Universidad Autónoma de Querétaro
Facultad de Ingeniería
Maestría en Instrumentación y Control Automático

Implementación de un algoritmo de control para el levantamiento de un helicóptero de cuatro rotores

Tesis

Que como parte de los requisitos para obtener el Grado de
Maestría en Ciencias en instrumentación y control automático

Presenta:
Jesús Norberto Guerrero Tavares

Dirigido por:
Dr. Víctor Manuel Hernández Guzmán

Dr. Víctor Manuel Hernández Guzmán Presidente	Firma
Dr. Roberto Valentín Carrillo Serrano Secretario	Firma
Dr. Jorge Alberto Orrante Sakanassi Vocal	Firma
Dr. Edgar Alejandro Rivas Araiza Suplente	Firma
Dr. Juvenal Rodríguez Reséndiz Suplente	Firma

Dr. Aurelio Domínguez González
Director de la Facultad

Dr. Irineo Torres Pacheco
Director de Investigación y Posgrado

Centro Universitario
Querétaro, Qro.
Noviembre del 2014
México

Resumen

Dentro del desarrollo de los vehículos aéreos no tripulados (UAV), el helicóptero de cuatro rotores ha sido una de las plataformas más atractivas para el diseño de controladores debido a la simplicidad que conlleva su construcción, mantenimiento y toda esa gama extensa de aplicaciones en las que puede ser enfocado. En los últimos años, diversas estrategias de control para la orientación del helicóptero de cuatro rotores se han desarrollado, siendo los controladores PD y PID algunas de las estrategias más divulgadas debido a su relación de fácil implementación y desempeño. En este trabajo, se presenta la implementación de un control PD. El control se prueba en una plataforma experimental diseñada en su totalidad por el autor de acuerdo a las necesidades del controlador, cuenta con un diseño simplificado que permite su reproducción sin mayor dificultad, que además puede ser usado como una herramienta para probar otros algoritmos de control y como un prototipo con fines didácticos. Los resultados demuestran la efectividad del controlador propuesto sobre la plataforma experimental.

Palabras Claves: Cuatrirrotor, controlador PD, DSC.

Summary

In the development of unmanned aerial vehicles (UAV), the four-rotor helicopter has been one of the most attractive for the design of controllers platforms due to the simplicity involved in construction, maintenance and all that vast range of applications it can be focused. In recent years, various control strategies for targeting four-rotor helicopter have been developed, with the PD and PID controllers reported some of the strategies because of its easy implementation and relationship performance. In this work, the implementation of a PD control is presented. The control is tested on an experimental platform designed entirely by the author according to the needs of the controller, it has a simplified design that allows reproduction without difficulty, also can be used as a tool to test other control algorithms and as a prototype for teaching purposes. The results demonstrate the effectiveness of the proposed controller on the experimental platform.

Key Words: Quadcopter, PD controller, DSC.

Dedicado a
mis amados padres

Agradecimientos

Deseo agradecer a mi director de tesis, Dr. Víctor Manuel Hernández Guzmán por proporcinarme la oportunidad de trabajar en uno de sus proyectos, por el tiempo que me dedico con sus asesorías y valiosas aportaciones al presente trabajo.

Mis agradecimientos para el Dr. Roberto Valentín Carrillo Serrano por sus aportaciones esenciales durante toda la etapa de la implementación del proyecto, así como también le agradezco que me haya brindado la oportunidad de trabajar con él.

También agradezco a mis amigos Salvador, Sergio, Oswaldo e Isabel por sus valiosos comentarios y correcciones a este documento.

Un agradecimiento para el Dr. Marcelino y los futuros Drs. Moisés y Fortino, por las facilidades proporcionadas en cuanto a la implementación del prototipo.

Un profundo agradecimiento a Cristina Chávez por su valiosa e invaluable ayuda en la revisión y corrección de este documento.

Finalmente, quiero agradecer a Conacyt por el apoyo financiero para realizar esta investigación.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En años recientes, investigadores e ingenieros de diversas áreas han trabajado intensamente en el desarrollo de vehículos aéreos capaces de realizar misiones específicas con mínima intervención humana. Este tipo de vehículos son comunmente conocidos como UAV (por sus siglas en inglés Unmanned Aerial Vehicle) o drones. (Madani y Benallegue, 2006a)

Un drone se define como un vehículo aéreo no tripulado reutilizable, controlado a distancia, automático o semi-automático, capaz de mantener un nivel de vuelo controlado y sostenido, propulsado por un motor de reacción o eléctrico, susceptible de llevar diferentes cargas que le servirán de ayuda para efectuar diferentes tareas durante un vuelo y que pueden variar en función de sus capacidades.

Los drones nacieron como un producto tecnológico con fines bélicos. El primer UAV resgistrado se remonta al año de 1917 cuando la marina de los Estados Unidos presentó el *"Aerial Torpedo"* (Figura 1.1), un vehículo bombardero biplano sin piloto fabricado con madera, con un peso de 270 Kg e impulsado con un motor Ford de 40 caballos de fuerza. Este vehículo contaba con las siguientes características: un giroestabilizador para mantener el nivel de la aeronave, un giroscopio de dirección automática encargado de mantener el vehículo en un rumbo preestablecido, un barómetro para indicar la altitud de crucero causando que la aeronave se estabilizara, un motor con contador de revoluciones para determinar cuando la nave debería de cortar el suministro de energía y navegar hasta su objetivo. Además, se instaló un generador eléctrico para proporcinar potencia a los motores giroscópicos y servomotores que movían las superficies de control de vuelo de la aeronave. El *"Kettering bug"* (Figura 1.2) es una versión mejorada, más liviana, presentada por el ejército de Estados Unidos.

Dentro de las versiones de UAV con radio control, en 1927, el ejército de Inglaterra mostró el *"Longe-Range Gun With Lynx Engine"*, un vehículo monoplano con una capacidad de cargamento de 114 Kg sobre un rango de operación de 480 Km, cabe mencionar que el radio control se utilizaba sólo para la maniobra de despegue, después de esto, el piloto automático

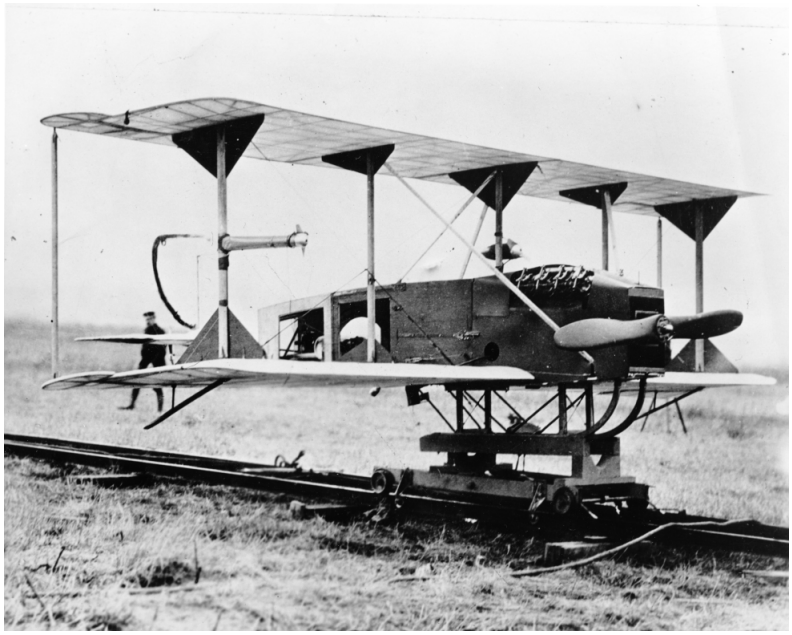

Fig. 1.1. El *Aerial Torpedo* considerado como uno de los primeros UAV.

