**DISEÑO E IMPLEMENTACIÓN DE UN ALGORITMO PROPORCIONAL-INTEGRAL-DERIVATIVO PARA LA ESTABILIZACIÓN DE UN CUADRICÓPTERO**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tutora Revisor

Ing. Evelenir Barreto

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tesista Tesista

Luis Vicens Yoshua Nava

Capítulo I – Problema

## I.1 – Planteamiento del problema

En tiempos recientes, con el desarrollo de las telecomunicaciones y la microelectrónica, existe una tendencia hacia el desarrollo de vehículos aéreos no tripulados, ya sea manejados a distancia o autónomos. En particular, se ha dado especial atención al desarrollo de multi-rotores, ya que estos brindan una gran maniobrabilidad y precisión durante el vuelo, características muy útiles para tareas de reconocimiento, vigilancia y exploración. Entre los multi-rotores, el que se atrae mayor interés en la actualidad, y sobre el cual se trabajará en el siguiente Trabajo Especial de Grado, es el cuadricóptero: un multi-rotor propulsado por cuatro motores posicionados en forma de cruz.

Muchas ideas sobre desarrollo de cuadricópteros vienen de épocas anteriores al descubrimiento del transistor, y sólo han podido llevarse a cabo en tiempos recientes, debido al avance de los microprocesadores y las baterías químicas. En principio sólo organizaciones militares y de investigación podían participar en el desarrollo de los cuadricópteros, por el alto costo de la microelectrónica en sus primeros años. Pero con el paso del tiempo, el desarrollo de cuadricópteros se ha convertido en un proyecto asequible entre aficionados a la electrónica e investigadores su construcción y estudio, ya que estos pueden acudir a un mercado muy competitivo de piezas y componentes de construcción de estos vehículos, y existe una gran cantidad de información relacionada en la Internet. Incluso, hay importantes proyectos de larga difusión, como el caso de Arduino o Raspberry Pi, que pueden brindar al usuario común la oportunidad de aproximarse al desarrollo de cuadricópteros facilitando el diseño y construcción de circuitos electrónicos, y la programación de los mismos.

Para usuarios novatos y con el prospecto de realizar proyectos de bajo coste, se han desarrollado trabajos como **[Nadales 2009]** y **[Burkamshaw 2010],** que han planteado el uso de diversas plataformas de hardware y software para la construcción de cuadricópteros de bajo coste. Pero, no se ha logrado la estandarización de los sistemas de control de estabilidad y altura para dichas plataformas.

Se propone desarrollar en el siguiente Trabajo Especial de Grado un algoritmo de control proporcional-integral-derivativo que permita regular la posición angular y altura de un cuadricóptero en vuelo.

## I.2 – Objetivos

### I.2.1 - Objetivo General:

Diseñar e implementar un algoritmo Proporcional-Integral-Derivativo que permita estabilizar la posición angular y altura de un cuadricóptero desarrollado sobre la plataforma Arduino.

### I.2.2 - Objetivos Específicos:

* Diseñar e implementar un cuadricóptero con una unidad de control basada en Arduino.
* Diseñar e implementar una interfaz de comunicación inalámbrica entre el cuadricóptero y una computadora para tareas de encendido, apagado y movimientos simples en tres dimensiones.
* Diseñar e implementar una interfaz de comunicación para la obtención y análisis de datos por telemetría.
* Diseñar e implementar un algoritmo Proporcional-Integral-Derivativo que permita la estabilización del cuadricóptero.
* Diseñar e implementar una plataforma de pruebas en tiempo real.

## I.3 - Alcance

El siguiente Trabajo Especial de Grado tiene como alcance el desarrollo de un algoritmo Proporcional-Integral-Derivativo para la estabilización angular y de altura de un cuadricóptero. El cuadricóptero en cuestión poseerá una unidad de control basada en la plataforma Arduino, y, para objeto de pruebas, podrá establecer comunicación con un agente externo que permitirá realizar las siguientes acciones:

* Encendido y apagado del cuadricóptero de forma remota.
* Control remoto de los movimientos del cuadricóptero desde la computadora. Se implementarán una serie de comandos de control los cuales permitirán mover el cuadricóptero hacia arriba, abajo, adelante, atrás, izquierda, y derecha; sin poder combinar entre sí estos movimientos.
* Obtención y visualización de datos de los sensores de posición angular y altura para evaluar la eficiencia del algoritmo y la estabilidad del robot.

