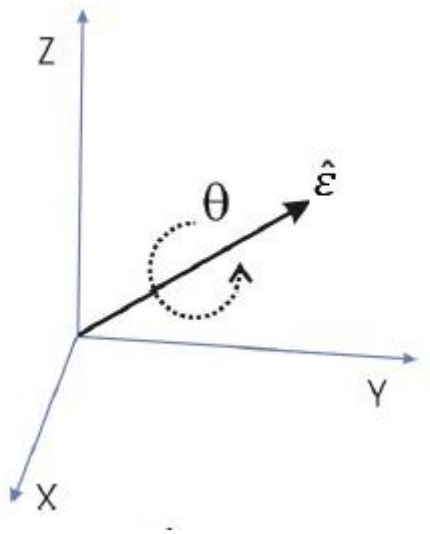

Figura 2-5 Representación de la orientación con cuaterniones.

Un cuaternion $\boldsymbol{q}$ se compone de cuatro elementos que se dividen en dos partes: una escalar η y otra vectorial $\boldsymbol{\varepsilon}$ [22].

$$\boldsymbol{q} = \begin{bmatrix} \boldsymbol{\varepsilon} \\ \eta \end{bmatrix} = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \eta \end{bmatrix} \tag{2.9}$$

De acuerdo al teorema rotacional de Euler, un cuaternión queda definido por un eje de rotación y un ángulo de rotación. La representación de una transformación de un sistema inercial a uno no inercial utilizando cuaterniones es definida por [23]:

$$\boldsymbol{q} = \begin{bmatrix} \boldsymbol{\varepsilon} \\ \eta \end{bmatrix} = \begin{bmatrix} \|\boldsymbol{\varepsilon}\| \sin(\frac{\theta}{2}) \\ \cos\left(\frac{\theta}{2}\right) \end{bmatrix} \tag{2.10}$$

en donde θ es el ángulo que debe de ser rotado sobre el eje ε (medidos en el sistema de referencia inercial) para obtener la orientación en el sistema de referencia no inercial.

La rotación a lo largo de tres ejes puede ser implementada multiplicando cuaterniones [16].

3. DINÁMICA DEL SATÉLITE

El movimiento general de un cuerpo rígido es una combinación de traslación y rotación. Para fijar la posición de un cuerpo rígido se necesitan seis datos (es decir, tiene seis grados de libertad), tres para fijar la posición de un punto cualquiera A y tres ángulos que definen la orientación del cuerpo con respecto al punto A. En la elección del punto A es natural tomar aquel que se corresponde con el centro de masa y describir las traslaciones del centro de masa y las rotaciones respecto de él [24].

Una forma de descomponer el estudio de la dinámica del satélite es dividirlo en cinemática y cinética. Para el movimiento traslacional, la cinemática es el estudio de cambio de posición dada una velocidad, mientras que la cinética estudia como las fuerzas causan cambios en la velocidad. Para el movimiento rotacional, la cinemática es el estudio de los cambios de orientación para una velocidad angular dada, y la cinética es el estudio de como los torques causan cambios en la velocidad angular [5].

Para analizar el movimiento de traslación del cuerpo rígido, se requiere solamente concentrarse en el centro de masa y aplicar los métodos de la mecánica de la partícula para determinar su movimiento [20,24].

Analizar la dinámica rotacional requiere del conocimiento de los momentos angulares del cuerpo, así como la manera en que la masa está distribuida a través del cuerpo. La distribución de masa es descrita por las seis componentes del tensor de inercia I [20].

$$I = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix} \tag{3.1}$$

Cualquiera que sea la forma del cuerpo y su distribución de densidad siempre hay tres ejes llamados ejes principales de inercia, ortogonales entre sí y que pasan por el centro de masa, tales que, si el cuerpo gira alrededor de uno de ellos, el momento angular L es paralelo a la velocidad angular ω [24].

Los valores de I_{xx}, I_{yy}, e I_{zz} correspondientes a los momentos de inercia para rotaciones alrededor de los ejes principales de inercia, se llaman momentos principales de inercia. Estos valores son constantes, independientes del tiempo. Sin embargo, la orientación de los ejes principales depende del tiempo y su variación describe las rotaciones del cuerpo [24].

3.1. Modelo del satélite

El movimiento de los sistemas mecánicos rotacionales se estudia con la ley análoga a la segunda ley de Newton. Ésta relaciona la cantidad de movimiento angular con el momento total que se ejerce sobre el cuerpo.

$$\tau = \frac{dL}{dt} \tag{3.2}$$

La ecuación 3.2 nos dice que el par absoluto que actúa sobre el sistema es igual a la derivada de la cantidad de movimiento angular absoluto calculado alrededor del centro de masa.

Al definir un sistema de referencia inercial R_i y un sistema fijo al cuerpo del satélite R_b se obtiene el modelo matemático del satélite, donde la ecuación 3.2 se reescribe como,

$$\tau_{ib}^{b} = \frac{d}{dt}\left(L_{ib}^{b}\right) \tag{3.3}$$

Donde:

τ_{ib}^{b} es el vector de par total que actúa sobre el satélite, medido en el sistema de referencia inercial pero expresado en el sistema de referencia fijo al cuerpo del satélite.

L_{ib}^{b} es el vector de cantidad de movimiento del satélite, medido en el sistema de referencia inercial pero expresado en el sistema de referencia fijo al cuerpo del satélite.

La expresión que nos permite calcular la cantidad de movimiento angular es:

$$L_{ib}^{b} = I\omega_{ib}^{b} \tag{3.4}$$

Donde:

I es la matriz de momentos de inercia del satélite, obtenida con respecto al sistema R_b.

ω_{ib}^{b} es la velocidad angular absoluta del satélite, es decir, con respecto al sistema inercial. Pero expresado en el sistema R_b.

El vector τ_{ib}^{b} calculado en el sistema de referencia no inercial R_b, se obtiene como la derivada de un vector fijo a un sistema no inercial que rota a cierta velocidad angular con respecto al sistema inercial [20, 25].

$$\tau_{ib}^{b} = \frac{d}{dt}\left(L_{ib}^{b}\right) + \omega_{ib}^{b}\, x\, L_{ib}^{b} \tag{3.5}$$

$$\tau_{ib}^{b} = I\dot{\omega_{ib}^{b}} + \omega_{ib}^{b} x\, I\omega_{ib}^{b} \tag{3.6}$$

Asumiendo que la matriz de inercia es diagonal $I = diag[I_{xx}\ I_{yy}\ I_{zz}]$, es decir, se trabaja con los momentos de inercia principales del cuerpo rígido, $\omega_{ib}^{b} = [\omega_{ibx}^{b}\ \omega_{iby}^{b}\ \omega_{ibz}^{b}]^T$ son las velocidades angulares del cuerpo rígido respecto a un marco de referencia inercial y $\tau_{ib}^{b} = [\tau_{ibx}^{b}\ \tau_{iby}^{b}\ \tau_{ibz}^{b}]^T$ son los torques que actúan sobre el satélite medidos en el sistema de referencia inercial.

De manera que las ecuaciones de movimiento en forma de componentes quedan como [24,25]:

$$\tau_{ibx}^{b} = I_{xx}\dot{\omega_{ibx}^{b}} + (I_{zz} - I_{yy})\omega_{iby}^{b}\omega_{ibz}^{b} \quad (3.7)$$

$$\tau_{iby}^{b} = I_{yy}\dot{\omega_{iby}^{b}} + (I_{xx} - I_{zz})\omega_{ibx}^{b}\omega_{ibz}^{b} \quad (3.8)$$

$$\tau_{ibz}^{b} = I_{zz}\dot{\omega_{ibz}^{b}} + (I_{yy} - I_{xx})\omega_{ibx}^{b}\omega_{iby}^{b} \quad (3.9)$$

que son conocidas como las ecuaciones de Euler.

3.2. Cinemática del satélite

La dinámica rotacional del satélite no es suficiente para conocer la orientación del mismo ya que las ecuaciones (3.7) (3.8) y (3.9) tienen como referencia al sistema inercial, y lo que se requiere es conocer la orientación respecto al sistema de referencia orbital R_o, para realizar las maniobras de control.

En la figura 3.1 se muestran los sistemas de referencia R_o y R_b. Como ya se mencionó en el capítulo 2, el eje Z_o del sistema R_o apunta siempre al centro de la Tierra y el eje X_o es tangente a la órbita. Además, este sistema de referencia rota alrededor de su eje Y_o a la misma razón a la que el satélite orbita la Tierra. Cabe mencionar que el vector de velocidad angular en el sistema R_b es negativo si es expresado en el mismo sistema R_b, ya que se considera una órbita directa o prograda, es decir, el satélite se mueve en el mismo sentido, [26], que el sentido de rotación de la Tierra.

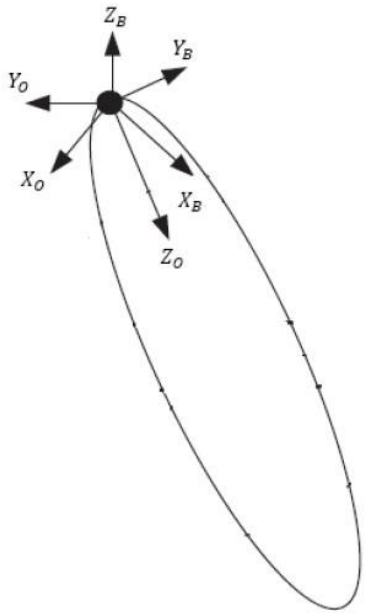

Figura 3-1 Sistema de referencia orbital y fijo al cuerpo del satélite.

En una misión donde el objetivo es tomar imágenes de la tierra (percepción remota) es de interés que los ejes principales de inercia del satélite coincidan con los ejes del sistema R_b, ya que el eje Z_o al apuntar al centro de la Tierra es normal al plano que forma la superficie Terrestre sobre la que el satélite se encuentra. Esto permite que se lleve a cabo una medición útil para la parte científica de la misión espacial [26]. Debido a esto, es de interés conocer la

orientación del sistema R_b con respecto al sistema R_o, puesto que éste último fungirá como el sistema de referencia.

La orientación de un sistema de referencia con respecto a otro, se describe a través de diversos métodos, por ejemplo: ángulos de Euler, matrices de cosenos directores, cuaterniones, entre otros. Utilizando cuaterniones la orientación del sistema R_b con respecto al sistema R_o, queda descrito por:

$$q_0^b = \begin{bmatrix} q_{01}^b \\ q_{02}^b \\ q_{03}^b \\ q_{04}^b \end{bmatrix} = \begin{bmatrix} n_{ox} sin\frac{\theta}{2} \\ n_{oy} sin\frac{\theta}{2} \\ n_{oz} sin\frac{\theta}{2} \\ cos\frac{\theta}{2} \end{bmatrix} \tag{3.10}$$

Y la cinemática está dada por la siguiente ecuación considerando que el sistema R_b se encuentra fijo al cuerpo del satélite (i.e. la velocidad angular de R_b es la misma que la del satélite) [27,2]:

$$\dot{q_o^b} = \frac{1}{2}\,\Omega(\omega_o^b) q_o^b \tag{3.11}$$

Donde:

$$\Omega(\omega_o^b) = \begin{bmatrix} 0 & \omega_{oz}^b & -\omega_{oy}^b & \omega_{ox}^b \\ -\omega_{oz}^b & 0 & \omega_{ox}^b & \omega_{oy}^b \\ \omega_{oy}^b & -\omega_{ox}^b & 0 & \omega_{oz}^b \\ -\omega_{ox}^b & -\omega_{oy}^b & -\omega_{oz}^b & 0 \end{bmatrix} \tag{3.12}$$

ω_{ox}^b, ω_{ox}^b, ω_{ox}^b, son las componentes de la velocidad angular del sistema R_b, relativo al sistema de referencia orbital R_o.

Para obtener ω_o^b se hace uso del concepto de movimiento relativo. Si se tienen dos cuerpos A y B que rotan con respecto a un sistema de referencia inercial y que también rotan entre ellos, entonces la velocidad absoluta de B puede ser descrita como la velocidad de B con respecto a A, más la velocidad absoluta de A, es decir [28]:

$$v_B = v_{B/A} + v_A \tag{3.13}$$

Trasladando lo anterior al problema que nos interesa, se tiene que:

$$\omega_{ib}^b = \omega_{ib}^o + \omega_{ob}^b \tag{3.14}$$

Donde:

ω_{ob}^b es la velocidad angular del satélite respecto al sistema de referencia orbital, expresado en el sistema de referencia fijo al cuerpo del satélite.

ω_{ib}^{o} es la velocidad angular absoluta del sistema de referencia orbital, expresado en el sistema de referencia fijo al cuerpo del satélite.

La ecuación anterior sostiene que la velocidad angular absoluta del satélite con respecto al sistema de referencia inercial con centro en la Tierra, expresado en el sistema de referencia fijo al cuerpo del satélite ω_{ib}^{b}, se escribe como la suma de la velocidad angular del satélite con respecto al sistema orbital ω_{ob}^{b}, más la velocidad angular absoluta del sistema de referencia orbital ω_{io}^{b}, ambas expresadas en el sistema de referencia fijo al cuerpo del satélite [23].

Despejando ω_{ob}^{b}

$$\omega_{ob}^{b} = \omega_{ib}^{b} - \omega_{ib}^{o} \tag{3.15}$$

El valor de ω_{ib}^{b} se obtiene de resolver las ecuaciones (3.7), (3.8) y (3.9). Para calcular ω_{ob}^{b} ya solo queda determinar el valor de ω_{ib}^{o}, para este propósito se debe de recordar que el sistema rota alrededor de su eje Y, a la misma razón que el satélite orbita la tierra, es decir, a la velocidad angular del satélite con respecto a la Tierra. De esta manera ω_{ib}^{o}, que es la velocidad absoluta del sistema de referencia orbital con respecto al sistema de referencia inercial, expresado en el sistema de referencia fijo al cuerpo del satélite, es igual a la velocidad del satélite en cada instante de su órbita. Debido a que se considera una órbita circular, la velocidad angular del satélite es constante y se calcula como:

$$n = \sqrt{\frac{\mu}{R^3}} \tag{3.16}$$

Donde μ es la constante de gravitación terrestre. En este trabajo se considera que $\mu = 398\,600\ km^3/s^2$. R es la distancia del centro de la Tierra al centro del satélite, es decir, la suma del radio de la tierra con la altitud del satélite. El radio de la tierra se toma como $R_E = 6378\ km$.

Con la ecuación (3.16) se obtiene la magnitud del vector ω_{ib}^{o}, por tanto, solo hace falta determinar su dirección y sentido. Para esto se determinará primero la dirección y sentido de ω_{io}^{o}. Se ha mencionado que el sistema orbital rota alrededor de su eje Y, por lo que su velocidad es normal al plano orbital, de tal manera que sus componentes en los ejes X y Z son nulos. Si se asume una órbita directa, entonces el sentido del giro del sistema orbital se realiza en el sentido horario, por lo que la componente en el eje Y del vector ω_{io}^{o} es negativo, como se muestra en la figura 3-2.

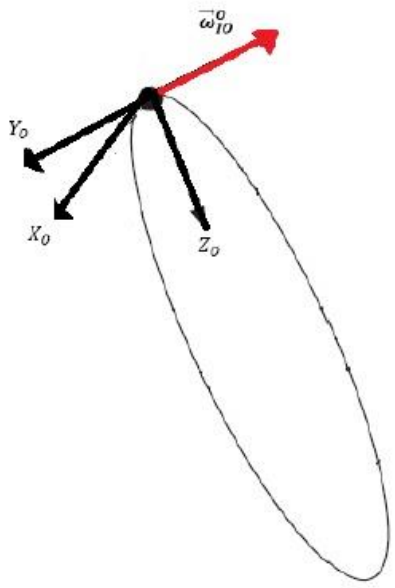

Figura 3-2 Sistema de referencia orbital y vector de velocidad angular.

Entonces,

$$\boldsymbol{\omega_{io}^{o}} = \begin{bmatrix} 0 \\ -n \\ 0 \end{bmatrix} \tag{3.17}$$

Finalmente, para obtener ω_{ib}^{o} se hace uso de la matriz de transformación $[C]_{ob}$, de tal manera que,

$$\boldsymbol{\omega_{ib}^{o}} = [C]_{ob}\boldsymbol{\omega_{io}^{o}} \tag{3.18}$$

donde

$$[C]_{ob} = \begin{bmatrix} q_1^2 - q_2^2 - q_3^2 + q_4^2 & 2(q_1\, q_2 + q_3\, q_4) & 2(q_1\, q_2 - q_3\, q_4) \\ 2(q_1\, q_2 - q_3\, q_4) & -q_1^2 + q_2^2 - q_3^2 + q_4^2 & 2(q_2\, q_3 + q_1\, q_4) \\ 2(q_1\, q_3 + q_2\, q_4) & 2(q_2\, q_3 - q_1\, q_4) & -q_1^2 - q_2^2 + q_3^2 + q_4^2 \end{bmatrix} \tag{3.19}$$

Por lo que,

$$\omega_{ob}^{b} = \omega_{ib}^{b} - [C]_{ob}\boldsymbol{\omega_{io}^{o}} \tag{3.20}$$

Simplificando

$$\omega_{ob}^{b} = \omega_{ib}^{b} + n[C_{1:3,2}]_{ob} \tag{3.21}$$

Donde $[C_{1:3,2}]_{ob}$ es el vector columna formado por la segunda columna de la matriz $[C]_{ob}$.

Para el caso de las pruebas en Tierra que se realizan en esta tesis, se da el caso que la plataforma de simulación no se traslada, de manera que el sistema se simplifica. En ese sistema no hay traslación solamente rotación alrededor del centro de masa. Las medidas que entregan los sensores lo hacen respecto al sistema de referencia inercial que se establece para las pruebas, este sistema de referencia coincide con el sistema de referencia anclado al satélite R_b y puesto que no hay traslación, también coincide con el sistema de referencia orbital R_o, de manera que cualquier transformación entre estos sistemas de referencia es unitaria.

3.3. Pares externos que afectan a los satélites

Los satélites en órbita se encuentran sujetos a distintas fuerzas con las cuales interactúan y que producen perturbaciones en la orientación y en la órbita de la nave. Shrivastaba y Modi [29] describen este tipo de fuerzas. A continuación, se presenta un resumen de los modelos de las perturbaciones.

3.3.1. Par de gradiente gravitacional

$$\tau_g = \frac{3[(I_{zz}-I_{yy})mn\hat{\imath} + (I_{xx}-I_{zz})ln\hat{\jmath} + (I_{yy}-I_{xx})lm\hat{k}]\dot{\theta}^2}{1+ecos\theta} \tag{3.22}$$

En esta ecuación se desprecian los elementos de orden superior y se asume que el satélite es un cuerpo rígido.