tomaba el curso preestablecido hasta llegar al objetivo definido. Luego del lanzamiento de este prototipo, el interés en el control de aeronaves por medio de radio frecuencia se incrementó, siendo que en el año de 1934, el prototipo *"Queen Bee"* fue presentado por la Armada Real de Inglaterra, este vehícullo alcanzaba distancias de hasta 480 km, con una velocidad de $160 km/h$ y una altitud de 5 km. Al mismo tiempo, la Radioplane Company de los Estados Unidos fabricaba el *"RP4"*, una aeronave desarrollada para funcionar como objetivo en prácticas de artillería. Mientras tanto, el ejército Nazi alemán lanzaba el *"Fieseler Fi 103"*, la primera nave de propulsión. Otro logro importante durante este periodo fue el prototipo presentado por el ejército de EU, el llamado *"Fox project"*, una aeronave equipada con una cámara de televisión RCA. Este vehículo era manejado por radio control desde una nave tripulada y equipada con una pantalla de televisión. El *"Fox project"* es considerado como una de los primeros intentos de control por tele-operación.

Durante la década de 1960, los UAV comenzaron a ser usados para propósitos de reconocimiento, como por ejemplo, el *"Firebee"*, que contaba con una cámara fotográfica. Éste era desplegado por un avión tripulado, contaba con un paracaídas y una vez obtenidas las fotografías del territorio enemigo, era recuperado por un helicóptero en alguna posición estratégica y posteriormente regresado a la base militar donde se analizaban las fotografías tomadas. Esta práctica cobró popularidad pronto, pues más allá de que se evitaba el exponer la vida de un ser humano a situaciones de riesgo, los gobiernos no se tendrían que preocupar en caso de que el enemigo capturara un piloto vivo agravando los conflictos diplomáticos.

En años posteriores, los UAV -que a menudo eran vistos como juguetes poco fiables y

Fig. 1.2. El *Kettering Bug* como versión mejorada del *Aerial Torpedo*.

caros- cobraron importancia debido a la victoria de la Fuerza Aérea Israelí sobre la Fuerza Aérea Siria en 1982. Los israelíes desplegaron sus aviones de combate junto con los UAV en su ataque provocando docenas de bajas sirias con pérdidas mínimas. Los UAV desplegados fueron utilizado como señuelos electrónicos, bloqueadores electrónicos y para reconocimiento con video en tiempo real.

La primera vez que los UAV se utilizaron para probarse como un arma efectiva en combate fue durante una simulación de batalla en 1987, estas aeronaves presentaban el sistema de control de vuelo más eficiente en la época. Después de esto, el ejército de EE. UU. consideró a los UAV como una herramienta de reconocimiento y combate fundamental en conflictos bélicos. La tendencia del uso de estas aeronaves se reforzó cuando comenzó el conflicto en Afganistán en 2001. El UAV *"Predator RQ -1L"*, fabricado por General Atomics, fue el primer UAV desplegado en los Balcanes durante 1995 y en Irak en 1996, demostrando su eficacia en la "Operación Libertad"para Irak.

Afortunadamente, desde hace algunos años, los drones ya no son de uso exclusivo de la milicia, también son utilizados en un pequeño pero creciente número de aplicaciones civiles. A continuación se presentan algunas de sus aplicaciones:

1. Labores de lucha contra incendios: en el campo forestal, los drones permiten una supervisión constante en horas de alto riesgo de un área boscosa, en busca de puntos activos o conatos de incendio. Ver Figura 1.3.

2. Seguridad Civil: en algunos países, los UAV son usados por la policía local como ayuda en la vigilancia de marchas.

Fig. 1.3. Un UAV tomando fotografías de un incendio en progreso.

3. Construcción e inspecciones: se utilizan para la inspección de grandes estructuras desde el aire. Como ejemplo, algunas empresas que dan mantenimiento a los aerogeneradores utilizan a los drones para detectar fallas en la estructura del generador.

4. Vigilancia de tráfico en carreteras: Los drones pueden monitorear ciertos puntos de control como soporte en la vigilancia de la carretera.

5. Operaciones de búsqueda aérea y salvamento: uno o más drones pueden buscar personas desaparecidas en lugares abiertos o de difícil acceso como zonas montañosas o nevadas. Su reducido tamaño permite tenerlos siempre disponibles en estaciones de montaña, reduciendo considerablemente el tiempo de búsqueda. Ver 1.4.

6. Predicción meteorológica: debido a que estas aeronaves cuentan con cierta capacidad de carga, la colocación de diferentes tipos de sensores es realizable.

7. Medio Ambiente: parametrización del índice de contaminación lumínica para elaborar mapas de polución lumínica y monitorizar la eficiencia de medidas ecoenergéticas; control y seguimiento de accidentes industriales con vertidos tóxicos en medios acuáticos y terrestres; control de áreas de depósito y almacenaje de residuos industriales y de su tratamiento.

8. Agricultura:Control y monitorización del estado de los cultivos mediante imágenes multiespectrales, control de la eficiencia de regadíos, conteo y supervisión de producción agrícola subvencionada (por ejemplo, número de árboles).

9. Geología: realización de mapas geológicos sedimentológicos, mineralógicos y geofísicos; control y monitorización de explotaciones mineras y su impacto ambiental: movimientos de tierras, producción de áridos, residuos metálicos, balsas de decantación, etc. Determinación y control a escala centimétrica de áreas con riesgos geológicos

Fig. 1.4. Un UAV puede ser usado en operaciones de búsqueda.

asociados o caracterización de zonas con riesgo de aludes utilizando imágenes multiespectrales para determinar la humedad de la nieve, cámaras térmicas para determinar su temperatura y técnicas estereoscópicas para determinar grosores.

1.1. Características del UAV

Un UAV está compuesto generalmente de dos partes: la estación en tierra y la parte aérea. La estación en tierra fija o móvil asegura la comunicación entre la parte aérea y los dispositivos que controlan el drone. Esta parte también esta conformada de medios de despegue y de recuperación, así como de medios de mantenimiento. Además se compone de una o varias plataformas aéreas integradas con una carga útil adaptada de acuerdo al objetivo (cámara, GPS, central inercial, etc.) y de un sistema de comunicación de información (Castillo et al., 2007).