## I.4 - Limitaciones

- La plataforma a utilizar para el manejo del cuadricóptero será un microcontrolador Arduino Nano 3.0, que trabaja a una frecuencia máxima de 16MHz.

- El sistema utilizará una batería de 1350mAh que servirá para alimentar a los motores, y durará entre 10 y 15 minutos de uso continuo.

- El sensor de ultrasonido HC-SR04 que se utilizará para medir la distancia respecto al suelo tendrá un rango máximo de cuatro metros, por lo cual, al realizar el despegue de forma autónoma, sólo se podría alcanzar dicha distancia máxima.

- El cuadricóptero durante el vuelo no podrá detectar objetos a su alrededor ni evadirlos, por lo cual el ambiente de pruebas debe estar totalmente despejado.

- Sólo se programará al cuadricóptero para realizar seis movimientos simples guiados por el usuario: adelante, atrás, izquierda, derecha, ascenso y descenso.

- Las pruebas en tiempo real serán realizadas en un ambiente a puerta cerrada.

## I.5 - Justificación

Considerando la situación económica actual de la nación, se considera que el presente Trabajo Especial de Grado representa un aporte al desarrollo de cuadricópteros de bajo coste**,** que pueden ser útiles para propósitos de investigación, educativos, y para el desarrollo de tecnología en el país. En particular, el presente Trabajo Especial de Grado busca complementar y mejorar la plataforma para cuadricópteros de bajo coste basada en Arduino desarrollada en **[Nadales 2009]**, mediante la implementación de los sistemas de control de posición angular y altura, y la obtención de datos por telemetría.

CAPÍTULO II – Marco Referencial

## Cuadricóptero

## Motor de corriente continua

## Batería de Polimero de Litio (LiPo)

## Unidad de Medida Inercial

Una unidad de medida inercial o IMU es un componente electrónico basado en sensores de aceleración y velocidad angular (acelerómetros y giróscopos respectivamente) la cual reporta el movimiento y orientación (Figura 1) que sufre dicha unidad. Es el componente principal de sistemas de guía inercial usados en vehículos aéreos, espaciales, marinos y aplicaciones robóticas.



Figura 1Figura 1. Orientación proporcionada por una IMU.

Fuente: [Bonastre 2010]

**Componentes de una IMU**

Cualquier unidad de medida inercial está compuesta como mínimo por un acelerómetro y un giróscopo para captar una aceleración y una velocidad angular en concreto. Generalmente, es interesante que las IMUs capten la aceleración y la velocidad angular en los tres ejes de coordenadas para conocer el movimiento exacto del componente.

1. **Acelerómetro**

Instrumento capaz de medir aceleración en uno, dos o tres ejes. Existen varios tipos de acelerómetros, dependiendo de su fabricación y funcionamiento. Las IMUs incorporan acelerómetros integrados en silicio, utilizando la tecnología llamada MEMS6, debido a la necesidad de reducir el tamaño total de la unidad. La mayoría de éstos son capacitivos, y calculan la aceleración mediante el voltaje obtenido entre dos placas una de las cuales varía su posición dependiendo del movimiento del acelerómetro. Se caracterizan por ser muy precisos en situaciones estables y tener un gran error en situaciones vibratorias o movimientos muy inestables. **[Bonastre 2010]**

1. **Giróscopo**

Dispositivo que mide la orientación, basándose en los principios de la conservación del momento angular. Las unidades de medida inercial utilizan giróscopos MEMS, es decir, integrados y de tamaño reducido. La salida de dicho sensor es un voltaje, la variación del cual nos indica en grados por segundo (V/º/s) la velocidad angular sufrida por el sensor. Se caracterizan por tener un error constante y lineal llamado bias el cual debemos tener en cuenta. **[Bonastre 2010]**

1. **Magnetómetro**

Estos dispositivos miden la fuerza i/o dirección de los campos magnéticos que los afectan respecto el campo magnético terrestre. Aunque cabe la posibilidad de que se vean afectados por variación de otros campos magnéticos en algunas zonas.**[Bonastre 2010]**