Donde:

τ_g es el par de gradiente gravitacional.

θ es la anomalía verdadera.

e es la excentricidad de la órbita.

I_{xx}, I_{yy}, I_{zz}, son los momentos principales de inercia.

l, m, n, son los cosenos directores de la línea satélite-Tierra con respecto a los ejes principales del cuerpo.

El par de gradiente gravitacional desaparece si:

-Dos cosenos directores son cero.

-$I_{xx} = I_{yy} = I_{zz}$

-Dos momentos de inercia principales son iguales y un coseno director es cero.

El par de gradiente gravitacional produce una perturbación considerable para un satélite que es estabilizado en tres ejes, a menos que su configuración sea tal que su máximo momento

de inercia de un eje sea perpendicular a la órbita y que el eje con mínimo momento de inercia sea paralelo a la vertical local.

El gradiente gravitacional puede ser utilizado como una forma de estabilización pasiva, así como una técnica de desaturación de momento. Para la estabilización, la desventaja es que es altamente susceptible a otras perturbaciones ambientales.

3.3.2. Par por arrastre aerodinámico

$$\tau_A = \varepsilon \; x \; F_A \tag{3.23}$$

$$F_A = 0.5 \; \rho_a \; V^2 \; A \; C_D \tag{3.24}$$

Donde:

ε es la posición del vector del centro de presión con respecto al centro de masa.

ρ_a es la densidad atmosférica.

V velocidad relativa del satélite respecto a la atmósfera.

C_D es el coeficiente de arrastre.

Las fuerzas aerodinámicas son considerables hasta una altura de 800 km y son estas perturbaciones ambientales las dominantes para órbitas bajas.

Las dificultades encontradas en el modelado de la atmósfera son debidas a las grandes variaciones en la densidad y peso, la posición relativa del Sol y su actividad. También la rotación atmosférica debe de ser considerada si se requiere mucha precisión en el modelo.

3.3.3. Par magnético

$$\tau_M = M \; x \; B \tag{3.25}$$

Donde:

M es el momento dipolar del satélite.

B es al campo geomagnético local.

La interacción con el campo magnético de la Tierra es una fuente considerable de perturbaciones para órbitas que se encuentran debajo de los 1500 km de altura.

Un modelo simplificado del campo magnético de la Tierra es como un pequeño dipolo magnético con una fuerza de $8.5 \; x \; 10^5 \; Wbm$, localizado en el centro de la misma y cuyo eje está inclinado 11.5 grados con respecto al eje polar.

La magnetización y desmagnetización de materiales ferromagnéticos a bordo del satélite conduce a un amortiguamiento por histéresis.

Para reducir las influencias indeseables del torque magnético se minimizan:

-El uso de materiales ferromagnéticos.

-El momento dipolar del satélite a través de un diseño correcto de los circuitos eléctricos.

3.4. Par producido por las ruedas de reacción

El modelo matemático para las ruedas de reacción colocadas en los ejes ortogonales x,y,z está dado por la siguiente ecuación [30]:

$$\tau_r^b = \left(\frac{dL_r}{dt}\right)^b + \omega_{ib}^b \, x \, L_r - \tau_{fricción}^b \tag{3.26}$$

Donde τ_r^b es el par causado por las ruedas de reacción, $L_r = [L_{rx}\, L_{ry}\, L_{rz}]^T = I_r\, \omega_r$ es el vector de momento angular de las propias ruedas y $\tau_{fricción}^b$ es el par que es causado por la fricción que generalmente se asume como cero. Expresando la ecuación en términos de sus componentes queda como:

$$\tau_r^b = \begin{bmatrix} \tau_{rx} \\ \tau_{ry} \\ \tau_{rz} \end{bmatrix} = \begin{bmatrix} \dot{L}_{rx} + L_{rz}\omega_y - L_{ry}\omega_z \\ \dot{L}_{ry} + L_{rx}\omega_z - L_{rz}\omega_x \\ \dot{L}_{rz} + L_{ry}\omega_x - L_{rx}\omega_y \end{bmatrix} \tag{3.27}$$

Estas ruedas de reacción producen momentos en los tres ejes en los que son montadas, haciendo rotar al satélite en la orientación deseada [30] [31].

Si las ruedas se ocupan para realizar rotaciones alrededor del punto de orientación deseado, entonces los pares se puede simplificar usando la siguiente ecuación.

4. LEY DE CONTROL

4.1. Ley de control: posición más velocidad angular

Para llevar a cabo el control de orientación del satélite, la ley de control a utilizar parte de la ley que se llama posición más velocidad angular, la cual tiene la siguiente forma [32].

$$\tau = -k_1\omega - k_2\theta \tag{4.1}$$

Donde θ es el error y ω la velocidad en cada uno de los ejes principales de inercia. De esta manera el torque es proporcional a la señal de error y a su derivada con respecto al tiempo; el término $-k_1\omega$ proporciona un amortiguamiento. Las constantes k_1 y k_2 son ganancias que dependen del momento de inercia del satélite. Es necesario contar con giróscopos para determinar la velocidad angular o utilizar un método de derivación para calcular ω.

Se sabe que:

$$\tau = I_r \frac{d\omega}{dt} \tag{4.2},$$

donde I_r es el valor del momento de inercia de la rueda y τ es el torque que ésta genera debido a un cambio en su velocidad angular.

Integrando:

$$\tau \int dt = I_r \int d\omega \tag{4.3}$$

$$\tau\Delta t = I_r(\omega_f - \omega_i) \tag{4.4},$$

donde $\tau\Delta t$ es el impulso angular que se produce debido a un cambio en la velocidad angular de la rueda.

Por otra parte, para el caso de la plataforma de simulación se tiene que

$$\tau = I_S \frac{d\omega}{dt} \tag{4.5}$$

donde I_S es el momento de inercia del sistema de simulación.

Integrando como se hizo anteriormente se obtiene.

$$\tau\Delta t = I_s(\omega_f - \omega_i) \tag{4.6}$$

Donde $\tau\Delta t$ nuevamente es el impulso angular que se produce por un cambio en la velocidad angular de la plataforma de simulación satelital.

Dado que las ruedas se encuentran ancladas a la plataforma de simulación satelital utilizando el principio de conservación de momento angular y las ecuaciones simplificadas de la rueda (3.9) se tiene.

$$I_s(\omega_f - \omega_i) = I_r(\omega_f - \omega_i) \tag{4.7}$$

Esta ecuación muestra que si ocurre un cambio en la velocidad angular de la rueda inercial que se encuentra en el satélite, se tiene que producir un cambio de velocidad en la plataforma de simulación para conservar la cantidad de movimiento angular.

De esta manera para hacer girar a la plataforma de simulación se hará uso de impulsos angulares. Multiplicando la ecuación (4.1) por Δt se obtiene la ley de control utilizando dichos impulsos:

$$\tau \Delta t = (-k_1\omega - k_2\theta)\Delta t \tag{4.8}$$

Por lo que queda determinar experimentalmente las constantes k_1 y k_2 lo cual se describirá en el siguiente capítulo.

La ventaja de utilizar esta ley de control que se ha desarrollado, es que los cálculos matemáticos efectuados no requieren de mucho procesamiento matemático, y se puede alcanzar la orientación dentro del error establecido que es de ±2°. Esta ley de control se trata de un algoritmo de control proporcional, esto quiere decir que, cuando el satélite se va acercando a la orientación deseada, el algoritmo de control va disminuyendo el valor de los impulsos que se envían a los actuadores.

5. REQUERIMIENTOS DE HARDWARE DEL SISTEMA DE CONTROL

La conformación de cada uno de los elementos del hardware, están encaminados a lograr que el control en la orientación sea de ± 2°. Como se menciona en [16], el error en el control de la orientación, en general, es función tanto de los errores en la determinación de la orientación como de los errores de control, porque el sistema de control toma sus entradas del sistema de determinación de orientación.

5.1. Prototipo de satélite

5.1.1. Sensores

Dado que el control de orientación se va a realizar en tres ejes, es necesario contar con sensores colocados a lo largo de los mismos. Estos ejes se eligen en el satélite de tal manera que conformen un sistema ortogonal, como se muestra en la figura 5-1. Así, los actuadores y los sensores se encuentran alineados en los mismos ejes. Se cuenta con un magnetómetro de tres ejes para medir la orientación del satélite y un giróscopo también de tres ejes para medir la velocidad angular de cada uno de ellos. La precisión del magnetómetro es menor a 0.5°, este valor es importante porque determina el error máximo en la orientación medida. Véase el anexo F para la información técnica de los sensores.

Figura 5-1 Sensores instalados en el prototipo de satélite. Se muestran los sistemas de referencia asociados a los sensores.

5.1.2. Actuadores

Los actuadores que son utilizados en este sistema, para cambiar la orientación del nanosatélite son las ruedas de reacción. En este caso se utilizan tres, que es el número mínimo

de actuadores requeridos para tener el control de orientación del sistema en cualquier dirección. Mas adelante se determinará la precisión en la orientación alcanzada por cada una de estas ruedas, que sumada a la precisión del magnetómetro nos dará la precisión total del sistema. Estas tres ruedas forman un sistema ortogonal como se muestra en la figura 5-2.

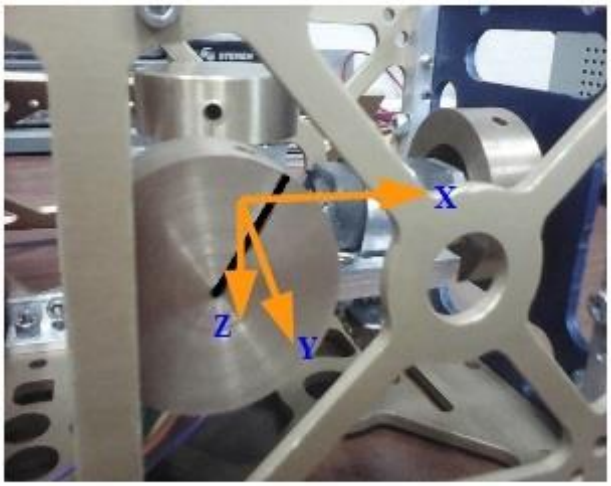

Figura 5-2 Ruedas de reacción colocadas ortogonalmente en el prototipo de satélite.

5.1.3. Computadora de a bordo

El propósito de la computadora de a bordo es leer los vectores de orientación y la velocidad angular entregada por los sensores, posteriormente, medir el error respecto de la orientación deseada, efectuar los cálculos y enviar las señales de control a los actuadores para hacer la corrección correspondiente. La computadora utilizada para el prototipo de satélite es una Raspberry Pi 2B. Esta microcomputadora corre un sistema operativo de Linux llamado Raspbian. Las características de esta computadora cumplen con los requerimientos en memoria y velocidad de procesamiento, para la ejecución del algoritmo de control. Por otra parte, el algoritmo de control fue escrito en un script de Python que corre sobre el sistema operativo Linux.

Figura 5-3 Computadora de a bordo (Raspberry Pi) montada en el prototipo de satélite.

5.1.4. Fuente de alimentación

El sistema utiliza dos baterías Lipo: una para alimentar a la computadora de a bordo y otra para las ruedas inerciales. Estas baterías están compuestas por tres celdas, cada una proporciona 3.7 V.

La computadora de a bordo es alimentada con 5 V, para ello se utiliza una fuente conmutada (SHT-164) para reducir el voltaje total de 11.1 V, de una de las baterías.

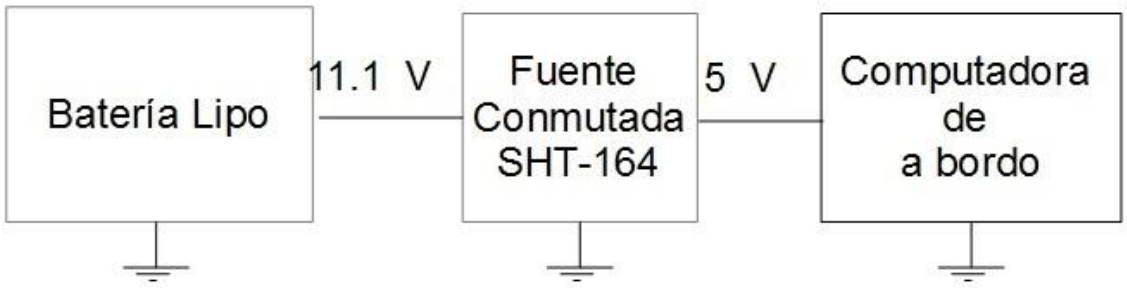

Figura 5-4 Diagrama de la fuente que alimenta a la computadora de a bordo del prototipo de satélite.

5.1.5. Motores

Los motores que mueven a las ruedas de reacción son conectados a una tarjeta de puentes H (Adafruit DC and Stepper Motor HAT) que establece una comunicación I2C con la computadora de a bordo. Esta tarjeta es alimentada con dos celdas de la segunda batería (7.4 V).

Los motores se pueden alimentar con un voltaje máximo de 5V, a este voltaje el consumo de corriente es de 1.2 A, esta corriente sobrepasa la que pueden manejar los puentes H, de modo

que la computadora de abordo genera una señal PWM que reduce el voltaje de alimentación a 2V. A este voltaje el consumo de corriente del motor es de 0.5 A, por lo que la potencia consumida por el motor es de 1W. La velocidad máxima que desarrolla el motor en estas condiciones es de 5500 revoluciones por minuto.

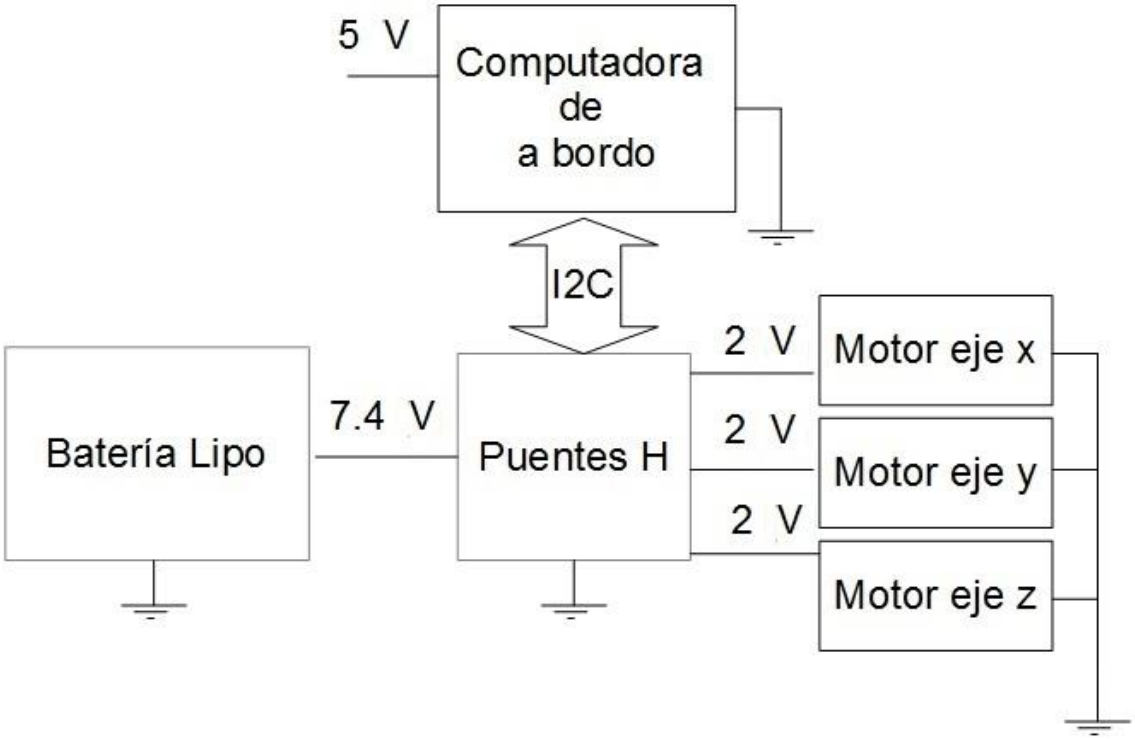

Figura 5-5 Fuente para alimentar a los puentes H.

5.2. Plataforma de simulación

Para realizar pruebas del control de orientación en Tierra se necesita contar con un sistema que tenga movimiento libre en tres ejes y que nos permita simular la falta de fricción que hay en el espacio, para ello se desarrolló un sistema como el que se muestra en las figuras (5.6, 5.7 y 5.8).

El sistema está conformado por una copa y una esfera hueca. El aire a presión, que se inyecta a la copa con un compresor, es expulsado por pequeños orificios de la misma, esto provoca que se forme un colchón de aire entre la copa y la esfera, ocasionando que ésta última flote, simulando así un medio sin fricción. Una vez que la esfera se encuentra soportada en el colchón de aire, esta gira 360° en los tres ejes.

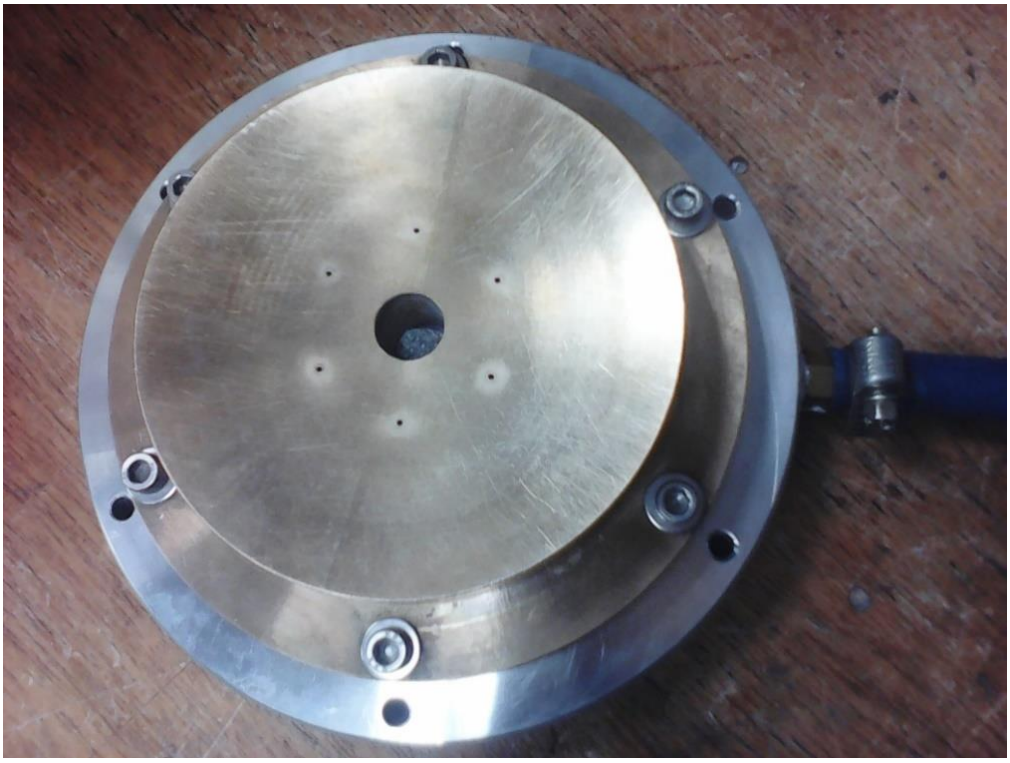

Figura 5-6 Copa que a través de seis orificios provee el aire a presión para sustentar la esfera en donde se coloca la estructura completa del nanosatélite.