Los drones son vehículos aéreos de escala pequeña, menos costosos y más simples de construir que un avión convencional. El tamaño de los drones puede variar (desde algunos centímetros hasta varios metros), al igual que su forma y su tipo de propulsión, por ejemplo algunos están equipados de reactores, otros de hélices o rotores, etc.

Es necesario mencionar que en algunas ocasiones los vehículos aéreos en miniatura se utilizan simplemente como plataformas tecnológicas, útiles para la validación de determinados conceptos aerodinámicos y leyes de control. Estas plataformas son de gran utilidad pues permiten observar y analizar los límites del vehículo con la finalidad de verificar su confiabilidad y desempeño en vuelo (Castillo et al., 2007).

En este trabajo de investigación se utilizó la estructura de un helicóptero de cuatro rotores comercial de la compañía *Draganfly*, en este caso se prescindió de la electrónica incluida con el helicóptero pues no permitía la reprogramación del procesador central por su estructura cerrada, por tanto, algunas de las tareas principales fueron precisamente el diseño, la

construcción y la prueba del prototipo donde se validaron los controladores.

1.2. Descripción del problema

La dinámica completa de un helicóptero de cuatro rotores es matemáticamente compleja, esto quiere decir que desarrollar un control tomando en cuenta toda la dinámica del sistema es poco eficiente pues se requeriría de un gran poder de procesamiento en un sistema embebido. La solución que se le da a la problemática es tomar en cuenta sólo una parte de dicha dinámica y enfocarse en la estabilización o levantamiento del vehículo.

Actualmente el problema que se tiene al diseñar un sistema de control de posición para un helicóptero de cuatro rotores es que la dinámica es no lineal y generalmente, el control del ángulo yaw se realiza mediante aproximaciones lineales puesto que analíticamente, éste aumenta la complejidad del análisis.

Un paso interesante es el de diseñar un controlador de posición para el helicóptero cuatri-rotor que estabilice al vehículo en cualquier posición cartesiana, incluyendo el control del ángulo yaw. Esto se puede intentar resolver mediante controladores Proporcionales-Derivativos (PD).

Esta investigación está enfocada en analizar la dinámica del sistema no lineal para controlar y estabilizar el helicóptero cuatri-rotor, para ello se desarrolla un controlador proporcional-derivativo (PD) con compensación de gravedad y prealimentación de aceleración (Hernández et al.,2010), el cual ha demostrado tener un desempeño favorable en la estabilización de la aeronave bajo simulación, con ciertas restricciones que son definidas en la sección correspondiente. En este trabajo se pretende probar un diseño con PD pero sintetizado en un controlador digital de señales (DSC, por sus siglas en inglés *Digital Signal Controller*), todo esto con el fin de darle un cierto grado de autonomía al helicóptero de cuatro rotores y a su vez, en un futuro, pueda ser utilizado en aplicaciones comerciales y/o civiles.

1.3. Hipótesis

La hipótesis general se describe a continuación:

Un sistema como el cuatrirotor es altamente no lineal y el control, en paralelo, del ángulo yaw incrementa la complejidad matemática del sistema por lo que en trabajos previos esta variable se controla mediante aproximaciones lineales, entonces, la introducción de controladores PD con compensación de gravedad aseguran estabilidad del sistema bajo ciertas condiciones permitiendo el análisis del sistema completo, incluyendo el ángulo yaw, para el control de la aeoronave.

1.4. Objetivos

Implementar controladores PD que permitan estabilizar el helicóptero cuatrirotor en el espacio euclidiano.

1.4.1. Objetivos particulares

1. Validar el desempeño del controlador PD en simulación.
2. Diseñar y construir el hardware del helicóptero de cuatro rotores.
3. Generar una estación de trabajo para probar y validar el algoritmo de control, que al mismo tiempo servirá como un equipo con fines de investigación y de herramienta didáctica.

1.5. Contribución

La principal contribución de este trabajo de investigación es el diseño, construcción y documentación de el prototipo de un helicóptero de cuatro rotores con una arquitectura abierta para uso, reproducción, modificación o mejoramiento con fines didácticos y/o alguna aplicación en proyectos futuros.

Capítulo 2

Estado del arte

Aunque las aeronaves no tripuladas nacieron como un desarrollo de tecnología bélica, las aplicaciones en otras áreas se han ido incrementando en el transcurso del tiempo. Con el objetivo de desarrollar vehículos completamente autónomos que sean capaces de realizar tareas específicas en escenarios reales, grupos de investigación en todo el mundo han estado describiendo nuevas estrategias de control para llevar a cabo dicha tarea. El helicóptero de cuatro rotores es una de las plataformas experimentales que se han ido popularizando debido a su versatilidad, complejidad y toda esa potencial gama de aplicaciones reales con las que cuenta. Día con día, nacen nuevas estrategias de control para este tipo de aeronaves se presentándose el diseño de nuevos controladores para su despegue-aterrizaje, estabilización de los ángulos de orientación, seguimiento de trayectorias o realización de maniobras agresivas. La mayoría de las técnicas desarrolladas requieren conocer los estados de orientación o de posición del vehículo, aunque inicialmente, los estados del sistema eran determinados por técnicas como el “Dead reckoning"(Navegación por estima), una técnica que estima las posiciones del vehículo mediante relaciones trigonométricas sin contar con una marco de referencia en tierra, siendo sus resultados, en general, pobres. Por ello, se ha utilizado toda una combinación de herramientas para determinar los estados del sistema, desde sensores colocados en el helicóptero hasta el uso de cámaras, todos con sus ventajas y desventajas.

En la Figura 2.1 se presenta algunas de las estrategias de control utilizadas para estabilizar un helicóptero de cuatro rotores. A continuación se muestra algunos trabajos respecto a cada una de las áreas mencionadas.

2.1. Control por modos deslizantes

En el artículo de Xu (2006), se presenta un controlador por modos deslizantes, describen la obtención del control mediante la partición del sistema del cuatrirotor en dos subsistemas,

Fig. 2.1. Estrategias de control utilizadas para estabilizar el cuatrirotor.

los cuales son transformados en cadenas de integradores. El controlador resultante logra estabilizar la aeronave y llevarla a cualquier posición deseada con cierto ángulo yaw deseado mientras trata de mantener a los otros dos ángulos de orientación (pitch y roll) sin variación. El controlador por modos deslizantes es comparado con un control PID en simulación. El efecto del chattering es minimizado introduciendo una versión continua de la función signo.

En Khelfi y Kacimi (2012), proponen un controlador por modos deslizantes de primer orden basado en la linealización por retroalimentación de la salida con el fin de atenuar las incertidumbres paramétricas del modelo, asumiendo que se conocen todos los estados del sistema. La aportación de este artículo es que el controlador propuesto prescinde de los parámetros del sistema y sólo se sintoniza con un ajuste de cuatro ganancias. El controlador por modos deslizantes se compara con un controlador “no robusto"donde se comprueba, bajo simulación, que dicho controlador puede estabilizar el cuatrirotor en el espacio y sin error en estado estacionario.