## Sensor de Ultrasonido

## Filtro complementario

## XBee

## Arduino

Arduino es un proyecto de desarrollo de tarjetas controladoras de hardware libre, de bajo costo y fácil programación, con el fin de acelerar el proceso de protipado y desarrollo de proyectos, y apoyar la educación en electrónica. Consta de una placa con entradas analógicas y digitales, y salidas digitales, y de un entorno integrado de desarrollo que se apoya sobre el lenguaje de programación Processing. Las primeras tarjetas Arduino utilizaban el microcontrolador ATMEL ATmega328, un chip de bajo costo y amplias capacidades de memoria y manejo de entradas y salidas por medio de sus puertos.Conforme ha ido avanzando el tiempo, se han desarrollado tarjetas Arduino con mejores procesadores, como lo son los modelos Arduino Mega2560 (8 bits), Arduino Tre (32 bits, ARM) y Arduino Galileo (32 bits, x86).  **[Banzi 2011]**



Ilustración 1: Imagen cara superior Arduino Nano 3.0

Fuente: <http://arduino.cc/es/Main/ArduinoBoardNano>

La placa Arduino Nano 3.0, la misma que se utilizará para el desarrollo de este Trabajo Especial de Grado, tiene las siguientes características:

|  |  |
| --- | --- |
| **Característica** | **Descripción** |
| Microcontrolador | Atmel ATmega328 |
| Frecuencia | 16 MHz |
| Memoria SRAM | 2 KB |
| EEPROM | 1 KB |
| Memoria Flash | 32 KB |
| Entradas analógicas | 8 |
| Entradas/salidas digitales (De los cuales 6 proveen PWM) | 14 |
| Interrupciones | 2 externas, |
| Protocolos de comunicación | USART, , SPI, AREF |
| Tensión de operación | 5V |
| Tensión de entrada (recomendada) | 7-12 V |
| Tensión de entrada (límites) | 6-20 V |
| Corriente máxima por cada pin | 40 mA |
| Dimensiones | 18,5mm x 43,2mm |

Tabla 1: Características del Arduino Nano 3.0

Fuente: <http://arduino.cc/es/Main/ArduinoBoardNano>

## Teoría de Control.

La teoría de control es una teoría matemática que rige la manipulación de los parámetros que afectan el comportamiento de un sistema, para producir un comportamiento deseado u óptimo. **[Zabczyk 1993].**

La teoría de control se ocupa del diseño de algoritmos de regulación de estado, observadores, e identificación de sistemas. Un sistema de control puede definirse como un arreglo de componentes acoplados de tal manera, que el arreglo pueda comandar, dirigir, o regularse a sí mismo o a otro sistema. **[Dulhoste 2011]**. Un sistema de control está constituido por entradas, salidas y estados **[Vidyasagar 2010].**

Se dice que un sistema o planta está en lazo abierto cuando las entradas no son afectadas o modificadas por los valores en las salidas de la planta **[Rodríguez 2013]**. La mayoría de los sistemas de lazo abierto son estables con entradas de referencia limitadas. De lo que normalmente carecen los sistemas de lazo abierto es de velocidad y precisión suficientes para seguir la entrada de referencia aplicada al sistema **[Alciatore 2008].**

Para un preciso control de un sistema es necesario usar retroalimentación de los sensores (por ejemplo, un codificador o un tacómetro). Al restar una señal de retroalimentación de una señal de entrada deseada (llamada valor de referencia de entrada), se tiene una medición del error en la respuesta. Al cambiar continuamente la señal de comando al sistema con base en la señal de error, se puede mejorar la respuesta del sistema. A esto se le llama control por retroalimentación o de lazo cerrado **[Alciatore 2008].**



Ilustración 3: Sistemas de control con y sin realimentación

Fuente: http://ayciaguillo.blogspot.com/2013/02/1-clase.html

## Algoritmo PID

Los controladores generales pueden tomar muchas formas, pero la mayoría de las aplicaciones industriales usan controladores PID o proporcional-integral-derivativo. La forma matemática de un controlador PID, donde la señal de error se expresa como e(t) es la siguiente:

Ecuación 1: Formula matematica del controlador PID

Fuente: **[Alciatore 2008]**

donde *Kp* se refiere como la ganancia proporcional, *Kd* es la ganancia derivativa y *Ki* es la ganancia integral. El control proporcional es el más intuitivo por que la señal de control es proporcional al error. Mientras más grande sea el error, mayor será la acción correctiva. Una enorme ganancia proporcional crea una respuesta rápida, pero puede conducir a exceso y oscilación, en especial si el sistema tiene poco amortiguamiento. La ganancia derivativa responde a la tasa de cambio de la señal de error. Esto permite al controlador anticipar cambios en la respuesta del sistema, que pueden resultar en menos exceso de oscilación amortiguada. La ganancia integral ayuda a eliminar error de estado estacionario al sumar errores a los largo del tiempo. Mientras mas tiempo permanezca el error en un lado de la entrada de referencia deseada, mas grande se vuelve la acción correctiva como resultado de la ganancia integral **[Alciatore 2008].**

CAPÍTULO III - Metodología

Para el desarrollo de este Trabajo Especial de Grado se utilizó una adaptación de la metodología de desarrollo de software en espiral. El modelo en espiral plantea que el software debe ser desarrollado en una serie de entregas incrementales. Durante las primeras iteraciones, la entrega incremental puede ser un modelo en papel o un prototipo. A lo largo de las últimas iteraciones, versiones en aumento más completas de los sistemas desarrollados son producidas. **[Pressman 2001].**

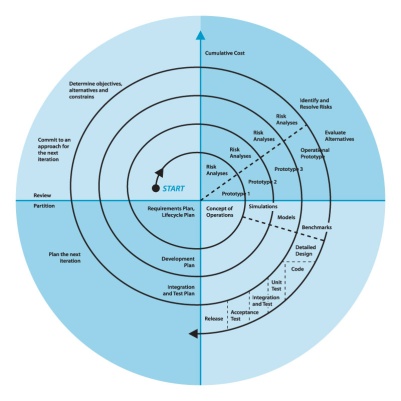


Figura 2 : Ciclo de vida en espiral.

*Fuente:* [*http://eternalsunshineoftheismind.files.wordpress.com/2013/03/spiral.jpg*](http://eternalsunshineoftheismind.files.wordpress.com/2013/03/spiral.jpg)

## III.1 – Descripción de las etapas de la metodología en espiral

Según lo expuesto en **[Pressman 2001],** un modelo en espiral está dividido en un número de etapas definidas, también llamadas regiones de tareas. Típicamente, hay entre tres (3) y seis (6) regiones de tareas.

Las etapas del modelo en espiral pueden ser resumidas de la siguiente manera:

1. Los requerimientos del nuevo sistema son definidos con tanto detalle como sea posible.
2. Un diseño preliminar es creado para el nuevo sistema.
3. Un primer prototipo del nuevo sistema es construido desde el diseño preliminar. Éste es usualmente una versión reducida del sistema, y representa una aproximación de las características del producto final.
4. Un segundo prototipo es desarrollado a partir de un cuádruple procedimiento:
   1. Evaluar el primer prototipo en términos de sus fortalezas, debilidades y riesgos.
   2. Definir los requerimientos del segundo prototipo.
   3. Planear y diseñar el segundo prototipo.
   4. Construcción y pruebas del segundo prototipo.
5. A la opción del consumidor, el proyecto entero puede ser cancelado si el riesgo es considerado muy alto. Factores de riesgo pueden involucrar sobrepasar el costo de desarrollo, cálculos erróneos en el costo de operación, o cualquier otro factor que podría, bajo el juicio del consumidor, resultar en un producto no satisfactorio.
6. El prototipo existente es evaluado de la misma manera que el segundo prototipo, y, si es necesario, otro prototipo es desarrollado a partir de él, de acuerdo con el cuádruple procedimiento expuesto más arriba.
7. Las etapas anteriores son repetidas hasta que el consumidor se encuentre satisfecho con la manera en que el prototipo refinado representa el producto final deseado.
8. El sistema final es construido, basado en el prototipo refinado.
9. El sistema final es evaluado y probado a fondo. Mantenimiento de rutina es llevado a cabo de manera continua para prevenir fallas de larga escala y prevenir falta de disponibilidad. **[Rouse 2007]**

## III.2 - Justificación de la metodología a utilizar

La elección de la metodología en espiral se debe principalmente, a que el desarrollo de componentes de software en robótica suele necesitar de un constante proceso de diseño, implementación, pruebas, y rediseño, con el fin de perfeccionar el producto final con cada iteración realizada. Esto porque, aunque pudiera realizarse un análisis previo de los requerimientos del software, es posible que hayan ciertas condiciones propias del sistema robótico que no se hayan podido observar antes del desarrollo del primer prototipo, y sí se puedan observar al realizar un rediseño basado en las fortalezas y debilidades de los prototipos anteriormente desarrollados.