Figura 5-7 Detalle del simulador satelital, la esfera está montada sobre la copa. Es en esta unión dónde se genera un medio sin fricción.

Figura 5-8 Prototipo de un nanosatélite CubeSat 3U, montado sobre un simulador satelital. En esta estructura se encuentra el sistema de control de orientación.

La esfera hueca ha sido diseñada para colocar dentro de ella a un satélite CubeSat 3U, cuyas dimensiones son de 30 cm x 10cm x 10 cm. El satélite permanece en cualquier posición dependiendo del comando que reciba el actuador con el que se estén realizando las pruebas, ya sea desde la computadora de abordo, o que dicho comando sea enviado durante una prueba efectuada desde la estación terrena.

5.3. Diseño de las ruedas inerciales

Para el diseño de las ruedas inerciales, se parte del principio de conservación de momento angular, comenzando con la ecuación (3.4) y considerado la ecuación (3.28). Para un eje de inercia principal del simulador se tiene que:

$$I_s \omega_s = I_r \omega_r \tag{5.1}$$

Esta ecuación nos dice que la velocidad angular ω_s desarrollada por el simulador multiplicado por su inercia I_s es igual al producto de la inercia de la rueda por la velocidad angular ω_r desarrollada por la rueda alrededor de un eje principal. En este caso las variables que se pueden conocer son: I_s, ω_s y ω_r. Por lo que, solo queda por determinar I_r, este valor de momento de inercia nos va a permitir determinar el tamaño de las ruedas de reacción.

Los cálculos para la determinación de los momentos de inercia de la plataforma de simulación que se encuentra conformada por la esfera hueca y el prototipo de satélite, se realizan de manera teórica. Para el primer elemento se toma el momento de inercia de una esfera hueca y para el segundo, considerando que el prototipo de satélite contiene elementos sólidos, una aproximación es considerarlo como un prisma rectangular sólido.

El momento de inercia de una esfera hueca se calcula con:

$$I = \frac{2}{3} mr^2 \tag{5.2}$$

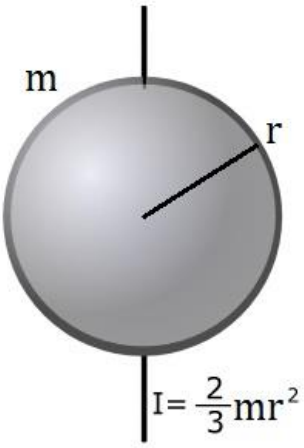

Figura 5-9 Momento de inercia de una esfera hueca.

La masa de la esfera es de 1.8 kg y el radio es de 36.7 cm, sustituyendo estos valores en la ecuación 5.2 se obtiene que la inercia de la esfera hueca es de $I_e = 0.1616\ kgm^2$

Los momentos de inercia para un prisma rectangular sólido en la dirección de los tres ejes ortogonales mostrados en la figura son:

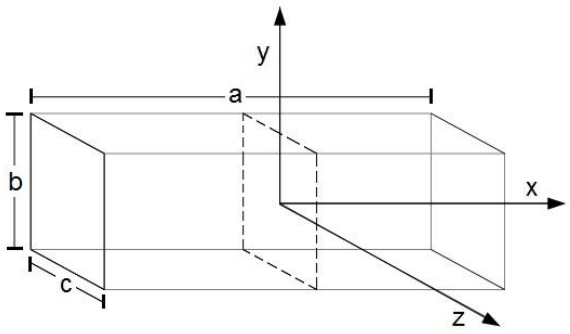

Figura 5-10 Ejes de momentos de inercia para un prisma rectangular sólido.

$$I_x = \frac{1}{12} m(c^2 + b^2) \quad (5.3)$$

$$I_y = \frac{1}{12} m(a^2 + c^2) \quad (5.4)$$

$$I_z = \frac{1}{12} m(a^2 + b^2) \quad (5.5)$$

Para el caso del satélite CubeSat 3U se tienen los siguientes datos:

a=0.3 m, b=0.1 m, c=0.1 m y m=2.025 kg

Sustituyendo los valores en las fórmulas se obtiene:

$$I_x = 0.003375\ kgm^2$$

$$I_y = 0.016875\ kgm^2$$

$$I_z = 0.016875\ kgm^2$$

De manera que la inercia total en cada uno de los ejes principales es la suma de los momentos de inercia de la esfera más los momentos de inercia del prisma rectangular por lo que se tiene:

$$I_y = I_z = 0.1616 + 0.016875 = 0.1785\ kgm^2$$

$$I_x = 0.1616 + 0.003375 = 0.165\ kgm^2$$

Para la determinación de las velocidades angulares se procedió como sigue. La velocidad angular ω_s es la máxima que puede alcanzar la plataforma de simulación y corresponde a la velocidad que podrán contrarrestar las ruedas en el sistema de control de orientación. Si las ruedas sobrepasan esta velocidad, es necesario contar con una fuerza externa que la contrarreste, en este caso se utiliza un conjunto de bobinas magnéticas. La velocidad angular ω_s máxima que se ha establecido en este sistema es de 5°/s. La velocidad máxima desarrollada por las ruedas, de acuerdo al valor de voltaje utilizado, se menciona en la sección (**5.1.5.**) y es de 5500 rev/minuto.

5.3.1. Determinación de las dimensiones de las ruedas de reacción

Para la determinación de las dimensiones de la rueda inercial se determinó que ésta tuviera la forma que se muestra en la figura (5-11). Para calcular el momento de inercia de esta rueda se utiliza una geometría que se le aproxime (cilindro hueco con radio externo r_2 y radio interno r_1) y cuyo momento de inercia se obtiene de la siguiente manera:

$$I_z = \frac{1}{2} m(r_2^2 + r_1^2) \quad (5.6)$$

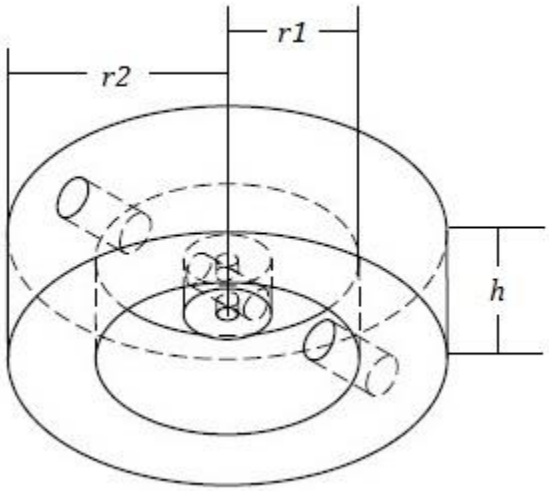

Figura 5-11 Geometría de las ruedas de reacción, dónde r1 es el radio interno, r2 el radio externo y h la altura.

De esta manera, definiendo la altura h de la rueda y el radio externo r_2, se define el radio interno r_1. Para incluir la altura y la densidad en el cálculo de la rueda se procede como sigue:

El volumen del cilindro hueco es:

$$V = \pi r_2^2 h - \pi r_1^2 h \tag{5.7}$$

Para calcular la masa se tiene que:

$$m = \rho V \tag{5.8}$$

De modo que sustituyendo el valor de V en la ecuación de la masa se tiene:

$$m = \rho \pi h (r_2^2 - r_1^2) \tag{5.9}$$

Sustituyendo el valor de la masa en la ecuación de inercia del cilindro hueco:

$$I_r = \frac{1}{2} \rho \pi h (r_2^2 - r_1^2)(r_2^2 + r_1^2) \tag{5.10}$$

Despejando r_1 :

$$r_1 = \sqrt[4]{r_2^4 - \frac{2I_r}{\rho \pi h}} \tag{5.11}$$

Del principio de conservación del momento angular (ecuación 5.1), tomando los valores máximos de velocidad angular tanto de la plataforma de simulación, como de la rueda de reacción, se obtiene que:

$$I_r = I_s \frac{\omega_{\max\_s}}{\omega_{\max\_r}} \tag{5.12}$$

Sustituyendo el valor de I_r en la ecuación 5.11 se tiene:

$$r_1 = \sqrt[4]{r_2^4 - \frac{2I_s}{\rho\pi h} \cdot \frac{\omega_{maxs}}{\omega_{maxr}}} \tag{5.13}$$

Esta ecuación permite calcular el radio interno r_1 de la rueda habiendo definido con anterioridad el radio externo r_2, la densidad del material a utilizar ρ, la altura de la rueda h, así como el valor de inercia de la plataforma de simulación I_s, la velocidad angular máxima a la que debe de girar la plataforma ω_{maxs} y la velocidad angular máxima que desarrollan las ruedas de reacción ω_{maxr}.

Calculo de r_1:

$r_2 = 0.02\ m$

$I_s = 0.016875\ kg\ m^2$

$\rho = 8500\ kg/m^3$

$h = 0.014\ m$

$\omega_{maxs} = 5\ °/s = 0.833\ rev/min$

$\omega_{maxr} = 5500\ rev/min$

Sustituyendo los valores en la ecuación (5.13) se obtiene un valor de:

$r_1 = 0.01113\ m\ \approx 0.011\ m$

5.4. Caracterización dinámica de la plataforma de simulación

Para caracterizar y obtener la dinámica de la plataforma de simulación satelital se utiliza el principio de conservación de momento angular, de modo que se tiene:

$$I_s(\omega_f - \omega_i) = I_r(\omega_f - \omega_i) \tag{5.14}$$

Donde el segundo miembro de la ecuación es igual a:

$$\tau\Delta t = I_r(\omega_f - \omega_i) \tag{5.15}$$

Esto nos dice que al aplicar un impulso angular a la rueda inercial se produce un cambio de velocidad en las ruedas; este cambio de velocidad produce de igual manera un cambio de velocidad en la plataforma de simulación, con el consiguiente cambio de orientación angular.

De esta manera se elaboró un programa mostrado en el apéndice C, mediante el cual se enviaron impulsos angulares a las ruedas y se midió tanto la velocidad angular resultante, como el desplazamiento angular causado por cada uno de ellos. En la gráfica 5-12 se puede ver la forma del impulso; un par τ aplicado en un tiempo Δt.

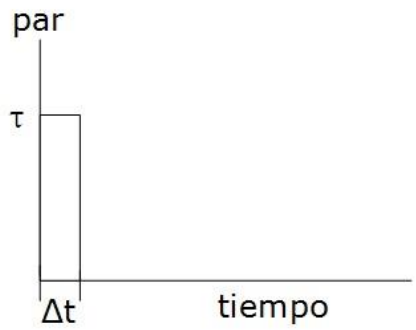

Figura 5-12 Forma del impulso angular aplicado a la plataforma de simulación para su caracterización.

Las gráficas 5-13, 5-14 y 5-15 nos muestran los desplazamientos y las velocidades angulares que son ocasionadas en la plataforma de simulación para cada impuso aplicado. Estas gráficas también permitieron obtener las constantes k1 y k2 del sistema de control que fue planteado en el capítulo 4.

Dado que los momentos de inercia de la plataforma de simulación son iguales para los ejes Y y Z y difieren del eje X, las pruebas se realizaron sobre los ejes Z y X, ya que, para cada valor de momento de inercia se obtienen constantes de control diferentes. Primero se muestran algunas gráficas de caracterización sobre el eje Z y posteriormente, sobre el eje X.

5.4.1. Pruebas sobre el eje Z

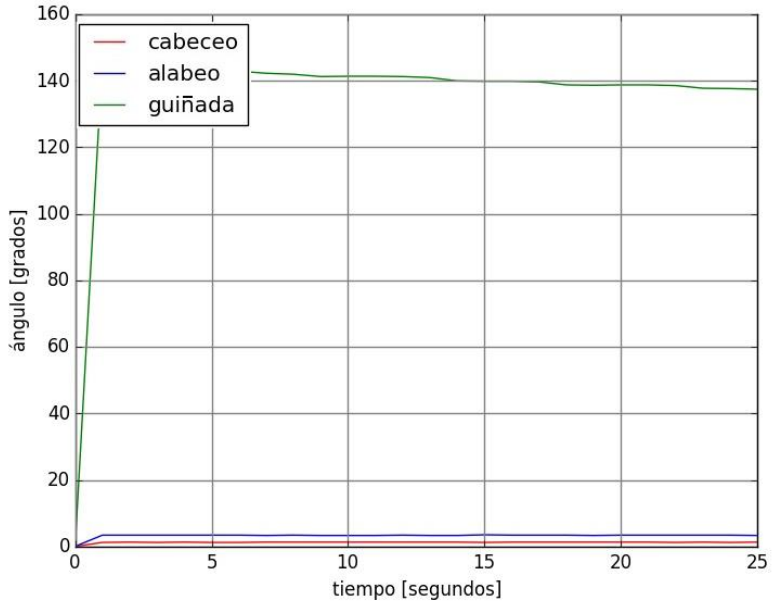

Figura 5-13 Impulso de 0.3 normalizado aplicado sobre el eje Z (guiñada).

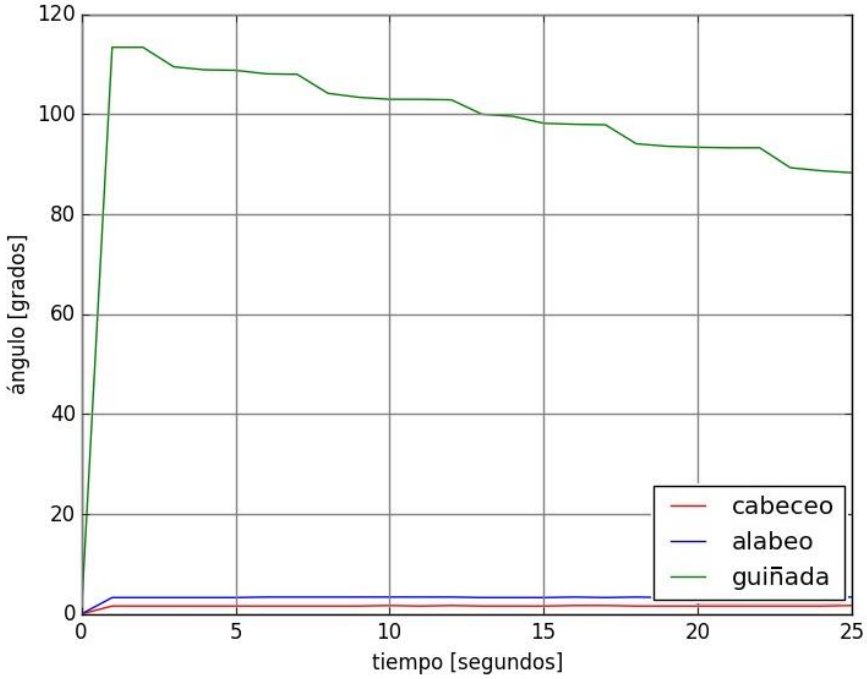

Figura 5-14 Impulso de 0.5 normalizado aplicado sobre el eje Z (guiñada).

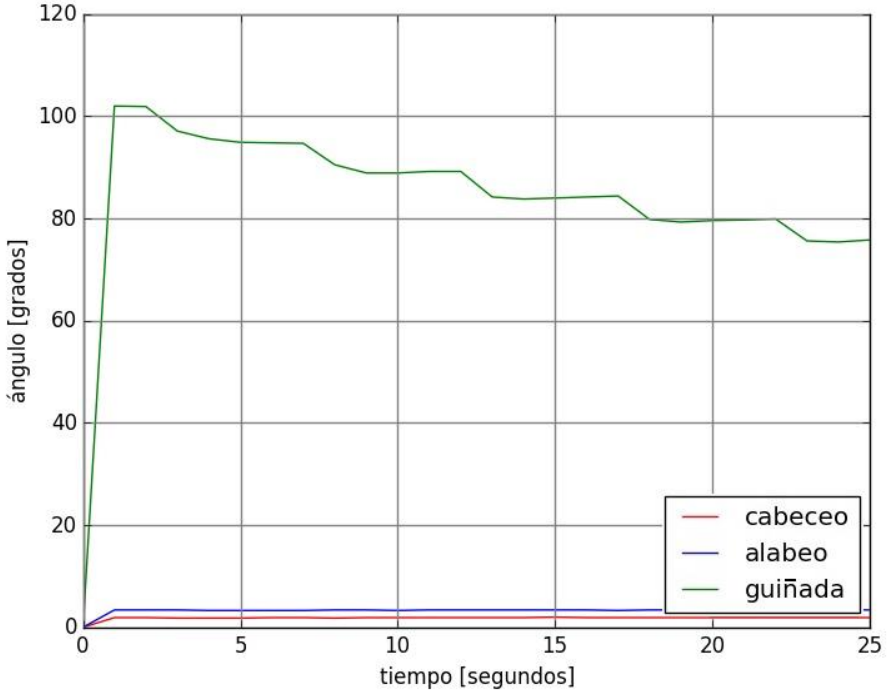

Figura 5-15 Impulso de 0.8 normalizado aplicado sobre el eje Z (guiñada).

De estas pruebas de caracterización se obtiene la tabla 5-1:

Impulso angular	Desplazamiento angular °	Velocidad angular °/s
0.2	0.92	0.92
0.3	1.6	1
0.5	4.98	4.98
0.7	5.7	5.7
0.8	7.12	7.12
0.9	8.5	8.5

Tabla 5-1 Valores de caracterización de la plataforma de simulación sobre el eje Z.

A continuación, se muestran las gráficas de los valores de desplazamiento angular y velocidad angular contra el impulso angular normalizado, en ellas se ha realizado una regresión lineal para aproximar los puntos a una recta. El algoritmo para realizar la regresión se encuentra descrito en el anexo B.