Posteriormente, en Runcharoon (2013) se describe un controlador por modos deslizantes de primer orden para el control de altitud, seguimiento de trayectorias y estabilización de un helicóptero de cuatro rotores. La ventaja principal de este controlador es su robustez ante perturbaciones, éste es probado con entradas adicionadas con ruido Gaussiano. La desventaja evidente de este controlador es el efecto de “chattering"debido a la introducción de señales signo en el controlador, para minimizar el impacto de este efecto, los autores sustituyen las funciones signo por funciones de saturación. Mediante simulación se demuestra que el controlador es capaz de llevar al helicóptero a una posición (x, y, z) deseada.

2.2. Backstepping

En Madani y Benallegue (2006a) se desarrolla un controlador backstepping para el seguimiento de trayectorias, en dicha estrategia se eligen los estados del sistema (x, y, z, yaw) para controlar al helicóptero. La obtención de los controladores se hace bajo la teoría clásica de backstepping, para ello dividen el sistema en tres subsistemas donde por medio de funciones de Lyapunov aseguran convergencia en tiempo finito. Los resultados se presentan en simulación. La desventaja de este método recae en la complejidad del control propuesto.

Más tarde, Madani y Benallegue (2006b) presentan el mismo controlador de su publicación anterior pero aplicado en un prototipo experimental. Usan un compás magnético para obtener la lectura del ángulo yaw y sensores ultrasónicos para medir las posiciones cartesianas (x, y, z) que requiere el controlador. El prototipo está fijo a una base que sostiene al helicóptero por la parte posterior de su estructura, una tarjeta adquisidora para los sensores que después fueron filtrados en la computadora usando un tiempo de muestreo de 10 ms. Los resultados muestran el seguimiento de la trayectoria con algunos errores de seguimiento.

En años recientes, (Saif et al., 2012) derivan un controlador backstepping modificado para abordar el problema de segumiento de trayectorias del cuatrirotor. Basándose en el procedimiento de la división del sistema en tres subsistemas -como lo hicieron los autores anteriores- describen la obtención de su controlador mediante teoría de backstepping clásica, introduciendo expresiones matemáticas que simplifican la obtención del controlador, resultando en un controlador simple con sólo seis ganancias de ajuste. Los resultados se presentan en simulación.

2.3. Pasividad

El artículo de Santos et al. (2013) muestra un algoritmo de control para la estabilización del helicóptero. Los autores utilizan pasividad para diseñar un control óptimo y es comparado con dos controladores: un controlador LQR y uno con retroalimentación de estado. Bajo simulación demuestran que el control propuesto es superior a los otros algoritmos de comparación y que además es robusto por el rechazo de perturbaciones debidas al viento.

2.4. Linealización por retroalimentación

El algoritmo de control de Al-Hiddabi (2009) se basa en la linealización por retroalimentación de entrada-salida para el seguimiento de trayectorias de un helicóptero de cuatro rotores. Debido a que la dinámica del vehículo es compleja, particionan el sistema general en

subsistemas de forma que cada subsistema sea de fase mínima y así aplicar la metodología para obtener un controlador por retroalimentación de entrada-salida.

En Voos (2009) se utiliza la misma metodología anterior solo que el algoritmo de control propuesto se simula con parámetros determinados por un prototipo experimental. Ambos controladores sólo son simulados.

2.5. Lógica difusa

Un controlador difuso, robusto y adaptable para la estabilización de un cuatrirotor se presenta en Coza y Macnab (2006). Utilizando las funciones de membresía Gausaianas normalizadas (GMF, por sus siglas en inglés) estiman las funciones no lineales mediante funciones lineales, donde a través de la variación de las funciones centro de la técnica conocida como “e-modification"se robustece el controlador ante perturbaciones periódicas. Los resultados bajo simulación muestran que el helicóptero puede ser estabilizado y llevado a una referencia deseada, siempre y cuando se cumplan ciertas condiciones internas del controlador fuzzy.

En el trabajo (Santos y López, 2010) se diseña un controlador por lógica difusa (*fuzzy*), el objetivo principal es el de estabilizar y mover el helicóptero a cualquier posición deseada, esto se logra mediante un controlador PID fuzzy modificado, donde las funciones de pertenencia modifican la velocidad ángular de cada motor, por ejemplo, para modificar la altura del cuatrirotor, se aumenta o decrece la velocidad por igual en los cuatro rotores, para el caso de la modicificación del ángulo pitch y roll, se incrementa la velocidad de un motor al mismo tiempo que decrementa proporcionalmente la velocidad del motor contrario dependiendo cual sea el movimiento deseado, para el yaw, se incrementan/decrementan en proporción la velocidad de los motores por parejas. Las pruebas en simulación sugieren que el control es capaz de estabilizar la aeronave y que además es capaz de llevarlo a una posición deseada. El inconveniente que presenta esta metodología es que sólo se puede modificar una acción de movimiento (a la izquierda, a la derecha, giro y ascenso/descenso) a la vez.

En Rabhi et al. (2011) se muestra un controlador fuzzy robusto para la estabilización de un helicóptero de cuatro rotores. Partiendo de un controlador fuzzy propuesto por Takagi-Sugeno y utilizando la técnica de compensación distribuida paralela, el controlador es transformado a una aproximación lineal del sistema, y para asegurar estabilidad se emplea el uso de las inecuaciones matriciales lineales (LMI, por sus siglas en inglés Linear Matrix Inequality) para encontrar un conjunto de matrices (A, B) que hagan al sistema asintóticamente estable. Además, como el sistema es multiplicado por una función que depende de parámetros inmedibles, entonces, en el articulo se emplea un diferenciador por modos deslizantes para proporcionarle robustez al controlador. Los resultados en simulación muestran que es

posible estabilizar el helicóptero de cuatro rotores.

2.6. PD/PID

El trabajo de Erginer y Altug (2007), se diseña un controlador PD clásico para estabilizar la maniobra de “hovering"(suspender en el aire) y el descenso de un helicóptero de cuatro rotores. Se controlan los ángulos roll, pitch y yaw, como derivado de los dos primeros, se controla también la posición cartesiana $x - y$, este control derivado posee una indeterminación en un punto de la expresión. También se presenta una metodología para estabilizar el helicóptero por medio del uso de cámaras en conjunto con un GPS y una IMU (Unidad de Medida Inercial, del inglés Inertal Measurement Unit) como sensores de posición y orientación del helicóptero, dichas posiciones son introducidas al controlador para realizar las maniobras antes mencionadas. El trabajo se presenta a nivel simulación bajo SIMULINK con un tiempo de convergencia máximo de 10 segundos en un desplazamiento máximo de dos metros.