Otra razón de peso por la que se eligió el modelo de desarrollo en espiral es porque el proyecto a realizar es de carácter experimental, y para alcanzar un desarrollo óptimo es muy probable que antes tengan que desarrollarse una serie de prototipos que permitan al equipo de desarrollo el análisis de las características cuya implementación pudiera facilitar la consecución de la investigación a realizar.

**Referencias bibliográficas**

**[Alciatore 2008]** Alciatore, D.; Histand, M. (2008). *Introducción a la Mecatrónica y los sistemas de medición.* Tercera edición. Interamericana editores.

**[Banzi 2011]** Banzi, M. y Cuartielles, D., *Descripción de la plataforma Arduino*. Obtenido de <http://www.arduino.cc/>

**[Bonastre 2010]** Bonastre, A. (2010) *Desarrollo de un Sistema Integrado de Navegación Inercial: Interfície IMU + FPGA.* Universitat Autónoma de Barcelona. Barcelona, España.

**[Burkamshaw 2010]** Burkamshaw, L. (2010). *Towards a Low Cost Quadrotor Research Platform.* Naval Postgraduate School. California, Estados Unidos de América.

**[Chin Kar 2007]** Chin Kar Wei. (2007). *Flight Dynamics and Control for an Indoor UAV.* Trabajo Especial de Grado de Ingeniería Mecánica. Universidad Nacional de Singapur.

**[Colton 2011 - 1]** Colton, S. (2011). *The great XBee 57.6kpbs mystery finally solved.* Obtenido de <http://scolton.blogspot.com/2011/09/great-xbee-576kbps-mystery-finally.html>

**[Colton 2011 – 2]** Colton, S. (2011). *PCB Quadrotor (Brushless).* Obtenido de <http://www.instructables.com/id/PCB-Quadrotor-Brushless/?lang=es>

**[Dignyu 2007]** Dignyu, X., YangQuan, C y Atherton, D. (2007). *Linear feedback control.* Siam.

**[Dulhoste 2011]** Dulhoste, J. *Teoría de control.* Universidad de los Andes. Mérida, Venezuela.

**[Nadales 2009]** Nadales, C. (2009). *Control de un quadrotor mediante la plataforma Arduino.* Trabajo Especial de Grado de Ingeniería Técnica de Telecomunicaciones, Universidad Politécnica de Catalunya, Barcelona, España.

**[Ogata 2011]** Ogata, K. (2011). *Ingeniería de control moderna.* Quinta edición.Pearson.

**[Pressman 2001]** Pressman, R. (2001). *Software Engineering: A practitioner’s approach.* Quinta edición. Mc-Graw Hill.

**[Rodríguez 2013]** Rodríguez, M. (2013). *Control a lazo abierto.* Obtenido de: http://prof.usb.ve/mirodriguez/control/Sistemas\_y\_transformada\_de\_laplace/control\_a\_lazo\_abierto.html

**[Rouse 2007]** Rouse, M. (2007). *Spiral model (spiral lifecycle model).* Obtenido de <http://searchsoftwarequality.techtarget.com/definition/spiral-model>

**[Shakev 2011]** Shakev, N.; Topalov, A.; Kaynak, O.; y Borisov, K. (2011). *Comparative Results on Stabilization of the Quadrotor Rotorcraft Using Bounded Feedback Controllers.* Trabajo presentado en el Journal of Intelligent and Robotics Systems 2011.

**[UniLeon 2013]** *El controlador PID básico.* Material de la cátedra de Laboratorio Remoto de Automática. Universidad de León. León, España.

**[Vidyasagar 2010]** Vidyasagar, M. (2010). *A tutorial overview of Control Theory for Non-Engineers.* The University of Texas at Dallas. Texas, Estados Unidos de América.

**[Zabczyk 1993]** Zabczyk, J. (1993). *Mathematical control theory: An introduction.* Birkhäuser. Boston, Massachusetts, Estados Unidos de América.