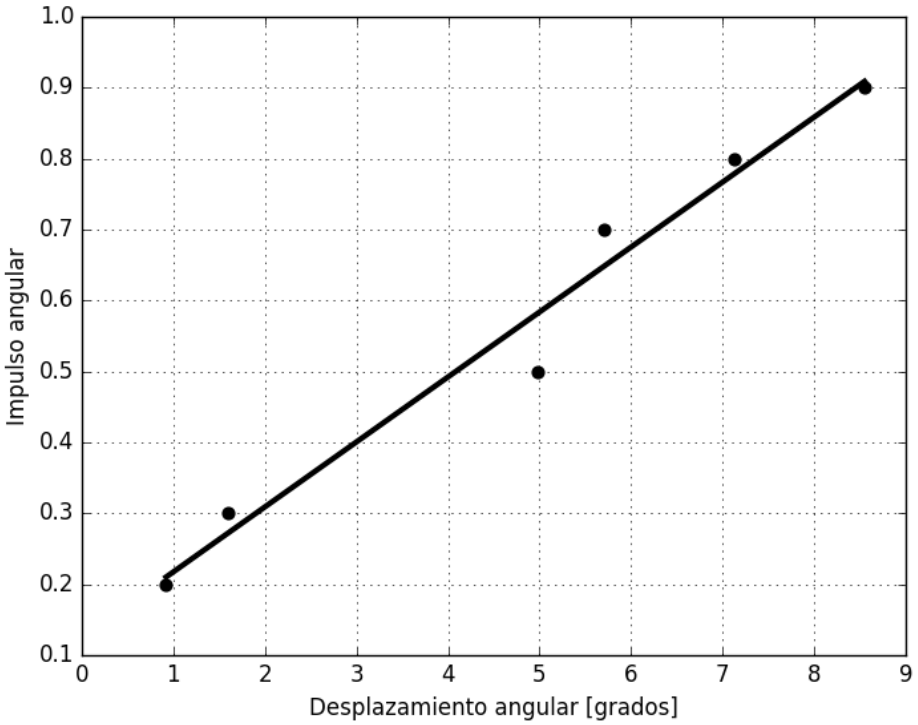

Figura 5-16 Gráfica para obtener las constantes k2 y c2 para los ejes Y y Z.

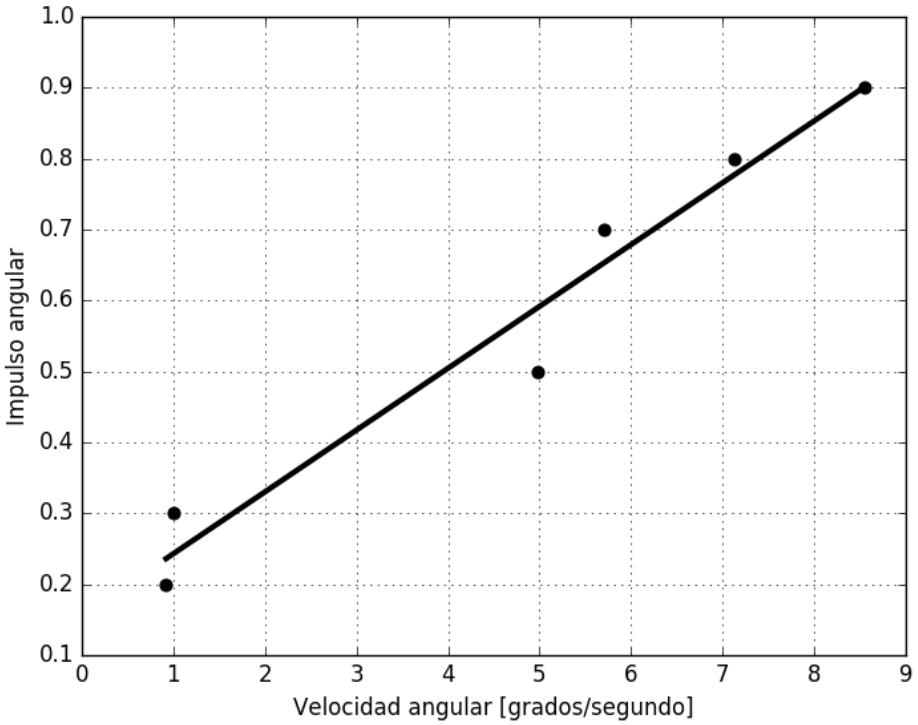

Figura 5-17 Gráfica para obtener las constantes k1 y c1 para los ejes Y y Z.

La constante k1 se obtiene del valor de la pendiente de la gráfica de impulso angular en función de la velocidad angular y la constante k2 del valor de la pendiente de la gráfica de impulso angular en función de desplazamiento angular.

Debido a que las ruedas de reacción presentan inercia, es decir, se necesita un valor de impulso angular mayor a cero para que estas comiencen a girar, se necesitan agregar dos términos más al algoritmo de control, valores que corresponden a la ordenada al origen, para cada una de las gráficas (c1, c2).

Los valores de las constantes obtenidas son mostrados en la tabla 5-2:

Gráfica de velocidad angular	Gráfica de desplazamiento angular
k1 = 0.0805	k2 = 0.0916
c1 = 0.203	c2 = 0.126

Tabla 5-2 Constantes de la señal de control para el eje Y y Z.

De este modo, la señal de control queda de la siguiente forma:

$$\tau\Delta t = (-k_1\omega - c1 - k_2\theta - c2)\Delta t \tag{5.16}$$

Las constantes c1 y c2 establecen un valor para vencer la inercia de las ruedas cuando el error de desviación angular o la velocidad angular es muy pequeño.

5.4.2. Pruebas sobre el eje X

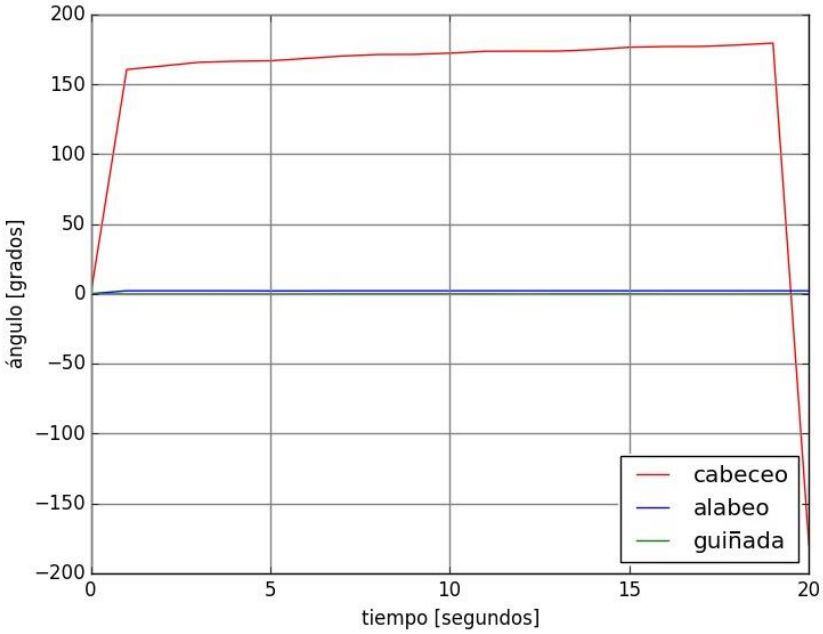

Figura 5-18 Impulso de 0.3 normalizado aplicado sobre el eje X (cabeceo).

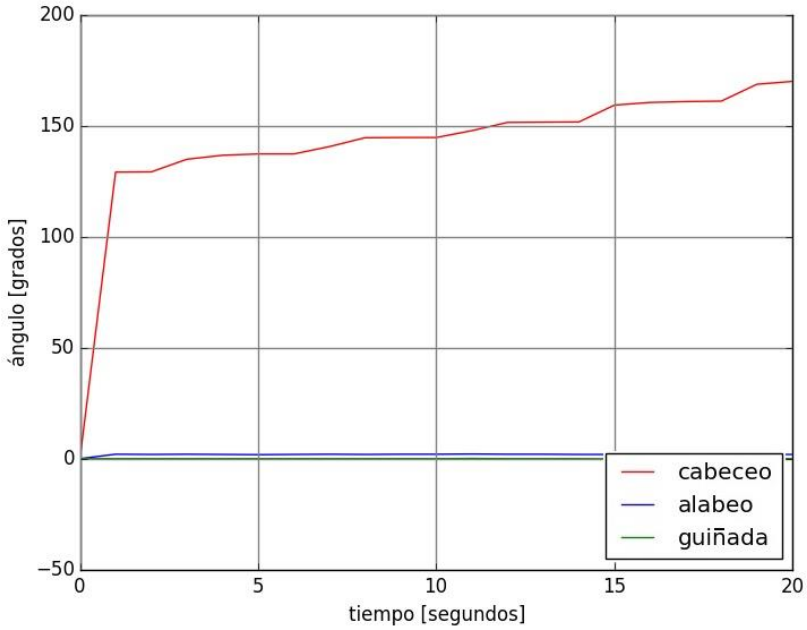

Figura 5-19 Impulso de 0.5 normalizado aplicado sobre el eje X (cabeceo).

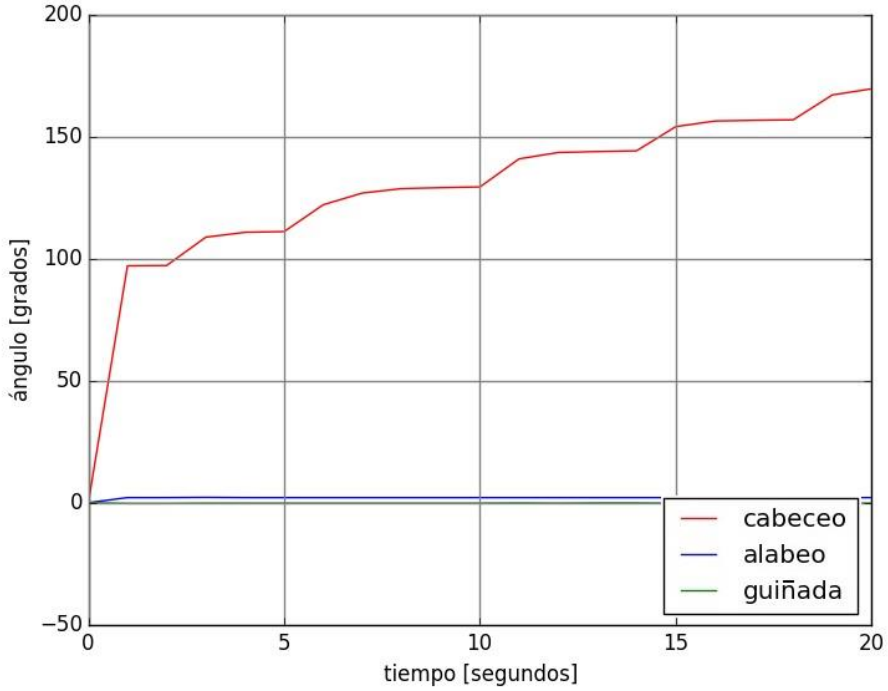

Figura 5-20 Impulso de 0.9 normalizado aplicado sobre el eje X (cabeceo).

De las gráficas 5-18, 5-19 y 5-20 se obtiene la tabla 5-3:

Impulso angular	Desplazamiento angular °	Velocidad angular °/s
0.1	1.5	0.6933
0.3	4.17	1.912
0.5	8	7.21
0.8	13.8	14.72
0.9	15	16

Tabla 5-3 Valores de caracterización de la plataforma de simulación sobre el eje X.

De estas graficas se obtienen las constantes para el control sobre el eje X (cabeceo) que tiene un momento de inercia diferente a los ejes de Y (alabeo) y Z (guiñada).

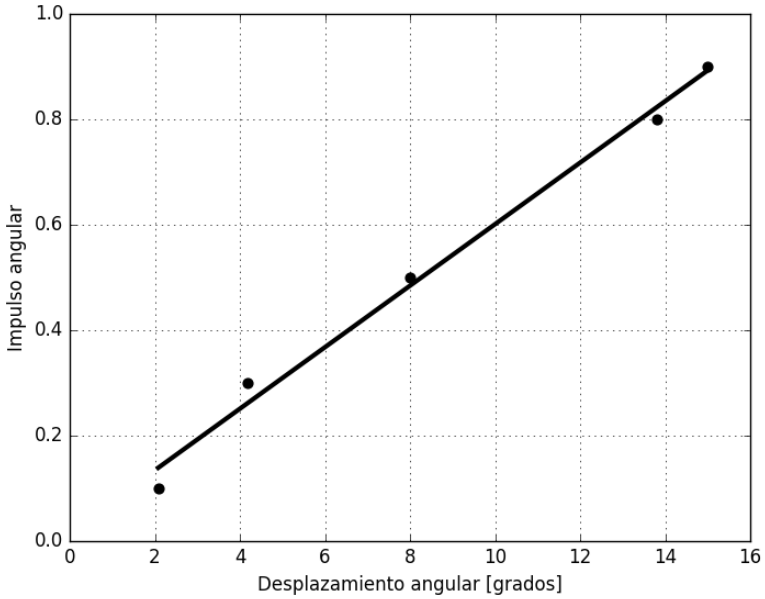

Figura 5-21 Gráfica para obtener las constantes k2 y c2 para el eje X.

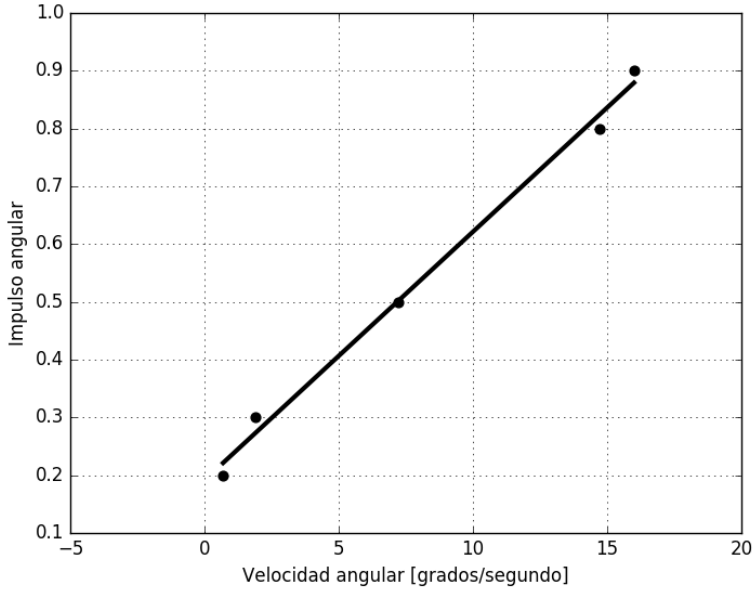

Figura 5-22 Gráfica para obtener las constantes k1 y c1 para el eje X.

Los valores de las constantes obtenidas son mostrados en la tabla 5-4:

Gráfica de velocidad angular	Gráfica de desplazamiento angular
k1 = 0.043	k2 = 0.0578
c1 = 0.1915	c2 = 0.0238

Tabla 5-4 Constantes de la señal de control para el eje X.

De este modo la señal de control para el eje X tiene la misma forma que la de la ecuación (5.16).

De los resultados obtenidos en las tablas, se observa que, para el impulso mínimo aplicado a la rueda de reacción, en el eje que tiene el menor momento de inercia, hay un desplazamiento de 1.5 °, este valor corresponde a la precisión en la orientación alcanzada por la rueda de reacción, este valor sumado a la precisión del magnetómetro (menor a 0.5°), nos da un valor máximo de 2°, el cual establece la precisión que es alcanzada en el control de orientación como se establecía al comienzo del capítulo.

6. ALGORITMO DE CONTROL

Inicialmente el algoritmo de control proporcional se implementó y simuló en Matlab para probar su funcionamiento, posteriormente se realizó la prueba del algoritmo en un simulador físico. En la sección **6.2** se da una revisión de un algoritmo LQR.

6.1. Algoritmo de control proporcional en Matlab

A continuación, se presenta la simulación del algoritmo de control implementado en Simulink de Matlab.

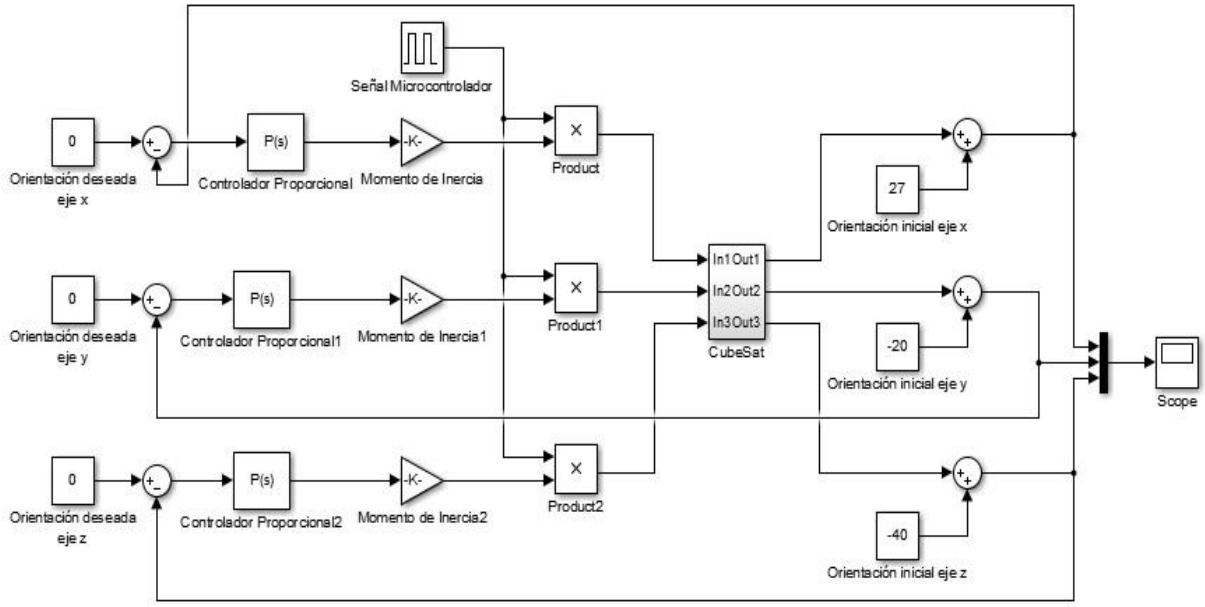

Figura 6-1 Algoritmo de control de orientación en Simulink de Matlab.