Con Naidoo et al. (2011) es desarrollado un controlador lineal Proporcional-Derivativo (PD) para estabilizar el cuatrirotor, el controlador se aplica a 4 grados de libertad, los grados de orientación roll, pitch y yaw, y un controlador para la dirección z, con el objetivo principal de que la aeronave se suspenda en el aire. No se considera alguna perturbación externa en el controlador. Se compara el comportamiento de la aeronave con y sin controlador, primero se le inyecta una velocidad constante a todos los motores, que en teoría, haría que el sistema se posicionara a cierta altura y se mantuviera estable. Cuando se prueba la hipótesis bajo simulación y sin controlador, se observa que efectivamente la velocidad se mantiene pero los ángulos de orientación no permanecen estáticos, hay una desviación que se incrementa conforme el tiempo transcurre. Cuando se prueba el controlador PD sobre simulación se observa que el estado z converge al estado deseado z_d y que los estados de orientación se mantienen cerca del valor deseado a excepción del ángulo yaw, que presenta un efecto de rampa cuando el controlador lo lleva al estado estacionario, que como mencionan los autores, se debe al error de integración acumulado en los estados. El controlador no se prueba en ninguna plataforma experimental.

En Li y Li (2011) se describe un controlador PID partiendo desde un modelo no lineal pero sin tomar en consideración perturbaciones externas. Para diseñar el controlador se particiona el modelo en dos subsistemas uno para el control de la posición cartesiana (x, y, z) y otro para la orientación $(roll, pitch, yaw)$ debido a que el movimiento lineal no afecta al movimiento angular pero sí en viceversa. Mediante el método de perturbaciones pequeñas se linealiza cada subsistema y se obtienen las funciones de transferencia de cada uno de los grados de libertad, para posteriormente plantear un controlador PID clásico para cada uno

de ellos (se retroalimenta la posición de la aeronave). El controlador PID es simulado bajo simulink donde es sometido a entradas tipo escalón, el controlador, efectivamente sigue la referencia. Todos los grados de libertad son expuestos a una señal tipo escalón. Finalmente, los autores implementan la estrategia de control, la evidencia que presentan muestra que al final sólo controlaron los ángulos de orientación (roll, pitch y yaw) con un error de seguimiento de 5° máximo.

En el mismo año, en Fang et al. (2011) se desarrolla un controlador PID clásico para estabilizar la postura de un cuatrirotor, donde se menciona que se controlan los seis grados de libertad. La innovación expuesta es la de una sintonización de las ganancias en línea, esto es, mientras la aeronave está en vuelo, sirve además como un método para rechazar perturbaciones. El ajuste de ganancias se hace con base en una tabla donde se evalúa el valor de la aceleración de los grados de libertad $x - y$, que como se ve en el modelo dinámico pueden ser calculadas mediante la actualización de los valores de los grados de orientación, así, por ejemplo si la aceleración de $x - y$ está en un rango superior a la unidad se eligen ciertas ganancias que fueron previamente probadas, si la aceleración cambia en cualquiera de los dos grados debido al efecto de una perturbación, entonces las ganancias se actualizan para estabilizar la aeronave y así sucesivamente, dependiendo cual sea el caso fue probado para 9 grupos de ganancias diferentes. Como se mencionó anteriormente, esta metodología ofrece una solución simple para el rechazo de perturbaciones, aunque el controlador se diseñó basado en un modelo que no contempla perturbaciones, esto no implica que lo hace inmune a cualquier perturbación, sólo a un cierto rango, que como se observa en el artículo, está limitado a pequeñas aceleraciones pero que representa una metodología que hace al controlador un poco más robusto que un PID clásico.

2.7. Investigaciones realizadas en la Universidad Autónoma de Querétaro

Dentro de la clasificación de trabajos de investigación realizados en la Universidad Autónoma de Querétaro se encuentran dos tesis de maestría que tratan sobre el control de un helicóptero de cuatro rotores. El primer trabajo desarrollado en la Universidad es de Molina J. (2009) en donde desarrolla una estrategia de control para posicionamiento del cuatrirotor mediante el uso de un PD clásico, demuestra que con el controlador propuesto es posible llevar a la aeronave a cualquier punto en el plano cartesiano. El trabajo sólo llego a simulación en MATLAB. Posteriormente, en la investigación de Nieto J. (2010) se propone un controlador PID para estabilizar el helicóptero, nuevamente se muestra en simulación que el controlador propuesto puede mover al helicóptero en cualquier posición deseada en el plano

cartesiano. Más allá de la simulación, el controlador fue probado en una plataforma experimental basado en un producto comercial (draganfly), sensando las posiciones de orientación referenciadas con potenciómetros. Con los datos de posición de los sensores, las operaciones del controlador son realizadas en la PC y las señales de regulación son enviadas vía inalámbrica mediante un control remoto modificado al helicóptero de cuatro rotores donde se ejecutan dichas señales. El prototipo estaba fijo a una base que sujetaba a toda a la estructura.

Capítulo 3

Descripción del helicóptero de cuatro rotores

En este capítulo se mostrará una descripción detallada del funcionamiento del helicóptero de cuatro rotores. Se introducirán las características principales de la aeronave y mediante el enfoque Euler-Lagrange se obtendrá el modelo matemático del mismo.

3.1. Descripción del cuatrirotor

El helicóptero de cuatro rotores, mostrado en la Figura 3.1, es un sistema de seis grados de libertad, multivariable, fuertemente acoplado y subactuado. La estructura del cuatrirotor se conforma por una base rígida y simétrica, posee cuatro rotores ubicados al final de los brazos de la base. El vehículo es controlado mediante la variación de la velocidad angular de cada uno de sus motores eléctricos. Los rotores frontal y trasero giran en sentido horario, mientras que los rotores derecho e izquierdo rotan en sentido anti-horario. Con este arreglo, los efectos giroscópicos y torques aerodinámicos tienden a cancelarse en vuelo sostenido. De la Figura 3.1se muestra un diagrama de cuerpo libre de un cuatrirotor, se puede observar que hay tres marcos de referencia, el marco inercial fijo a tierra, E, con ejes coordenados (x,y,z) apuntando al norte, este y hacia abajo, y el marco de cuerpo, el origen del marco local, L, está localizado en el centro de gravedad del cuatrirotor pero sus ejes coordenados (x_L, y_L, z_L) están fijos y definidos exactamente como los ejes coordenados del marco inercial. El origen del cuerpo, B, está fijo al centro de gravedad del cuatrirotor con ejes coordenados (x_B, y_B, z_B) apuntando hacia fuera de la nariz, hacia fuera del lado derecho del cuatrirotor y debajo de la parte inferior del helicóptero, respectivamente (Alderete,2010).

La orientación del cuatrirotor se realiza mediante el ajuste de los ángulos de Euler: yaw ψ, pitch θ y roll ϕ. El posicionamiento en el plano $x - y$ está relacionado directamente con

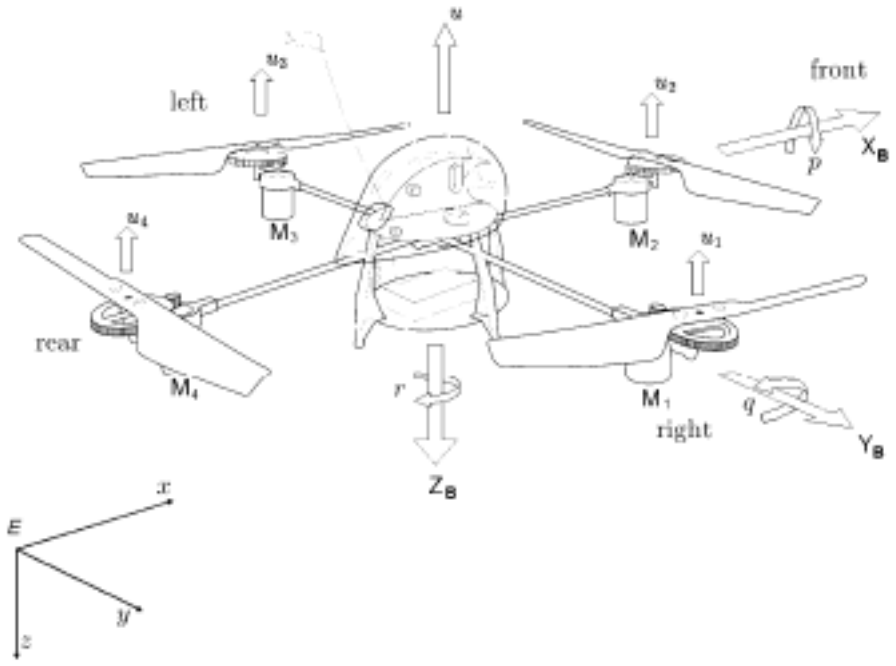

Fig. 3.1. Descripción del helicóptero de cuatro rotores.

la modificación de dichos ángulos.Como ya se mencionó, el helicóptero es controlado por la manipulación de las fuerzas de empuje proporcionado por cada motor así como el equilibrio del torque de arrastre. Por ejemplo, para que el helicóptero realice la acción de “hovering", todos los rotores deberán de aplicar una fuerza de empuje constante que mantendrá el balance de la aeronave. Para modificar el movimiento vertical, las velocidades de los motores se deberán incrementar o decrementar simultaneamente de forma que se mantenga el balance del helicóptero pero que a su vez sea capaz de modificar la altura. Para el cambio de dirección, el ángulo yaw debe ser controlado mediante la manipulación del torque de balanceo, dependiendo la dirección en la que se desee que el helicóptero rote. De igual forma, el posicionamiento del helicóptero en el plano cartesiano se logra mediante la variación de los ángulos pitch y roll, aplicando fuerzas de empuje diferenciales en rotores opuestos como se puede observar en la Figua R.

De la misma figura se puede observar que el motor M_i con $i = 1, ..., 4$ produce una fuerza f_i, la cual es proporcional a la raíz cuadrada de la velocidad angular, esto es, $f_i = k\omega_i^2$. Debido a que los rotores del helicóptero sóo pueden girar en sólo una dirección fija, la fuerza f_i siempre es positiva. El empuje principal u es la sima de los empujes proporcionados por cada motor. El torque en pitch está en función de una diferencia $f_1 - f_3$, el torque en roll es una función de $f_2 - f_4$, mientras que el torque de yaw es la suma de $\tau_{M_1} + \tau_{M_2} + \tau_{M_3} + \tau_{M_4}$, donde τ_{M_i} es el torque del motor i debido a la aceleración del eje y a las hélices. El torque

del motor se opone a una resistencia aerodinámica τ_{drag}, tal que:

$$I_{rot}\dot{\omega} = \tau_{M_i} - \tau_{drag} \tag{3.1}$$

Donde I_{rot} representa el momento de inercia de un rotor alrededor de un eje. La resistencia aerodinámica se define como sigue:

$$\tau_{drag} = \frac{1}{2}\rho A \upsilon^2 \tag{3.2}$$

donde ρ es la densidad del aire, A es el área del rotor y υ es la velocidad relativa del aire. Para un rotor con velocidad angular ω, la velocidad lineal es proporcional al radio de rotación R,

$$\upsilon = \omega R \tag{3.3}$$

Con la relación anterior, la resistencia aerodinámica puede ser reescrita como

$$\tau_{drag} = k_{drag}\omega^2 \tag{3.4}$$

Donde k_{drag} es una constante positiva dependiente de la densidad del aire, el radio, el área del rotor y otros factores. Para maniobras estacionarias, ω es una constante, entonces:

$$\tau_{M_i} = \tau_{drag} \tag{3.5}$$

Así, el movimiento en pitch se obtiene mediante el incremento de la velocidad del motor trasero M_3 mientras se reduce la veloidad del motor delantero M_1. Similarmente, el movimiento en roll se obtiene modificando las velocidades de los motores laterales. Para yaw, se requiere incrementar el torque de los motores, trasero y delantero (τ_{M_1} y τ_{M_3}), mientras se reducen los torque de los motores laterales (τ_{M_2} y τ_{M_4}). Tales movimientos se pueden realizar mientras se mantiene constante el empuje total.

3.2. Modelo del helicóptero de cuatro rotores

El modelo del cuatrirotor es obtenido mediante la representación de la aeronave como un cuerpo rígido envuelto en un espacio tridimensional y sujeto al un empuje y tres torques: roll, pitch y yaw.

De acuerdo a (Alderete; Benallegue et al., 2006; Beard, 2008) el modelo dinámico del

cuatrirotor está dado como:

$$m\ddot{\psi} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} mg + F_\xi + \begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix} \tag{3.6}$$

$$J\dot{\omega} = -\omega \times \omega + \tau + \begin{bmatrix} A_p \\ A_q \\ A_r \end{bmatrix} \tag{3.7}$$

$$\hat{F} = \begin{bmatrix} 0 \\ 0 \\ -u \end{bmatrix}; F_\xi = R^{-1}\hat{F} \tag{3.8}$$

donde el símbolo «x» denota el producto vectorial, m es la masa del helicóptero, g es la constante gravitatoria, $\xi = [x, y, z]^T$, R es la matriz de rotación definida como sigue:

$$R = \begin{bmatrix} c\psi c\theta & s\psi c\theta & -s\theta \\ -s\psi c\phi + c\psi s\theta s\phi & c\psi c\phi + s\psi s\theta s\phi & c\theta s\phi \\ s\psi s\phi + c\psi s\theta c\phi & -c\psi s\phi + s\psi s\theta c\phi & c\theta c\phi \end{bmatrix} \tag{3.9}$$

donde $c = cos(\cdot)$ y $s = sin(\cdot)$. La velocidad angular expresada con respecto al marco del cuerpo está definida como $\omega = [p, q, r]^T$ y $J = diagJ_{xB}, J_{yB}, J_{zB}$ como la matriz de inercia donde J_{xB}, J_{yB}, J_{zB} representan las inercias del cuatrirotor medidas alrededor de los ejes del marco del cuerpo. Por otro lado, $[A_x, A_y, A_z]^T$ y $[A_p, A_q, A_r]^T$ son las fuerzas aerodinámicas y momentos actuando sobre el cuatrirotor y son calculadas como $A_i = \frac{1}{2}\rho_{air}C_iW^2$, donde ρ_{air} es la densidad del aire, W es la velocidad del helicóptero con respecto a la velocidad del aire y C_i es el coeficiente aerodinámico el cual depende del ángulo entre el helicóptero y la dirección del viento. Se asume que las perturbaciones $[A_x, A_y, A_z]^T$ y $[A_p, A_q, A_r]^T$ son desconocidas, variantes en el tiempo y acotadas.