En el diagrama 6-1 se muestra la implementación un sistema de control de orientación proporcional en tres ejes, el cual utiliza impulsos angulares para llevar a cabo el control. Inicialmente se establece la orientación deseada, la cual corresponde a (0,0,0) en grados, así como el error de orientación inicial del satélite. Dado que se está trabajando con los ejes principales de inercia, se necesita introducir el valor de inercia de cada eje, que se calculó en la sección **5.3**.

El modelo de la planta corresponde a la ecuación (3.2). De manera tal que por cada impulso que es aplicado al sistema, hay un cambio en la cantidad de momento angular, que se traduce en un cambio de orientación del satélite. Si el error en el satélite es cada vez menor, el impulso aplicado es menor, al igual que el cambio en momento angular mismo que ocasiona un cambio de orientación menor. Estos resultados son mostrados en las gráficas de

caracterización del sistema de la sección **5.4**, en ellas se observa que si el impulso es menor el cambio de la orientación es menor.

El resultado de la simulación se despliega en la figura 6-2.

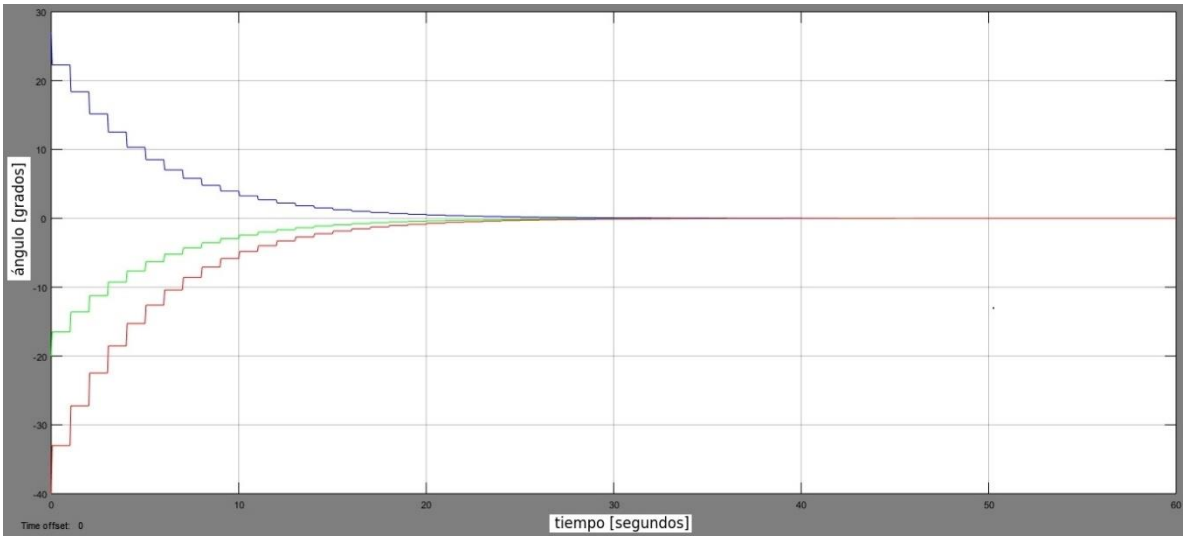

Figura 6-2 Respuesta del algoritmo de control proporcional. Eje x (azul). Eje y (verde). Eje z (rojo).

En la gráfica puede verse que, partiendo de diferentes errores de orientación para cada uno de los ejes principales del satélite, el sistema alcanza la orientación deseada. También que la respuesta de este control proporcional es subamortiguada.

6.2. Implementación del algoritmo en la computadora de a bordo

Para realizar pruebas físicas del algoritmo de control se necesita contar con una plataforma de simulación (**sección 5.2**) y el prototipo de satélite (**sección 5.1**) el cual cuenta con una computadora de a bordo que ejecuta el algoritmo de control. Este algoritmo se encarga de medir el error en la orientación del satélite con los sensores y de enviar las señales necesarias a los actuadores para llevar al satélite a la orientación deseada.

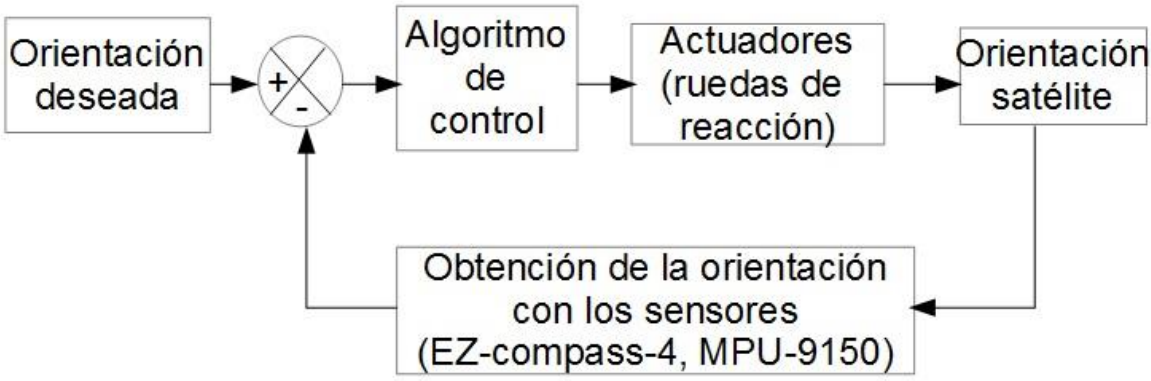

Figura 6-3 Diagrama de bloques del sistema de control de orientación del satélite.

El algoritmo de control se programó en lenguaje Python, ya que este cuenta con diversas librerías para implementar funciones de control. Estas están implementadas en Fortran al igual que las librerías utilizadas por Matlab. Para poder ser utilizadas en Python necesitan ser compiladas a este lenguaje. Esto nos permite disminuir el tiempo de desarrollo del sistema. Las librerías de Python que se utilizan para realizar operaciones con matrices y bases de datos para generar gráficas son las siguientes: scipy, slycot y matplotlib.

El diagrama de flujo del algoritmo se muestra en la figura (6-4). El algoritmo funciona de la siguiente manera: una vez que se han importado las librerías e inicializado los valores de las constantes obtenidas experimentalmente, lo que procede es tomar las mediciones de orientación con el sensor ez-compass-4 y la velocidad angular con el sensor mpu-9150. Una vez que se obtienen estos valores, se pasa a un ciclo de comparación que comprende tres grupos, que tienen que ver con la orientación del sistema. El control proporcional solo esta implementado para un error de ±20° esto se debe a que para generar un impulso que haga una corrección fuera del rango se requiere que haya un cambio de velocidad que ya no puede generar el sistema debido a las condiciones de voltaje (**sección 5.1.5.)** con las que ha sido alimentado el sistema. De manera tal que para errores mayores a ±20° se mantiene un valor de impulso constante para que una vez que la orientación se encuentre dentro del rango de ±20° entre en funcionamiento el control proporcional. Si el sistema se encuentra dentro del error de ±2°, el algoritmo solamente se encarga de verificar continuamente la orientación, de tal forma que, si el sistema sale de este rango, el algoritmo activa nuevamente las señales de control en los actuadores para corregir la orientación,

En el algoritmo están separadas las comparaciones que comprenden de ángulos mayores a 20°, a ángulos menores a -20° y a los intervalos que van de (2°, 20°) y (-2°, -20°), esto se realiza de esta forma, debido a que, para corregir la orientación, el sentido de giro de la rueda de reacción cambia si el error es positivo o negativo. Para un error positivo la rueda de reacción gira en el sentido horario y viceversa.

Cabe señalar que este algoritmo solo opera sobre un eje, para tener el control sobre los tres ejes, se replica este mismo modelo, y las señales de control son enviadas a la rueda de reacción que corresponda con el eje en cuestión.

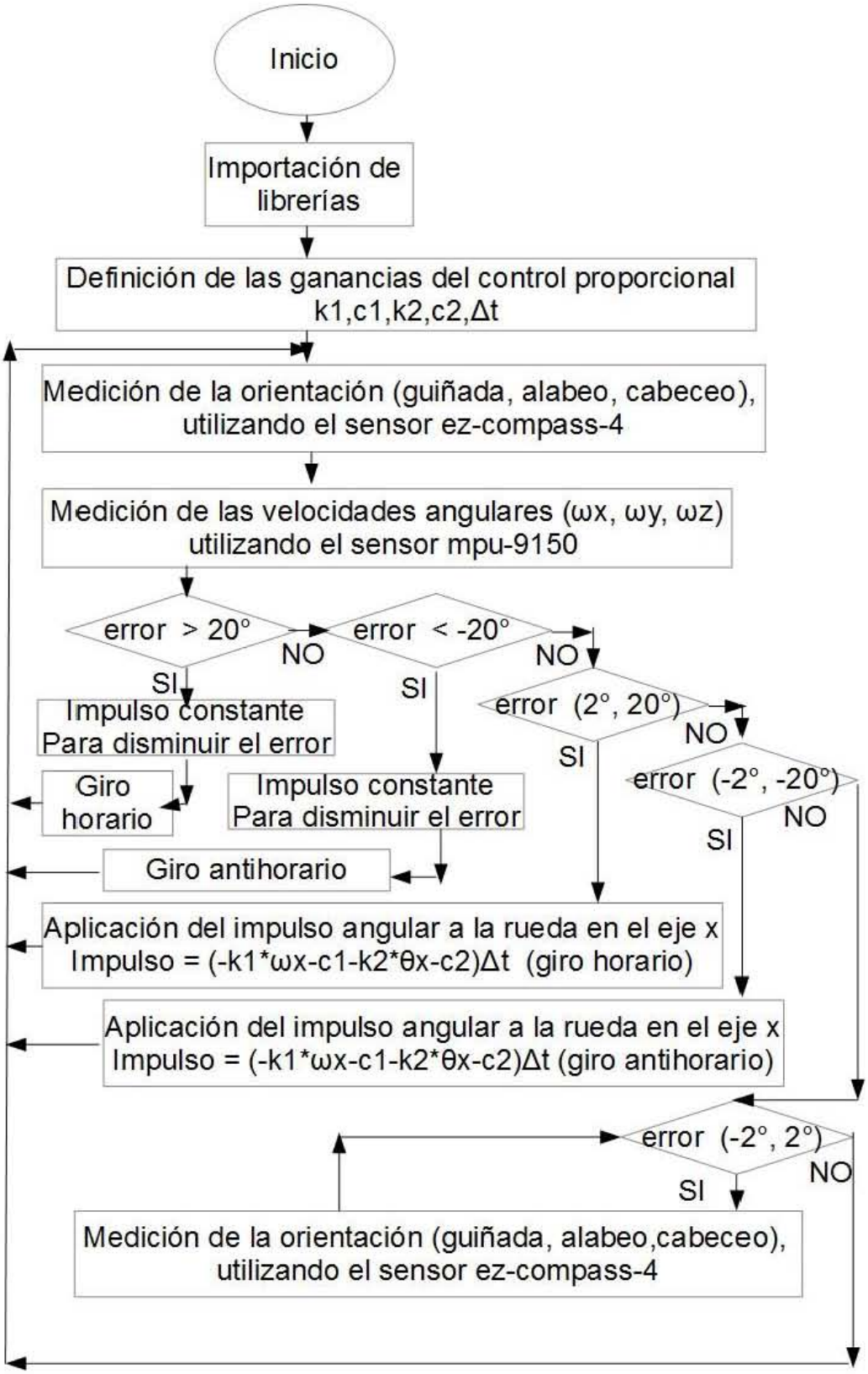

Figura 6-4 Diagrama de flujo del algoritmo de control de orientación.

6.3. Algoritmo de control óptimo LQR (Regulador Lineal Cuadrático)

Para la implementación de un algoritmo LQR se debe contar con un modelo en variables de estado del sistema y al igual que en el caso del control proporcional se deben de conocer tanto la orientación como las velocidades angulares en cada uno de sus ejes principales de inercia.

La técnica de control LQR pertenece al control óptimo y lo que se busca es hacer que la dinámica del satélite opere al menor costo, es decir, que el camino para lograr la orientación se da a través de una geodésica.

El diseño de este controlador plantea la siguiente ecuación de costo, la cual debe de ser mínima [17,33]:

$$J(u) = \frac{1}{2} \int_{t_0}^{T} [\tilde{x}^T \, Q \, \tilde{x} + u^T \, R \, u \,] dt \tag{6.1}$$

Donde:

$\tilde{x} = x - \, x_d$. $\tilde{x}$ es el error entre los estados reales menos los estados deseados.
$Q = diag[q_1, q_2, \dots, q_{ns}]$, con ns igual al número de estados.
$R = diag[r_1, r_2, \dots, r_{na}]$, con na igual a la cantidad de actuadores.

Para el diseño del controlador se emplean los siguientes pasos.

1) Se propone $Q = Q^T \geq 0$ y $R = R^T > 0$.
2) Se resuelve la siguiente ecuación matricial para $P = P^T > 0$

$$PA + A^T P - PBR^{-1}B^T P = \, -Q \tag{6.2}$$

3) Calcular

$$K = R^{-1}B^T P \tag{6.3}$$

El método descrito en los pasos anteriores se calcula con el comando $K = lqr(A, B, Q, R)$ que es similar tanto en Matlab como en la librería de control de Python. La diferencia es que en Matlab solamente se obtiene la simulación y en Python el algoritmo de control queda programado directamente en la computadora de a bordo. La constante K es sometida, a un proceso de iteraciones sucesivas variando los valores de las matrices Q y R hasta obtener un resultado acorde con los requerimientos de apuntamiento del satélite.

La ecuación para el satélite, de acuerdo a un modelo en variables de estado es de la siguiente forma:

$$\dot{x} = \, Ax + Bu \tag{6.4}$$

Donde $x = [\varepsilon_1 \, \dot{\varepsilon}_1 \, \varepsilon_2 \, \dot{\varepsilon}_2 \, \varepsilon_3 \, \dot{\varepsilon}_3]^T$ es el vector de estados del sistema [30],

donde ε_1, ε_2 y ε_3, son las componentes en cuaterniones de un satélite y $\dot{\varepsilon_1}$, $\dot{\varepsilon_2}$ y $\dot{\varepsilon_3}$ son las componentes de la velocidad angular.

y $u = [\dot{L}_{rx}\ \dot{L}_{ry}\ \dot{L}_{rz}]^T$ es el vector de entradas.

A y B son las siguientes matrices:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ -4k_x\omega_0^2 & 0 & 0 & 0 & 0 & (1-k_x)\omega_0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -3k_y\omega_0^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & -(1-k_z)\omega_0 & 0 & 0 & -k_z\omega_0^2 & 0 \end{bmatrix} \tag{6.5}$$

$$B = \begin{bmatrix} 0 & 0 & 0 \\ \frac{1}{2I_{xx}} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \frac{1}{2I_{yy}} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2I_{zz}} \end{bmatrix} \tag{6.6}$$

Donde:

ω_0 es la velocidad angular del satélite.

$$k_x = \frac{I_{yy}-I_{zz}}{I_{xx}} \tag{6.7}$$

$$k_y = \frac{I_{xx}-I_{zz}}{I_{yy}} \tag{6.8}$$

$$k_z = \frac{I_{yy}-I_{xx}}{I_{zz}} \tag{6.9}$$

e I_{xx}, I_{yy}, I_{zz} los momentos principales de inercia.

La diferencia entre el control proporcional y el LQR es que, para realizar un control óptimo es necesario encender los tres actuadores al mismo tiempo, a diferencia del control proporcional en donde el control de orientación se hace sobre un eje a la vez. Las pruebas físicas se realizaron para el control proporcional debido a los efectos de la gravedad.

Las pruebas de simulación en Matlab son mostradas a continuación:

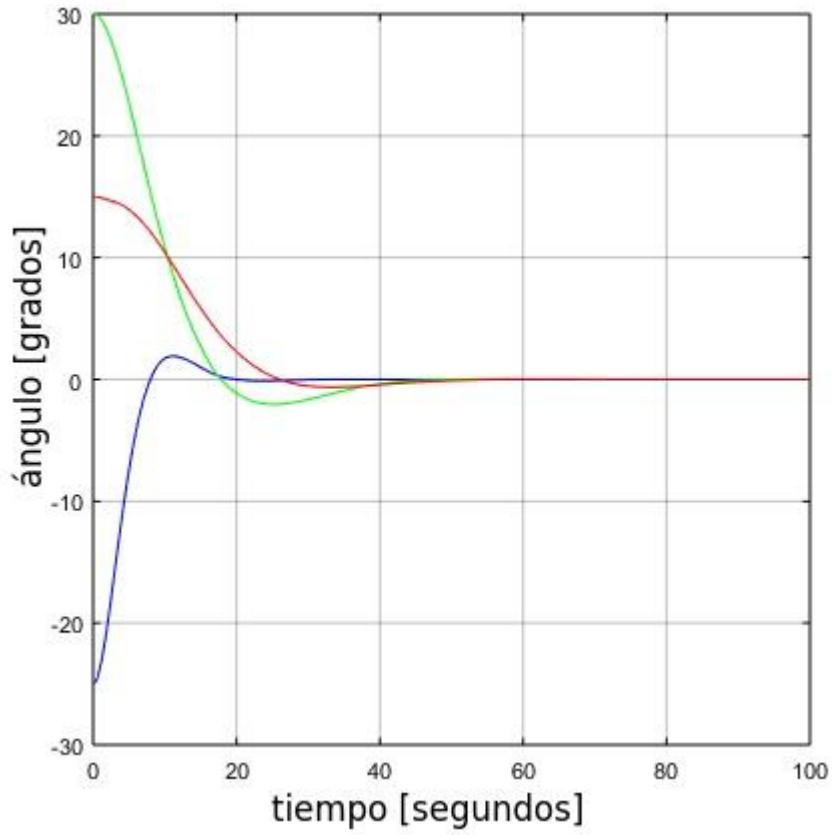

Figura 6-5 Respuesta del algoritmo de control LQR. Eje x (rojo). Eje y (verde). Eje z (azul).

De las gráficas se ve la orientación fijada que corresponde a los cero grados (0°, 0°, 0°) en los tres ejes, se alcanza en un tiempo similar al control proporcional mostrado anteriormente, pero a diferencia de este hay un ligero sobrepaso en cada uno de los ejes. En el anexo E se muestra el código en Python para obtener la matriz de ganancias del controlador LQR.

7. RESULTADOS Y CONCLUSIONES

7.1. Resultados

Para realizar las pruebas físicas del algoritmo de control se necesita establecer un sistema de referencia inercial. Para ello se ha utilizado el sistema de referencia del magnetómetro, el cual tiene al eje X apuntando hacia el norte magnético, el eje Y hacia el Este y el eje Z hacia el centro de la Tierra. Este sistema define la orientación 0,0,0. Las desviaciones angulares, es decir los errores en la orientación de cada uno de los ejes ortogonales en el satélite, son medidos con respecto a este sistema de referencia.