También se define $\eta = [\psi, \theta, \phi]$, donde se tiene que:

$$\omega = W_\eta \dot{\eta} \tag{3.10}$$

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

Es evidente que W_η es no singular en $\frac{-\pi}{2} < \theta < \frac{\pi}{2}$.

3.2.1. Modelo del helicóptero de cuatro rotores simplificado

Supóngase ahora que sólo se permiten pequeñas variaciones en los ángulos de pitch y roll; esto es, $\theta \approx 0$ y $\phi \approx 0$, por lo tanto (3.10) y la matriz R en (3.9) se simplifica de la siguiente manera:

$$\omega = [\dot{\psi}, \dot{\theta}, \dot{\phi}]^T = \dot{\beta}^T \tag{3.11}$$

$$W_\eta = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

$$R = \begin{bmatrix} c\psi & s\psi & 0 \\ -s\psi & c\psi & 0 \\ s\psi s\phi + c\psi s\theta c\phi & -c\psi s\phi + s\psi s\theta c\phi & c\theta c\phi \end{bmatrix} \tag{3.12}$$

En consecuencia, si se eliminan todas las perturbaciones del viento, entonces el modelo (3.6), (3.7) se convierte en:

$$m\ddot{\psi} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} mg + F_\xi \tag{3.13}$$

$$J\dot{\omega} = -\dot{\beta} \times J\dot{\beta} + \tau \tag{3.14}$$

$$\hat{F} = \begin{bmatrix} 0 \\ 0 \\ -u \end{bmatrix}; F_\xi = R^{-1}\hat{F} \tag{3.15}$$

Usando la definición del producto vectorial así como también el hecho de que $R^{-1} = R^T$,

se tiene que el modelo se transforma en:

$$m\ddot{\xi} = \begin{bmatrix} -u(s\psi s\phi + c\psi s\theta c\phi) \\ -u(-c\psi s\phi + s\psi s\theta c\phi) \\ mg - uc\theta c\phi \end{bmatrix} \tag{3.16}$$

$$J\ddot{\beta} = -\begin{bmatrix} \dot{\theta}\dot{\psi}J_{zB} - \dot{\theta}\dot{\psi}J_{yB} \\ -(\dot{\phi}\dot{\psi}J_{zB} - \dot{\phi}\dot{\psi}J_{xB}) \\ \dot{\phi}\dot{\theta}J_{yB} - \dot{\phi}\dot{\theta}J_{xB} \end{bmatrix} + \tau \tag{3.17}$$

Donde se define a $J = diagJ_{xB}, J_{yB}, J_{zB}$ como la matriz de inercia y sus componentes J_{xB},J_{yB} y J_{zB} representan la inercia de el cuatrirotor medida alrededos de los ejes del marco del cuerpo.

Capítulo 4

Controlador del helicóptero de cuatro rotores

En este capítulo se describirá el controlador implementado en el prototipo experimental de helicóptero de cuatro rotores. El control del helicóptero cuatrirotor consiste en una serie de 6 controladores PD: un controlador de altura, tres controladores más para la estabilización y dos controladores más para el posicionamiento del vehículo en el plano $x - y$ cartesiano. En la última parte se presenta el desempeño del controlador bajo simulación en el software MATLAB.

4.1. Diseño del controlador

Considere el modelo (3.35) y (3.36) en lazo cerrado con la ley de control:

$$u = \frac{r_1}{cos(\theta)cos(\phi)} \tag{4.1}$$

donde:

$$r_1 = mg - m\ddot{z}_d + k_{dz}(\dot{z} - \dot{z}_d) + k_{pz}(z - z_d) \tag{4.2}$$

$$\tau = J\ddot{\beta}_d + k_P\tilde{\beta} + k_D\dot{\tilde{\beta}}, \tag{4.3}$$

Definiendo a $\tilde{\beta}$ como una matriz del error de los ángulos de orientación, así:

$$\tilde{\beta} = \begin{bmatrix} \phi_d - \phi \\ \theta_d - \theta \\ \psi_d - \psi \end{bmatrix} \tag{4.4}$$

Y las ganancias del controlador se definen como sigue:

$$k_P = diag\{k_{p\phi}, k_{p\theta}, k_{p\psi}\}$$
$$K_D = diag\{k_{d\phi}, k_{d\theta}, k_{d\psi}\}$$

Donde los ángulos de orientación deseados dependen de la posición, como sigue:

$$\begin{bmatrix}\phi_d \\ \theta_d\end{bmatrix} = -\frac{1}{r_1}\begin{bmatrix}s\psi & -c\psi \\ c\psi & s\psi\end{bmatrix}\begin{bmatrix}u_x \\ u_y\end{bmatrix} \tag{4.5}$$

Definiendo:

$$u_x = m\ddot{x}_d + k_{dx}\dot{\tilde{x}} + k_{px}\tilde{x} \tag{4.6}$$
$$u_y = m\ddot{y}_d + k_{dy}\dot{\tilde{y}} + k_{py}\tilde{y} \tag{4.7}$$

Definiendo el error de posición como:

$$\tilde{x} = x_d - x$$
$$\tilde{y} = y_d - y$$

Si se eligen todas las ganancias del controlador como positivas, entonces el origen es exponencialmente estable y el error de seguimiento $(\tilde{x}, \tilde{y}, (z_d - z), \tilde{\beta}^T) = (0, 0, 0, 0, 0, 0)$ convergerá exponencialmente a cero.

Prueba

Reemplazando (4.1) y (4.3) en (3.35), se tiene que:

$$\begin{bmatrix}m\ddot{x} \\ m\ddot{y} \\ m\ddot{z}\end{bmatrix} = \begin{bmatrix}-r_1(s\psi\frac{s\phi}{c\theta c\phi} + c\psi\frac{s\theta c\phi}{c\theta c\phi}) \\ -r_1(-c\psi\frac{s\phi}{c\theta c\phi} + s\psi\frac{s\theta c\phi}{c\theta c\phi}) \\ m\ddot{z}_d - k_{dz}(\dot{z} - \dot{z}_d) - k_{pz}(z - z_d)\end{bmatrix}$$

Renonmbrando a la tangente $tg(\zeta) = \frac{s\zeta}{c\zeta} \approx \zeta$ y $c\zeta \approx 1$ si $\zeta \approx 0$, se puede escribir, entonces:

$$\begin{bmatrix}m\ddot{x} \\ m\ddot{y} \\ m\ddot{z}\end{bmatrix} = \begin{bmatrix}-r_1(\phi s\psi + \theta c\psi) \\ -r_1(-\phi c\psi + \theta s\psi) \\ m\ddot{z}_d - k_{dz}(\dot{z} - \dot{z}_d) - k_{pz}(z - z_d)\end{bmatrix}$$