Una vez que se fija la orientación que corresponde a la orientación 0,0,0 de la brújula, se llevan a cabo las pruebas de simulación física en donde se muestra el funcionamiento del algoritmo de control.

Al realizar las pruebas en Tierra no se prueban los tres ejes de manera simultánea debido a que al ser diferentes los momentos de inercia en los tres ejes, hay una gran afectación debido a los momentos de fuerza que se ocasionan en la plataforma de simulación por la gravedad, de manera que las pruebas se realizaron sobre un eje a la vez. De esta forma lo que se hizo fue acomodar al satélite en el simulador físico de tal forma que el eje de prueba fuera el eje vertical.

A continuación, se muestran las pruebas sobre los ejes X, Y y Z.

Control sobre el eje x (cabeceo)

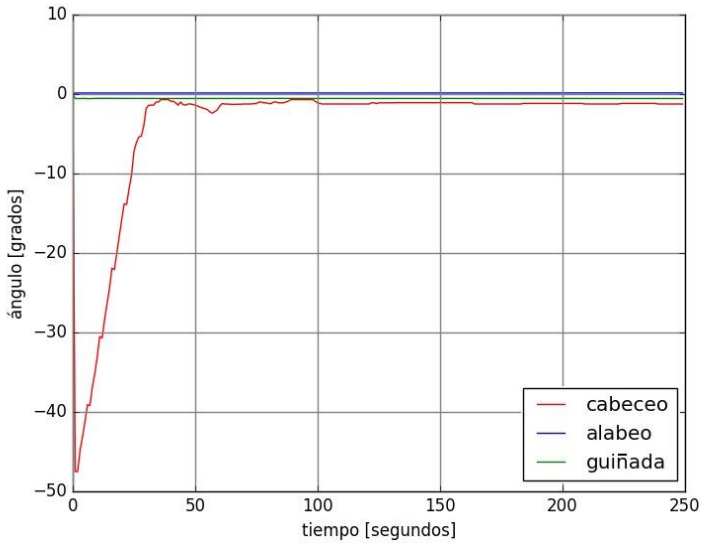

Figura 7-1 Control de orientación sobre el eje X (cabeceo).

Control sobre el eje y (alabeo)

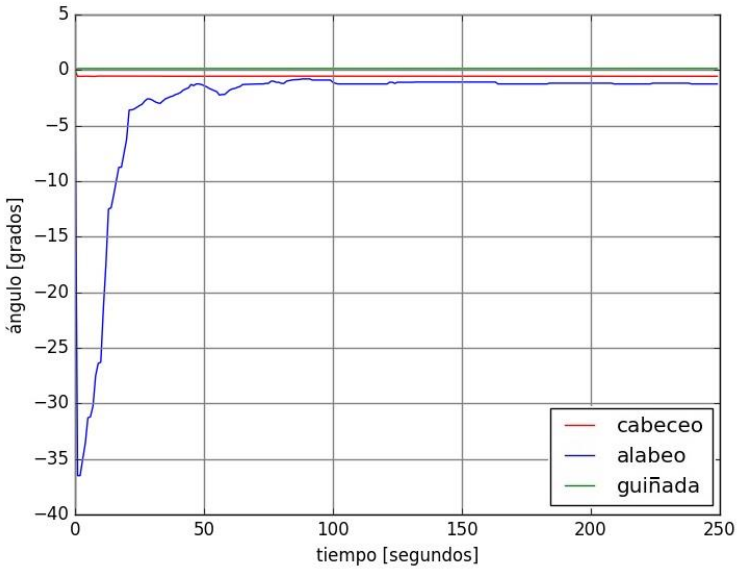

Figura 7-2 Control de orientación sobre el eje Y (alabeo).

Control sobre el eje z (guiñada)

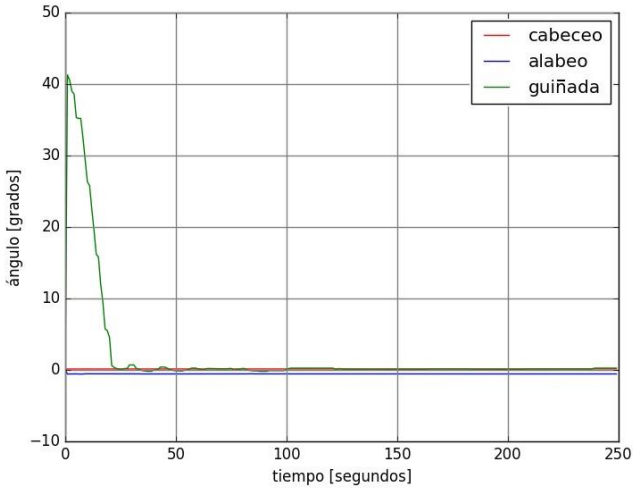

Figura 7-3 Control de orientación sobre el eje Z (guiñada).

De las gráficas obtenidas se observa que en el caso del control en el eje X, que parte de un error de 47°, alcanza la orientación fijada dentro del error permitido que es de ± 2° a los 30 segundos y la estabilización a los 60 segundos. En el caso del eje Y, parte de un error de 36° y alcanza la orientación deseada dentro del error permitido a los 42 segundos, la

estabilización ocurre a los 80 segundos. Para el caso del eje Z, el control parte de un error de 41° y alcanza la orientación deseada a los 25 segundos, y la estabilización del sistema a los 50 segundos.

En este caso, en el que las simulaciones se realizan en Tierra, no se requirió de utilizar varios sistemas de referencia como hubiese sido el caso de que el algoritmo fuera probado en un satélite que es enviado al espacio. Una de las razones por las que solamente se utiliza un sistema de referencia es porque en la plataforma de simulación no se simula la traslación del satélite, es decir las pruebas solamente se hacen simulando que el satélite se encuentra en un punto de la órbita a la vez. Para el caso de la prueba en órbita de este sistema de control se necesita contar con tres sistemas de referencia, un sistema de referencia fijo al satélite, un sistema de referencia orbital y un sistema de referencia inercial, además de contar con un método para determinar la orientación del satélite en el espacio para determinar el error en la orientación. En el sistema de control que ha sido probado en Tierra, la matriz de transformación entre cada uno de estos sistemas de referencia ha sido unitaria, de manera tal que, solo se ha utilizado el sistema de referencia que nos proporciona la brújula y no se ha utilizado un método para determinar la orientación puesto que el sensor EZ-compass-4 nos ha proporcionado la orientación del satélite con respecto al sistema de referencia que se eligió.

7.2. Conclusiones

El objetivo del presente trabajo fue elaborar un algoritmo de control de orientación que se pudiera implementar y probar en un simulador físico. Para lograrlo se tuvo que hacer una selección de los elementos que conformarían al sistema de control, lo cual incluye, los sensores adecuados para medir la orientación; la computadora de a bordo, la cual es la encargada de ejecutar el algoritmo; así como la selección de los actuadores. Como la aplicación para la cual se ha diseñado el control de orientación del prototipo de satélite es la percepción remota, se requirió de un sistema de control en tres ejes; para lograrlo se eligieron como actuadores tres ruedas de reacción, las cuales, colocadas ortogonalmente en la estructura del satélite, nos permiten tener este tipo de control. Se eligió una geometría en forma de anillo para las ruedas y se calculó el tamaño que deberían tener para que fueran capaces de mover a la plataforma de simulación. También se integró al sistema una fuente de alimentación para que no hubiera cables intermedios a la hora de realizar las pruebas del algoritmo. Esta fuente consistió de baterías y una fuente conmutada. Una vez que se contó con el sistema de control de orientación integrado en el prototipo de satélite se procedió a realizar las pruebas montándolo en la media esfera que flota sobre la copa, simulando el ambiente sin fricción que hay en el espacio.

Cada una de las pruebas fueron realizadas sobre un eje del satélite a la vez, ya que, al tener momentos de inercia diferentes hay pares gravitacionales que tienden a mantener al sistema en la posición de equilibrio, de manera que para cambiar o mantener una orientación sobre ejes diferentes a la línea de acción de la gravedad se necesita tener encendidos los actuadores.

De manera que, colocando al satélite de modo que coincidiera cada eje de prueba con la vertical, se pudieron llevar a cabo las pruebas de control de orientación sobre los tres ejes.

La precisión alcanzada con el algoritmo de control implementado fue de ±2°. Este valor puede mejorarse utilizando un algoritmo PI o un PID. Adicionalmente, utilizando este tipo de algoritmos se eliminan las oscilaciones que se aprecian en las gráficas que fueron presentadas en los resultados.

Para trabajo a futuro lo que se plantea es desarrollar una plataforma que sea lo más ligera posible, y elegir motores que tengan una velocidad de operación mayor con él objetivo de reducir el tamaño de las ruedas de reacción. La disminución en el tamaño de las ruedas ocasiona que el consumo de energía sea menor lo cual es muy importante para estos sistemas, en donde la disponibilidad de potencia es pequeña.

Adicionalmente, al sistema de control con ruedas de reacción es necesario agregarle otro sistema de control que generalmente está basado en bobinas, ya que las ruedas solo están diseñadas para compensar una cierta velocidad angular; si el sistema sobrepasa esta velocidad, es necesario un sistema alterno, el cual puede ser implementado utilizando las bobinas, además de que este sistema es utilizado en la etapa inicial cuando un satélite es puesto en órbita para contrarrestar sus altas velocidades angulares.

BIBLIOGRAFÍA

[1] J. Bouwmeester, J. Guo, "Survey of worldwide pico and nanosatellite missions, distributions and subsystem technology" *Acta Astronautica, 67,* 2010

[2] M. Tassano, P. Monzon, J. Ramos, G. De Martino and J. Pechiar, "An Expensive Attitude Determination System for the Uruguayan Cubesat, AntelSat" in IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings, 2014

[3] B. Sheela Rani, R. R. Santhosh, L. Sam Prabhu, M. Federick, V. Kumar, and S. Santhosh, "A Survey to Select Microcontroller for Sathyabama Satellite's On Board Computer Subsystem", in Recent Advances in Space Technology Services and Climate Change 2010 (RSTS & CC-2010)

[4] M. Mirghani Daffalla, A. TagElsir, A. Saeed Kajo, *"*Hardware Selection for Attitude Determination and Control Subsystem of 1U Cube Satellite*",* in International Conference on Computing, Control, Networking, Electronics and Embedded Systems Engineering (ICCNEEE), 2015

[5] C. D. Hall, "Spacecraft Attitude Dynamics and Control" *Chris Hall January 12,* 2003

[6] J. R. Cordova "Estimación y control de orientación para el nanosatélite Humsat-México", Tesis de maestría, UNAM, México D.F, 2011

[7] L.J Wiley, J.R. Wertz, "Space Mission Analysis and design",2nd ed. Microcosm Inc, Torrance Cal, 1992

[8] G. Swinerd. "How Spacecraf Fly. Spaceflight without formulae", Hampshire, Reino Unido: Praxis Publishing. 2008

[9] C. Whitcomb Crowell, "Development and Analysis of a Small Satellite Attitude Determination and Control System Testbed" Tesis de maestría, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, 2011

[10] D. Bhanderi, B. Orsted Andresen, C. Gron, R. Hviid Knudsen, C. Nielsen, K. Kjaer Sorensen, and D Taagaard, "Attitude Control System for AAUSAT-II", Institute of Electronic System, Aalborg University, 2005

[11] http://www.space.aau.dk/aausat3/index.php?n=Tech.AAUSAT3InDetails consultado el 27 de noviembre del 2016

[12] M. Lungu, R. Lungu, M. Ioan, "Automatic Control of the Satellites´ Attitude and Stored Energy using Inertial Wheels" in 1*7th International Carpathian Control Conference,* 2016

[13] J-H. Huang, T-Y. Lin, C-M. Liu and J-C Juang, "Desig and evaluation of the Attitude Control System of the PHOENIX CubeSat," in *CACS International Automatic Control Conference,* December 2013

[14] M. Tassano, P. Monzon, and J. Pechiar, “Attitude Determination and Control System of the Uruguayan CubeSat, AntelSat,” in *16th International Conference on Advanced Robotics (ICAR)*, 2013

[15] X. Xun and X.Wu, “A CubeSat Attitude Control System with Linear Piezoelectric Actuator”, in *Symposium on Piezoelectricity, Acoustic Waves, and Device Applications*, Beijin, China, 2014

[16] F. Landis Markley, J. L. Crassidis, “Fundamentals of Spacecraft Attitude Determination and contol”, Space Technology Library, Springer, 2014

[17] H. Hernández Arias, “Diseño de algoritmos de control de orientación para satélites pequeños” Tesis de maestría, UNAM, México D.F., 2013

[18] S. Sondersrod Ose, “Attitude Determination for the Norwegian student satellite nCube”, Tesis de Maestría, Department of Engineering Cibernetics, Norwegian University of Science and Technology, Trondheim, Norway, 2004

[19] R. Reyes Morales, “Sistema de orientación y estabilización para un satélite pequeño de percepción remota”, Tesis de licenciatura, UNAM, México D. F. 2012

[20] H. Curtis, “Orbital Mechanics for Engineering Students”, *Elsevier Aerospace Engineering Series*, 2005

[21] R. Cepeda Gómez “Sistema de Control Robusto, Basado en Cuaterniones, para un Satélite de Órbita Baja”, Tesis de maestría, Pontificia Universidad Javeriana, Bogotá D. C., 2010

[22] R. Reyes Morales, “Diseño de un esquema de control para la orientación en tres ejes de un micro-satélite de percepción remota con ruedas inerciales”, Tesis de Maestría, UNAM, Mexico D.F. 2015.

[23] K. Grobekatthofer, Z. Yoon, “Introduction into quaternions for spacecraft attitude representation”, Technical University of Berlin, Berlin, Germany, 2012

[24] J. Gratton, “Mecánica Tomo I: Mecánica del punto y del cuerpo rígido”, Buenos Aires, 2006

[25] K. Magne Fauske, “Attitude Stabilization of an Underactuated Rigid Spacecraft”, Department of Engineering Cybernetics, Norwegian University of Technology and Science, 2003

[26] A. K Maini, V. Agrawal, “Satellite Technology: Principles and Applications”, Wiley, Noida, India, 2011

[27] N. Sugimura, T. Kuwahara, K. Yoshida, “Attitude Determination and Control System for Nadir Pointing Using Magnetorquer and Magnetometer” in *IEEE Aerospace Conference*, 2016

[28] A. Bedford, W. Fowler, “Dinámica”, Addison Wesley Longman de México, 2000.

[29] S. K. Shrivastava, V. J. Modi, “Satellite Attitude Dynamics and Dynamics and Control in the Presence of Environmental Torques – A Brief Survey”, Journal of Guidance, Vol. 6, No 6, December 1983.

[30] C. Kaplan, “Leo Satellites: Attitude Determination and Control Components; some Linear Attitude Control Techniques” Tesis de Maestría, Middle East Technical University, 2006.

[31] S. Karatas, “Leo Satellites: Dynamic Modelling, Simulation and some Nonlinear Attitude Control Techniques”, Tesis de Maestría, Middle East Technical University, 2006

[32] J. Prado Molina, “Sistema de simulación para pruebas de algoritmos de orientación y control de satélites pequeños”, Tesis de Doctorado, UNAM, México, 2007

[33] S. I. Flores Gómez, “Simulación y pruebas de control para micro-satélites”, Tesis de Licencitura, UNAM, México D.F. 2012

ANEXOS

A

Código en Python del algoritmo de control

```
#!/usr/bin/python       #Ruta del ejecutable de python
#coding=utf-8           #Formato de codificación Unicode
import sys, getopt      # Acceso a variables usadas por el interprete
from Adafruit_MotorHAT import Adafruit_MotorHAT, Adafruit_DCMotor      #Importación de funciones de puentes H
import atexit           #Funciones de limpieza de los registros
from time import time    #Importacion de funciones de tiempo
from time import sleep   #Importación de retardos de tiempo
import math              #Importación de funciones matemáticas
from numpy import *              # Funciones para operaciones con vectores y matrices
from matplotlib.pyplot import *  # Funciones para graficar datos
from control.matlab import *     # Funciones de control
import numpy as np               # Funciones para bases de datos
import serial                    # Funciones para comunicación serial
from brujula2 import brujula_1   # Funciones para obtenner datos de la brújula
sys.path.append('.')             # Funcion para importar librerias
import RTIMU                     #Importación de funciones para operar el integrado mpu9150
import os.path                   #Funciones para manejar las rutas de los archivos
from giro_2 import gyro_d        #Funciones del giróscopo

mh = Adafruit_MotorHAT(addr=0x63)   #Definición de un objeto motor
def turnOffMotors():                #Deshabilitación de motores
        mh.getMotor(1).run(Adafruit_MotorHAT.RELEASE)
        mh.getMotor(2).run(Adafruit_MotorHAT.RELEASE)
        mh.getMotor(3).run(Adafruit_MotorHAT.RELEASE)
        mh.getMotor(4).run(Adafruit_MotorHAT.RELEASE)

# Asignación de variables a los tres motores de DC
myMotor1 = mh.getMotor(1)
myMotor2 = mh.getMotor(2)
myMotor3 = mh.getMotor(3)

# Se establece velocidad 0 (apagado) a 255 (máxima velocidad)
myMotor1.setSpeed(150)
myMotor1.run(Adafruit_MotorHAT.FORWARD);
# Encendido de motor 1
myMotor1.run(Adafruit_MotorHAT.RELEASE);
myMotor2.setSpeed(150)
myMotor2.run(Adafruit_MotorHAT.FORWARD);
# Encendido de motor 2
myMotor2.run(Adafruit_MotorHAT.RELEASE);
myMotor3.setSpeed(150)
```

```
myMotor3.run(Adafruit_MotorHAT.FORWARD);
# Encendido de motor 3
myMotor3.run(Adafruit_MotorHAT.RELEASE);

#Velocidad a 0 motor 1
myMotor1.run(Adafruit_MotorHAT.FORWARD)
myMotor1.setSpeed(0)
#Velocidad a 0 motor 2
myMotor2.run(Adafruit_MotorHAT.FORWARD)
myMotor2.setSpeed(0)
Velocidad a 0 motor 3
myMotor3.run(Adafruit_MotorHAT.FORWARD)
myMotor3.setSpeed(0)

#Inicialización de vectores para captura de datos
roll = 0
R = np.array([0.0])    #Vector para el eje x
pitch = 0
P = np.array([0.0])     #Vector para el eje y
yaw = 0
Y = np.array([0.0])    #Vector para el eje z

# Establecimiento  del valor PWM
a = 60
# Ganancia de las rectas
b = 0.6
b2 = 0.6
# Contantes de los valores de las rectas de control
g_pa = 0.0916
c_pa = 0.126
g_va = 0.0805
c_va = 0.203

while True:
   orientacion = brujula_1()     #Obtención de la orientación
   roll = orientacion[0]           #Orientación en el eje x
   pitch = orientacion[1]          #Orientación en el eje y
   yaw = orientacion[2]            #Orientación en el eje z
   #Impresion en pantalla de los valores de orientación
   print roll
   print pitch
   print yaw
   #Almacenamiento de los datos de orientación de el eje x en el documento array_roll.txt
   R = np.append(R, roll)
   np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_roll.txt', R, fmt = '%.4e')
   #Almacenamiento de los datos de orientación de el eje y en el documento array_pitch.txt
   P = np.append(P, pitch)
   np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_pitch.txt', P, fmt = '%.4e')
   #Almacenamiento de los datos de orientación de el eje z en el documento array_yaw.txt
   Y = np.append(Y, yaw)
   np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_yaw.txt', Y, fmt = '%.4e')
```

```
    #Obtención de los datos del giróscopo
    gy = gyro_d()
    #Conversión de la velocidad angular en rad a grados/segundo
    va_roll = gy[0] * (180/math.pi)
    va_pitch = gy[1] * (180/math.pi)
    va_yaw = gy[2] * (180/math.pi)
    #Impresión en pantalla de las velocidades angulares en cada uno de los ejes
    print va_roll
    print va_pitch
    print va_yaw

    #Si la orientación en cada uno de los ejes se encuentra entre (-2° y 2°), los motores se mantienen
apagados
    if (roll > -2.0) and roll < 2.0 :
        print "0 x"
        myMotor1.run(Adafruit_MotorHAT.FORWARD)
        myMotor1.setSpeed(0)
    if (pitch > -2.0) and pitch < 2.0:
        print "0 y"
        myMotor2.run(Adafruit_MotorHAT.FORWARD)
        myMotor2.setSpeed(0)
    if (yaw > -2.0) and yaw < 2.0:
        print "0 z"
        myMotor3.run(Adafruit_MotorHAT.FORWARD)
        myMotor3.setSpeed(0)

#Si la orientación del sistema es mayor a 20° es enviada una señal constante a los motores para
disminuir el error
    if yaw > 20.0:
        print "Forward! z"
        print "70"
        myMotor3.run(Adafruit_MotorHAT.FORWARD)  #Dirección de giro del motor
        myMotor3.setSpeed(70)     #Señal voltaje enviada al motor
        sleep(2)                  #Retardo del impulso
        myMotor3.setSpeed(0)      #Apagado del motor

        for i in range (1,2):

            sleep(0.1)
           #Sensado de la orientación del sistema
            orientacion = brujula_1()
            #print orientacion
            roll = orientacion[0]
            pitch = orientacion[1]
            yaw = orientacion[2]
            print roll
            print pitch
            print yaw
           #Almacenamiento de los datos de orientación
           R = np.append(R, roll)
```

```
        np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_roll.txt', R, fmt = '%.4e')
        P = np.append(P, pitch)
        np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_pitch.txt', P, fmt = '%.4e')
        Y = np.append(Y, yaw)
        np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_yaw.txt', Y, fmt = '%.4e')

    #Si el error en la orientación es menor a -20° es enviada una señal constante para disminuir el error
    if yaw < -20.0:
       print "Backward! z"
       print "70"
       myMotor3.run(Adafruit_MotorHAT.BACKWARD)  #Dirección de giro del motor
       myMotor3.setSpeed(a)                      #Señal voltaje enviada al motor
       sleep(1)                                  #Retardo del impulso
       myMotor3.setSpeed(0)                      #Apagado del motor

       for i in range (1,2):

          sleep(0.1)
         #Sensado de la orientación
          orientacion = brujula_1()
          roll = orientacion[0]
          pitch = orientacion[1]
          yaw = orientacion[2]
          print roll
          print pitch
          print yaw

         #Almacenamiento de los datos de orientación
          R = np.append(R, roll)
          np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_roll.txt', R, fmt = '%.4e')
          P = np.append(P, pitch)
          np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_pitch.txt', P, fmt = '%.4e')
          Y = np.append(Y, yaw)
          np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_yaw.txt', Y, fmt = '%.4e')

#Si el error en la orientación se encuentra entre (2° y 20°) se utiliza el control proporcional
if yaw > 2.0 and yaw < 20.0 :
       impulso_z_r = abs((g_pa * yaw + c_pa + g_va * va_yaw + c_va)*(b2))     #Ley de control

       print "Forward! z"
       myMotor3.run(Adafruit_MotorHAT.FORWARD)  #Dirección de giro del motor
       myMotor3.setSpeed(a)                     #Señal de voltaje enviada el motor
       sleep(impulso_z_r)                       #Retardo del impulso
       myMotor3.setSpeed(0)                     #Apagado de motor




for i in range (1,2):

          sleep(0.2)
```

```
        #Sensado de orientación
        orientacion = brujula_1()
        roll = orientacion[0]
        pitch = orientacion[1]
        yaw = orientacion[2]
        print roll
        print pitch
        print yaw

        #Almacenamiento de los datos de orientación
        R = np.append(R, roll)
        np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_roll.txt', R, fmt = '%.4e')
        P = np.append(P, pitch)
        np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_pitch.txt', P, fmt = '%.4e')
        Y = np.append(Y, yaw)
        np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_yaw.txt', Y, fmt = '%.4e')

    #Si el error en la orientación se encuentra entre (-2° y -20°) se utiliza el control proporcional
    if yaw < -2.0 and yaw > -20.0 :
        impulso_z_l = abs ((g_pa * yaw - c_pa +( g_va * va_yaw) - c_va)*(b))   #Ley de control

        print "Backward! z"
        myMotor3.run(Adafruit_MotorHAT.BACKWARD)   #Dirección de giro del motor
        myMotor3.setSpeed(a)                       #Señal de voltaje enviada al motor
        sleep(impulso_z_l)                         #Retardo del impulso
        myMotor3.setSpeed(0)                       #Apagado de motor

        for i in range (1,2):
            sleep(0.2)          #Sensado de la orientación
            orientacion = brujula_1()
            roll = orientacion[0]
            pitch = orientacion[1]
            yaw = orientacion[2]
            print roll
            print pitch
            print yaw

            #Almacenamiento de los datos de orientación
            R = np.append(R, roll)
            np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_roll.txt', R, fmt = '%.4e')
            P = np.append(P, pitch)
            np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_pitch.txt', P, fmt = '%.4e')
            Y = np.append(Y, yaw)
            np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_yaw.txt', Y, fmt = '%.4e')

sleep(0.05)



# Librería de la brújula ez-compass-4
```

```
import serial                #Funciones de comunicación serial
import time                  #Funciones de tiempo
from numpy import *          #Funciones para operar con vectores y matrices

def brujula_1():

    brujula = serial.Serial('/dev/ttyACM0', baudrate=9600, timeout=1)  #Definición de Puerto serial
    brujula.close()          #Se cierra Puerto serial
    time.sleep(0.1)          #Retardo
    brujula.open()           #Apertura de Puerto serial
    line = brujula.readline()   #Lectura de datos de la brújula
    print "\n"
    #Definición de variables para la extracción de datos del vector line
    r = line.find('R')
    p = line.find('P')
    t = line.find('T')
    y = line.find('C')
    f = line.find('*')
   #Extracción de los datos de orientación
    ro = line[r+1:p]
    pi = line[p+1:t]
    ya = line[y+1:f]
   #Conversión a varibles de tipo flotante
    rol = float(ro) * 1
    pit = float(pi) * 1
    yaw = float(ya) * 1
    #Adecuación de los valores leídos para la orientación en el eje z (yaw)
    if yaw > 180 :
        yaw = yaw - 360
   #Almacenamiento de los datos de orientación en un vector
    attitude = array([rol, pit, yaw])
    brujula.close()   #Se Cierra el puerto serial
    return attitude    #Se retorna el vector de orientación

# Librería del sensor MPU-9250

#coding=utf-8         #Codificación de caracteres Unicode
import sys, getopt    #Funciones del sistema
sys.path.append('.')  #Funciones para importar librerias
import RTIMU          #Funciones del integrado MPU-9150
import os.path        #Funciones para manejar las rutas de los archivos
import time           #Funciones de tiempo
import math           #Funciones matematicas

def gyro_d():
    #Inicialización de la funciones del integrado MPU-9150
    SETTINGS_FILE = "RTIMULib"
    if not os.path.exists(SETTINGS_FILE + ".ini"):
        print("Las configuaraciones no existen, seran creadas")
    s = RTIMU.Settings(SETTINGS_FILE)
    imu = RTIMU.RTIMU(s)
```

```
    if (not imu.IMUInit()):
        print("IMU Init Failed")
        sys.exit(1)
    else:
        a=1
    #Habilitación de los sensores en el integrado MPU-9150
    imu.setSlerpPower(0.02)
    imu.setGyroEnable(True)
    imu.setAccelEnable(True)
    imu.setCompassEnable(True)
    poll_interval = imu.IMUGetPollInterval()

    c=0
    while c<=1:
        if imu.IMURead():
            data = imu.getIMUData()  #Obtención de los datos del giróscopo
            gyro = data["gyro"]          #Almacenamiento de los datos de velocidad angular en la variable
gyro
            c = c+1
            time.sleep(poll_interval*1.0/1000.0)
    return gyro                          #Retorno de datos del vector gyro
```

B

Código para realizar la regresión lineal de los datos de caracterización de la plataforma de simulación.

```
import numpy as np                  #Funciones para operar con matrices y vectores
import pylab as pl                  #Funciones para graficar datos
from sklearn import datasets, linear_model   #Funciones para realizer la regression lineal de datos

#x1 = [[0.92],[1.6],[4.98],[5.7],[7.125],[8.55]]
#y1 = [0.2,0.3,0.5,0.7,0.8,0.9]
x1 = [[0.92],[1],[4.98],[5.7],[7.125],[8.55]]
y1 = [0.2,0.3,0.5,0.7,0.8,0.9]
#x1 = [[1.6],[4.17],[8],[13.8],[15]]
#y1 = [0.1,0.3,0.5,0.8,0.9]
#x1 = [[0.693],[1.912],[7.21],[14.72],[16]]
#y1 = [0.1,0.3,0.5,0.8,0.9]

regr = linear_model.LinearRegression()    #Regresion lineal
regr.fit(x1,y1)                           #Regresion para los vectores x1 y y1
print('Coefficients: \n', regr.coef_, regr.intercept_)  #Impresion de pendiente y ordenada al origen de
la recta obtenidad en la regression lineal
#Configuración de la impresión en pantalla de la gráfica
pl.figure(1)
pl.plot(x1, regr.predict(x1), color='black', linewidth=3)
pl.scatter(x1,y1, s=40, marker='o', color='k')
pl.xlabel('Velocidad angular [grados/segundo]')
pl.ylabel('Impulso angular')
pl.grid()
#pl.xlim(.0, 5.0)
#pl.ylim(.0, 5.0)
#Impresión de la gráfíca.
pl.show()
```

C

Código para la caracterización dinámica de la plataforma

```
#!/usr/bin/python    #Ruta del ejecutable de python
#coding=utf-8        #Codificación de caracteres Unicode
from Adafruit_MotorHAT import Adafruit_MotorHAT, Adafruit_DCMotor #Funciones para el Puente H
import atexit          #Funciones para limpiar los registros
from time import time    #Funciones de tiempo
from time import sleep   #Funciones de retardo
import math              #Funciones matemáticas
from numpy import *     #Funciones para operaciones con vectores y matrices
from matplotlib.pyplot import *   #Funciones para graficar
from control.matlab import *      #Funciones de control
import numpy as np                #Librería para bases de datos
import serial                     #Funciones de comunicación serial
from brujula2 import brujula_1    #Librería para operar con la brújula
mh = Adafruit_MotorHAT(addr=0x63)  # Creacion de un objeto para manejar los puentes H a través de I2C

def turnOffMotors():  #Deshabilitación de motores
      mh.getMotor(1).run(Adafruit_MotorHAT.RELEASE)
      mh.getMotor(2).run(Adafruit_MotorHAT.RELEASE)
      mh.getMotor(3).run(Adafruit_MotorHAT.RELEASE)
      mh.getMotor(4).run(Adafruit_MotorHAT.RELEASE)
#Asignacion de variables a los motores
myMotor1 = mh.getMotor(1)
myMotor2 = mh.getMotor(2)
myMotor3 = mh.getMotor(3)
#Se establece velocidad en el motor 0 (apagado), 255 (velocidad maxima)
myMotor1.setSpeed(0)
myMotor1.run(Adafruit_MotorHAT.FORWARD);
myMotor1.run(Adafruit_MotorHAT.RELEASE);
myMotor2.setSpeed(0)
myMotor2.run(Adafruit_MotorHAT.FORWARD);
myMotor2.run(Adafruit_MotorHAT.RELEASE);
myMotor3.setSpeed(0)
myMotor3.run(Adafruit_MotorHAT.FORWARD);
myMotor3.run(Adafruit_MotorHAT.RELEASE);

#Inicialización de vectores para  almacenar datos
roll = 0
R = np.array([0.0])   #Vector para el eje x
pitch = 0
P = np.array([0.0])     #Vector para el eje y
yaw = 0
Y = np.array([0.0])    #Vector para el eje z

j=1      #Indice para contar los ciclos de muestreo
```

```
tempo = 0   #Variable para almacenar el tiempo transcurrido
while j <=5 :

    tiempo_inicial = time()      #Inicio de la cuenta de tiempo
    orientacion = brujula_1()   #Obtención de los datos de orientación
    roll = orientacion[0]          #Orientación en eje x
    pitch = orientacion[1]        #Orientación en eje y
    yaw = orientacion[2]          #Orientación en eje z

    R = np.append(R, roll)
    # Almacenamiento de los datos de orientación del eje x en el documento array_roll.txt
    np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_roll.txt', R, fmt = '%.4e')
    P = np.append(P, pitch)
    # Almacenamiento de los datos de orientación del eje y en el documento array_pitch.txt
    np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_pitch.txt', P, fmt = '%.4e')
    Y = np.append(Y, yaw)
    # Almacenamiento de los datos de orientación del eje z en el documento array_yaw.txt
    np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_yaw.txt', Y, fmt = '%.4e')

    #Encendido de motor 3
    myMotor3.run(Adafruit_MotorHAT.FORWARD)
    myMotor3.setSpeed(60)
    sleep(1.6)   #Retardo
    myMotor3.setSpeed(0)   #Apagado de motor
    for i in range (1,5):        #Muestreo de 5 ciclos
        sleep(0.5)
        orientacion = brujula_1()  #Obtención de los datos de orientación
        roll = orientacion[0]
        pitch = orientacion[1]
        yaw = orientacion[2]

        R = np.append(R, roll)
        # Almacenamiento de los datos de orientación del eje x en el documento array_roll.txt
        np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_roll.txt', R, fmt = '%.4e')
        P = np.append(P, pitch)
        # Almacenamiento de los datos de orientación del eje y en el documento array_roll.txt
        np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_pitch.txt', P, fmt = '%.4e')
        Y = np.append(Y, yaw)
        # Almacenamiento de los datos de orientación del eje z en el documento array_roll.txt
        np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_yaw.txt', Y, fmt = '%.4e')
    tiempo_final = time()  #Medición de tiempo final
    time_transcurrido = tiempo_final - tiempo_inicial  #Obtención del tiempo trasncurrido
    print time_transcurrido
    tempo += time_transcurrido    #Acumulador del tiempo transcurrido
    j += 1
    print j
    sleep(0.05)
while True:
    print tempo
    print "prueba terminada"
    sleep(2)
```

D

Control de la orientación con bobinas y ruedas de reacción

```
#!/usr/bin/python       #Ruta del ejecutable de python
#coding=utf-8           #Codificación de caracteres Unicode
import sys, getopt      #Acceso a variables utilizadas por el interprete
import math             #Importación de funciones matemáticas
from numpy import *           #Funciones para operar con vectores y matrices
from matplotlib.pyplot import *  #Funciones para graficar datos
from control.matlab import *       #Funciones de control
import serial                      #Funciones de comunicación serial
from brujula import brujula_1     #Librería para operar con la brújula
from Adafruit_MotorHAT import Adafruit_MotorHAT, Adafruit_DCMotor  #Funciones para operar
con los puentes H
import atexit     #Funciones para limpiar registros
from time import time   #Funciones de tiempo
from time import sleep  #Funciones de retardo
mh = Adafruit_MotorHAT(addr=0x63)       #Definición de un objeto motor mh
mh_b = Adafruit_MotorHAT(addr=0x64)   #Definición de un objeto motor mh_b
# Deshabilitación de motores
def turnOffMotors():
        mh.getMotor(1).run(Adafruit_MotorHAT.RELEASE)
        mh.getMotor(2).run(Adafruit_MotorHAT.RELEASE)
        mh.getMotor(3).run(Adafruit_MotorHAT.RELEASE)
        mh.getMotor(4).run(Adafruit_MotorHAT.RELEASE)
        mh_b.getMotor(1).run(Adafruit_MotorHAT.RELEASE)
        mh_b.getMotor(2).run(Adafruit_MotorHAT.RELEASE)
        mh_b.getMotor(3).run(Adafruit_MotorHAT.RELEASE)
        mh_b.getMotor(4).run(Adafruit_MotorHAT.RELEASE)
atexit.register(turnOffMotors)    #Apagado de motores
#Prueba de los motores de dc
myMotor1 = mh.getMotor(1)
myMotor2 = mh.getMotor(2)
myMotor3 = mh.getMotor(3)
myMotor1_b = mh_b.getMotor(1)
myMotor2_b = mh_b.getMotor(2)
myMotor3_b = mh_b.getMotor(3)
myMotor1.setSpeed(150)
myMotor1.run(Adafruit_MotorHAT.FORWARD);
myMotor1.run(Adafruit_MotorHAT.RELEASE);
myMotor1_b.setSpeed(150)
myMotor1_b.run(Adafruit_MotorHAT.FORWARD);
myMotor1_b.run(Adafruit_MotorHAT.RELEASE);
myMotor2.setSpeed(150)
myMotor2.run(Adafruit_MotorHAT.FORWARD);
myMotor2.run(Adafruit_MotorHAT.RELEASE);
myMotor2_b.setSpeed(150)
myMotor2_b.run(Adafruit_MotorHAT.FORWARD);
myMotor2_b.run(Adafruit_MotorHAT.RELEASE);
```

```
myMotor3.setSpeed(150)
myMotor3.run(Adafruit_MotorHAT.FORWARD);
myMotor3.run(Adafruit_MotorHAT.RELEASE);
myMotor3_b.setSpeed(150)
myMotor3_b.run(Adafruit_MotorHAT.FORWARD);
myMotor3_b.run(Adafruit_MotorHAT.RELEASE);

#Cálculo de la velocidad angular utilizando los datos de la brujula
orientacion = brujula_1()   #Obtención de la orientación
roll = orientacion[0]          #Orientación en eje x
pitch = orientacion[1]         #Orientación en eje y
yaw = orientacion[2]           #Orientación en eje z
sleep(0.2)                     #Retardo de tiempo
orientacion = brujula_1()    #Obtención de la orientación
roll2 = orientacion[0]         #Orientación en eje x
pitch2 = orientacion[1]        #Orientación en eje y
yaw2 = orientacion[2]          #Orientación en eje z
#Obtención de la velocidad angular
va_roll = (roll2 - roll)/(0.2)
va_pitch = (pitch2 - pitch)/(0.2)
va_yaw = (yaw2 - yaw)/(0.2)

#Definición de los vectores para almacenar los datos de orientación
roll = 0
R = np.array([0.0]) #Vector para eje x
pitch = 0
P = np.array([0.0]) #Vector para eje y
yaw = 0
Y = np.array([0.0]) #Vector para eje z

# Establecimiento del valor PWM
a = 60
# Ganancia de las rectas
b = 0.3
b2 = 0.3
# Contantes de los valores de las rectas de control
g_pa = 0.0916
c_pa = 0.126
g_va = 0.0805
c_va = 0.203

while True:
    #Si la velocidad de la plataforma es mayor a 5°/s  o menor a -5°/s entonces se activa el control
    #con bobinas
    if va_roll > 5 or va_roll < -5 or va_pitch > 5 or va_pitch < -5 or va_yaw > 5 or va_yaw < -5:
        orientacion = brujula_1()  #Obtención del vector de orientación
        roll = orientacion[0]          #Orientación en eje x
        pitch = orientacion[1]         #Orientación en eje y
        yaw = orientacion[2]           #Orientación en eje z
```

```
R = np.append(R, roll)
# Almacenamiento de los datos de orientación del eje x en el documento array_roll.txt
np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_roll.txt', R, fmt = '%.4e')
P = np.append(P, pitch)
# Almacenamiento de los datos de orientación del eje y en el documento array_pitch.txt
np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_pitch.txt', P, fmt = '%.4e')
Y = np.append(Y, yaw)
# Almacenamiento de los datos de orientación del eje z en el documento array_yaw.txt
np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_yaw.txt', Y, fmt = '%.4e')
sleep(0.2)        #Retardo
orientacion = brujula_1()   #Obtención de los datos de orientación
roll2 = orientacion[0]        #Orientación en eje x
pitch2 = orientacion[1]       #Orientación en eje y
yaw2 = orientacion[2]         #Orientación en eje z

R = np.append(R, roll)
# Almacenamiento de los datos de orientación del eje x en el documento array_roll.txt
np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_roll.txt', R, fmt = '%.4e')
P = np.append(P, pitch)
# Almacenamiento de los datos de orientación del eje y en el documento array_pitch.txt
np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_pitch.txt', P, fmt = '%.4e')
Y = np.append(Y, yaw)
# Almacenamiento de los datos de orientación del eje z en el documento array_yaw.txt
np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_yaw.txt', Y, fmt = '%.4e')
#Obtención de la velocidad angular
va_roll = (roll2 - roll)/(0.2)
va_pitch = (pitch2 - pitch)/(0.2)
va_yaw = (yaw2 - yaw)/(0.2)

if va_roll > 5 :  #Si la velocidad es mayor a 5°/s se realiza el control con bobinas
myMotor1_b.run(Adafruit_MotorHAT.BACKWARD)   #encendido de bobina 1
myMotor1_b.setSpeed(70)    #Se establece un voltaje utilizando PWM
sleep(1)                    #Tiempo que se mantiene encendida la bobina
myMotor1_b.setSpeed(0)      #Se apaga la bobina

for i in range (1,2):  #Se realizan dos ciclos para evaluar el cambio de orientación
    sleep(0.2)          #Retardo entre cada ciclo
    orientacion = brujula_1()   #Obtención del vector de orientación
    roll = orientacion[0]         #Orientación en el eje x
    pitch = orientacion[1]        #Orientación en el eje y
    yaw = orientacion[2]          #Orientación en el eje z

    R = np.append(R, roll)
    # Almacenamiento de los datos de orientación del eje x en el documento array_roll.txt
    np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_roll.txt', R, fmt = '%.4e')
    P = np.append(P, pitch)
    # Almacenamiento de los datos de orientación del eje y en el documento array_pitch.txt
    np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_pitch.txt', P, fmt = '%.4e')
    Y = np.append(Y, yaw)
    # Almacenamiento de los datos de orientación del eje z en el documento array_yaw.txt
    np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_yaw.txt', Y, fmt = '%.4e')
```

```
    if va_roll < -5 :  #Si la velocidad es menor a -5°/s se realiza el control con bobinas
    myMotor1_b.run(Adafruit_MotorHAT.FORWARD) #Encendido de bobina 1
    myMotor1_b.setSpeed(70)  #Se establece un voltaje utilizando PWM
    sleep(1)                 #Tiempo que se mantiene encendidad la bobina
    myMotor1_b.setSpeed(0)   #Se apaga la bobina
    for i in range (1,2):    #Se realizan dos ciclos para evaluar el cambio de orientación
       sleep(0.2)            #Retardo entre cada ciclo
       orientacion = brujula_1()  #Obtención del vector de orientación
       roll = orientacion[0]      #Orientación en eje x
       pitch = orientacion[1]     #Orientación en eje y
       yaw = orientacion[2]       #Orientación en eje z

       R = np.append(R, roll)
       # Almacenamiento de los datos de orientación del eje x en el documento array_roll.txt
       np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_roll.txt', R, fmt = '%.4e')
       P = np.append(P, pitch)
       # Almacenamiento de los datos de orientación del eje y en el documento array_pitch.txt
       np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_pitch.txt', P, fmt = '%.4e')
       Y = np.append(Y, yaw)
       # Almacenamiento de los datos de orientación del eje z en el documento array_yaw.txt
       np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_yaw.txt', Y, fmt = '%.4e')

 #Si la velocidad es menor a los +-5°/s entonces se pone en marcha el control con ruedas de
 # reacción
else:
    orientacion = brujula_1()  #Obtención del vector de orientación
    roll = orientacion[0]      #Orientación en eje x
    pitch = orientacion[1]     #Orientación en eje y
    yaw = orientacion[2]       #Orientación en eje z
    R = np.append(R, roll)
    # Almacenamiento de los datos de orientación del eje x en el documento array_roll.txt
    np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_roll.txt', R, fmt = '%.4e')
    P = np.append(P, pitch)
    # Almacenamiento de los datos de orientación del eje y en el documento array_pitch.txt
    np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_pitch.txt', P, fmt = '%.4e')
    Y = np.append(Y, yaw)
    # Almacenamiento de los datos de orientación del eje z en el documento array_yaw.txt
    np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_yaw.txt', Y, fmt = '%.4e')
    sleep(0.2)             #Intervalo esperado para realizar la segunda medición de orientación
    orientacion = brujula_1()  #Obtención del vector de orientación
    roll2 = orientacion[0]     #Orientación en eje x
    pitch2 = orientacion[1]    #Orientación en eje y
    yaw2 = orientacion[2]      #Orientación en eje z
    R = np.append(R, roll)
    # Almacenamiento de los datos de orientación del eje x en el documento array_roll.txt
    np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_roll.txt', R, fmt = '%.4e')
    P = np.append(P, pitch)
    # Almacenamiento de los datos de orientación del eje y en el documento array_pitch.txt
    np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_pitch.txt', P, fmt = '%.4e')
    Y = np.append(Y, yaw)
```

```
        # Almacenamiento de los datos de orientación del eje z en el documento array_yaw.txt
        np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_yaw.txt', Y, fmt = '%.4e')

       #Obtención de la velocidad angular
        va_roll = (roll2 - roll)/(0.2)
        va_pitch = (pitch2 - pitch)/(0.2)
        va_yaw = (yaw2 - yaw)/(0.2)

        #Si la orientación del sistema se encuentra en un rango de (-2, 2) los motores se mantienen apagados
        if (roll > -2.0) and roll < 2.0 :
           print "0 x"
           myMotor1.run(Adafruit_MotorHAT.FORWARD)
           myMotor1.setSpeed(0)
        if (pitch > -2.0) and pitch < 2.0 :
           print "0 y"
           myMotor2.run(Adafruit_MotorHAT.FORWARD)
           myMotor2.setSpeed(0)
        if (yaw > -2.0) and yaw < 2.0 :
           print "0 z"
           myMotor3.run(Adafruit_MotorHAT.FORWARD)
           myMotor3.setSpeed(0)
        #Si la orientación del sistema es mayor a 20° es enviada una señal constante a los motores para disminuir el error
        if yaw > 20.0 :
           print "Forward! z"
           print "70"
           myMotor1.run(Adafruit_MotorHAT.BACKWARD)   #Dirección de giro del motor
           myMotor1.setSpeed(70)          #Señal de voltaje enviada al motor
           sleep(2)                       #Retardo del impulso
           myMotor1.setSpeed(0)           #Apagado del motor
           for i in range (1,2):
              sleep(0.2)
              orientacion = brujula_1()       #Obtención del vector de orientación
              roll = orientacion[0]           #Orientación en eje x
              pitch = orientacion[1]          #Orientación en eje y
              yaw = orientacion[2]            #Orientación en eje z

              R = np.append(R, roll)
              # Almacenamiento de los datos de orientación del eje x en el documento array_roll.txt
              np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_roll.txt', R, fmt = '%.4e')
              P = np.append(P, pitch)
              # Almacenamiento de los datos de orientación del eje y en el documento array_pitch.txt
              np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_pitch.txt', P, fmt = '%.4e')
              Y = np.append(Y, yaw)
              # Almacenamiento de los datos de orientación del eje z en el documento array_yaw.txt
              np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_yaw.txt', Y, fmt = '%.4e')
           sleep(0.1)
```

```
        #Si la orientación del sistema es menor a -20° es enviada una señal constante a los motores para disminuir el error
        if yaw < -20.0 :
           print "Backward! z"
           print "70"
           myMotor1.run(Adafruit_MotorHAT.FORWARD)  #Dirección de giro del motor
           myMotor1.setSpeed(70)                    #Señal de voltaje enviada al motor
           sleep(2)                                 #Retardo del impulso
           myMotor1.setSpeed(0)                     #Apagado de motor
           for i in range (1,2):
              sleep(0.2)
              orientacion = brujula_1()             #Obtención  del vector de orientación
              roll = orientacion[0]                 #Orientación en eje x
              pitch = orientacion[1]                #Orientación en eje y
              yaw = orientacion[2]                  #Orientación en eje z
              R = np.append(R, roll)
             # Almacenamiento de los datos de orientación del eje x en el documento array_roll.txt
              np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_roll.txt', R, fmt = '%.4e')
              P = np.append(P, pitch)
             # Almacenamiento de los datos de orientación del eje y en el documento array_pitch.txt
              np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_pitch.txt', P, fmt = '%.4e')
              Y = np.append(Y, yaw)
              # Almacenamiento de los datos de orientación del eje z en el documento array_yaw.txt
              np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_yaw.txt', Y, fmt = '%.4e')
           sleep(0.1)
     #Si el error en la orientación se encuentra entre (2° y 20°) se utiliza el control proporcional
        if yaw > 2.0 and yaw < 20.0 :
           delay_z_r = abs((g_pa * yaw + c_pa + g_va * va_yaw + c_va)*(b2))     #Ley de control
           print "Forward! z"
           myMotor1.run(Adafruit_MotorHAT.BACKWARD)  #Dirección de giro del motor
           myMotor1.setSpeed(a)                      #Señal de voltaje enviada al motor
           sleep(delay_z_r)                          #Retardo del impulso
           myMotor1.setSpeed(0)                      #Apagado del motor
           for i in range (1,2):
              sleep(0.2)
              orientacion = brujula_1()   #Obtención del vector de orientación
              roll = orientacion[0]       #Orientación en eje x
              pitch = orientacion[1]      #Orientación en eje y
              yaw = orientacion[2]        #Orientación en eje z
              R = np.append(R, roll)
              # Almacenamiento de los datos de orientación del eje x en el documento array_roll.txt
              np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_roll.txt', R, fmt = '%.4e')
              P = np.append(P, pitch)
              # Almacenamiento de los datos de orientación del eje x en el documento array_picth.txt
              np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_pitch.txt', P, fmt = '%.4e')
              Y = np.append(Y, yaw)
              # Almacenamiento de los datos de orientación del eje x en el documento array_yaw.txt
              np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_yaw.txt', Y, fmt = '%.4e')
           sleep(0.1)
```

```
#Si el error en la orientación se encuentra entre (-2° y -20°) se utiliza el control proporcional
        if yaw < -2.0 and yaw > -20.0 :
           delay_z_l = abs ((g_pa * yaw - c_pa + (g_va * va_yaw) - c_va)*(b))   #Ley de control
           print "Backward! z"
           myMotor1.run(Adafruit_MotorHAT.FORWARD) #Dirección de giro del motor
           myMotor1.setSpeed(a)                    #Señal de voltaje enviada al motor
           sleep(delay_z_l)                        #Retardo del impulso
           myMotor1.setSpeed(0)                    #Apagado de motor
           for i in range (1,2):
              sleep(0.2)
              orientacion = brujula_1()        #Obtención de la orientación
              roll = orientacion[0]            #Orientación en eje x
              pitch = orientacion[1]           #Orinetación en eje y
              yaw = orientacion[2]             #Orientación en eje z
              R = np.append(R, roll)
              # Almacenamiento de los datos de orientación del eje x en el documento array_roll.txt
              np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_roll.txt', R, fmt = '%.4e')
              P = np.append(P, pitch)
              # Almacenamiento de los datos de orientación del eje x en el documento array_pitch.txt
              np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_pitch.txt', P, fmt = '%.4e')
              Y = np.append(Y, yaw)
              # Almacenamiento de los datos de orientación del eje x en el documento array_yaw.txt
              np.savetxt('/home/pi/Desktop/test_control/p_brujula/array_yaw.txt', Y, fmt = '%.4e')
           sleep(0.1)
        sleep(0.05)
```

E

Implementación del LQR en python

```
from numpy import *                #Funciones para operar con vectores y matrices
from matplotlib.pyplot import *   # Funciones para realizar gráficas
from control.matlab import *       #Librería de funciones de control

w = 0                 #Velocidad angular del satélite
Ixx = 0.165           #Momento de inercia principal en el eje x
Iyy = 0.1785          #Momento de inercia principal en el eje y
Izz = 0.1785          #Momento de inercia principal en el eje z

kx = (Iyy-Izz)/Ixx    #Constante kx
ky = (Ixx-Izz)/Iyy    #Constante ky
kz = (Iyy-Ixx)/Izz    #Constante kz

# Definición de la matriz A
A = matrix(
   [[ 0,                  1,            0,          0,          0,           0],
    [ -4*kx*(w**2),   0,            0,          0,          0,       (1-kx)*w],
    [ 0,                  0,            0,          1,          0,           0],
    [ 0,                  0,   -3*ky*(w**2),   0,          0,           0],
    [ 0,                  0,            0,          0,          0,           1],
    [ 0,            -(1-kz)*w,       0,          0,    -kz*(w**2),       0 ]])

# Definición de la matriz B
B = matrix([[0,                 0,        0],
            [1/(2*Ixx),         0,        0],
                  [0,           0,        0],
                  [0,       1/(2*Iyy),    0],
                  [0,           0,        0],
                  [0,           0,    1/(2*Izz)]])

# Definición de la matriz C
C = matrix([[1, 0, 0, 0, 0, 0],
                [0, 0, 1, 0, 0, 0],
                [0, 0, 0, 0, 1, 0]])

Q = diag([1, 0, 1, 0, 1, 0])*0.007;      #Matriz diagonal Q
R = diag([1, 1, 1])*4500;                 #Matriz diagonal R
(K, X, E) = lqr(A, B, Q, R); K1a = matrix(K);     #Obtención de la matriz de ganancias lqr
print K1a
```

F

Sensores

MPU-9150

El mpu-9150 es un dispositivo que combina un giróscopo MEMS de tres ejes, un acelerómetro MEMS de tres ejes, un magnetómetro MEMS de tres ejes y un procesador digital de movimiento (DMP), que es un motor acelerador de hardware.

El MPU9150 combina la aceleración el movimiento rotacional y la información de rumbo en un flujo de datos simple para alguna aplicación.

El MPU9150 tiene tres convertidores analógico-digitales de 16 bits para digitalizar las salidas del giróscopo, tres convertidores analógico-digitales de 16 bits para digitalizar las salidas del acelerómetro y tres convertidores analógico-digitales de 13 bits para digitalizar las salidas del magnetómetro. El giróscopo puede medir desde ±250 hasta ±2000°/s (0.0305°/s). El acelerómetro puede medir desde ±2g hasta ±16g y el magnetómetro puede medir hasta ±1200 μT.

La comunicación con todos los registros del dispositivo es realizada utilizando I2C a 400 kHz. Soporta vibraciones de hasta 10,000g y tiene programado un filtro paso bajas paro el giróscopo, el acelerómetro y el magnetómetro y tiene un sensor de temperatura.

Giróscopo.

Cuando el giróscopo es rotado sobre uno de sus ejes sensibles, el efecto Coriolis causa una vibración que es convertida a un voltaje proporcional a la velocidad angular. Este voltaje es digitalizado mediante un convertidor analógico-digital de 16 bits. Se pueden obtener desde 3.9 hasta 8000 muestras por segundo, también posee un filtro paso bajo programable con un amplio rango de frecuencias de corte.

Figura E-1 MPU-9150

5.1.2 EZ-compass-4

EZ-compass-4 es un magnetómetro de estado sólido de bajo costo. Posee filtros IIR para filtrar el ruido gris y permitir una rápida transferencia de datos por el puerto serial. La salida provee información continuamente sobre la orientación, el campo, la inclinación en dos ejes y la temperatura sobre el puerto serial USB o el TTL RS-232. El azimut es generado con el magnetómetro de tres ejes con una resolución de 12 bits (0.08°). La inclinación es disponible con una resolución de 12 bits hasta 180°. Provee hasta 10 datos por segundo.

Figura E-2 Magnetómetro