O bien:

$$\begin{bmatrix} m\ddot{x} \\ m\ddot{y} \\ m\ddot{z} \end{bmatrix} = \begin{bmatrix} -r_1[(\phi-\phi_d)s\psi + (\theta-\theta_d)c\psi] + u_x \\ -r_1[-(\phi-\phi_d)c\psi + (\theta-\theta_d)s\psi] + u_y \\ m\ddot{z}_d - k_{dz}(\dot{z}-\dot{z}_d) - k_{pz}(z-z_d) \end{bmatrix} \tag{4.8}$$

Donde se define:

$$u_x = -r_1(\phi_d s\psi + \theta_d c\psi)$$
$$u_y = -r_1(\phi_d c\psi + \theta_d s\psi)$$

Nótese que de esta expresion se puede obtener (4.5). Ya que (4.6),(**??**) también son satisfechas, entonces (4.8) es equivalente a:

$$\begin{bmatrix} m\ddot{\tilde{x}} + k_{dx}\dot{\tilde{x}} + k_{px}\tilde{x} \\ m\ddot{\tilde{y}} + k_{dy}\dot{\tilde{y}} + k_{py}\tilde{y} \\ m\ddot{\tilde{z}} + k_{dz}\dot{\tilde{z}} + k_{pz}\tilde{z} \end{bmatrix} = \begin{bmatrix} -r_1[(\phi-\phi_d)s\psi + (\theta-\theta_d)c\psi] \\ -r_1[-(\phi-\phi_d)c\psi + (\theta-\theta_d)s\psi] \\ 0 \end{bmatrix} \tag{4.9}$$

Donde $\tilde{z} = z_d - z$. Por el otro lado, reemplazando (4.3) en (3.36) y eliminando los términos de segundo orden (debido a que se asumió que $\dot{\phi} \approx \dot{\theta} \approx 0$), se obtiene que:

$$J\ddot{\tilde{\beta}} + k_D\dot{\tilde{\beta}} + k_P\tilde{\beta} = 0 \tag{4.10}$$

En consecuencia, la dinámica en lazo cerrado está dado por (4.5), (4.9) y (4.10). El único punto de equilibrio es $(\tilde{x}, \tilde{y}, \tilde{z}, \tilde{\beta}^T) = (0,0,0,0,0,0)$. Todas las ganancias del controlador son positivas. Nótese que la dinámica en lazo cerrado es lineal, compuesta por ecuaciones diferenciales de segundo orden, excepto las primeras dos ecuaciones en (4.9), están desacopladas. Esto siginifica que $\|(\tilde{\beta}(t)^T, \dot{\tilde{\beta}}(t)^T)\|$ y $\|(\tilde{z}(t), \dot{\tilde{z}}(t))\|$ decrece exponencialmente a cero. Más aún, ya que la perturbación está en términos:

$$m\ddot{\tilde{x}} + k_{dx}\dot{\tilde{x}} + k_{px}\tilde{x} = -r_1[(\phi-\phi_d)s\psi + (\theta-\theta_d)c\psi]$$
$$m\ddot{\tilde{y}} + k_{dy}\dot{\tilde{y}} + k_{py}\tilde{y} = -r_1[-(\phi-\phi_d)c\psi + (\theta-\theta_d)s\psi]$$

Pueden ser acotodas por una funcion cuadrática y una lineal de $\|(\tilde{\beta}(t)^T, \dot{\tilde{\beta}}(t)^T)\|$ y $\|(\tilde{z}(t), \dot{\tilde{z}}(t))\|$ (también acotadas), entonces se concluye que $\|[\tilde{x}, \dot{\tilde{x}}, \tilde{y}, \dot{\tilde{y}}]\|$ puede ser acotada y converge a una bola de radio dependiente de $\|(\tilde{\beta}(t)^T, \dot{\tilde{\beta}}(t)^T)\|$ y $\|(\tilde{z}(t), \dot{\tilde{z}}(t))\|$. Debido a que la norma converge exponencialmente a cero, se concluye que $\|[\tilde{x}, \dot{\tilde{x}}, \tilde{y}, \dot{\tilde{y}}]\|$ tambiém converge a cero exponencialmente.

Observación 1. Aunque u y τ son las entradas de control para el modelo (3.35),(3.36),

es de práctica importancia encontrar una expresión la cual permitirá calcular los comandos para cada motor. De acuerdo a la sección 1,las velocidades $\omega_{c1}, \omega_{c2}, \omega_{c3}, \omega_{c4}$ al mando de cada motor de pueden calcular a partir de:

$$\begin{bmatrix} k_1 & k_1 & k_1 & k_1 \\ -k_1 l & 0 & k_1 l & 0 \\ 0 & k_1 l & 0 & -k_1 l \\ k_2 & -k_2 & k_2 & -k_2 \end{bmatrix} \begin{bmatrix} \omega_{c_1}^2 \\ \omega_{c_2}^2 \\ \omega_{c_3}^2 \\ \omega_{c_4}^2 \end{bmatrix} = \begin{bmatrix} u \\ \tau_{xB} \\ \tau_{yB} \\ \tau_{zB} \end{bmatrix} \tag{4.11}$$

Donde, renombramos a $\tau = [\tau_{xB}, \tau_{yB}, \tau_{zB}]^T$, ya que la matriz en esta expresión es no-singular. Estas velocidades son alcazadas por cada motor usando un lazo interno de velocidad con un PI para cada motor.

4.2. Simulación del controlador

En esta sección se presentan las simulaciones correspondientes al controlador PD, la simulación se realizó bajo el software de MATLAB.

En la Figura 4.1 se muestra el desempeño del controlador para estabilizar los ángulos de orientación del helicóptero de cuatro rotores. Las condiciones iniciales son $\psi(0) = 0.5[rad]$, $\theta(0) = -0.3[rad]$ y $\phi(0) = 0.3[rad]$. Las ganancias del controlador son las siguientes: Para el ángulo roll, $k_{proll} = 30$ y $k_{droll} = 0.1$. Para el ángulo pitch, $k_{ppitch} = 40$ y $k_{dpitch} = 0.15$. Finalmente, para el ángulo yaw, $k_{pyaw} = 25$ y $k_{dyaw} = 0.31$.

En la Figura 4.2 se muestra la acción del controlador para estabilizar los ángulos de orientación del helicóptero de cuatro rotores. En la gráfica se aprecia la introducción de una señal de referencia tipo escalón sobre el ángulo yaw, se observa que el helicóptero sigue el giro predeterminado con un error ínfimo de seguimiento. Los ángulos de roll y pitch se mantienen en un valor cercano de cero. Las condiciones iniciales y las ganancias se mantienen respecto a la simulación mostrada en el párrafo anterior.

De la Figura 4.3 se mantiene el control estabilizador sobre los ángulos de orientación y además se introduce el controlador sobre la altura z. Se introdujo una referencia $z_d = 4[m]$, en la figura se observa la acción del controlador sobre dicho eje. Mantiene los ángulos de orientación cerca del cero. Tanto las condiciones iniciales como las ganancias del controlador de los ángulos Euler se mantienen respecto a las simulaciones anteriores. La condiciones iniciales para el controlador de altura se define como $z(0) = 0$ y las ganancias son $k_{pz} = 20$ y $k_{dz} = 6$.

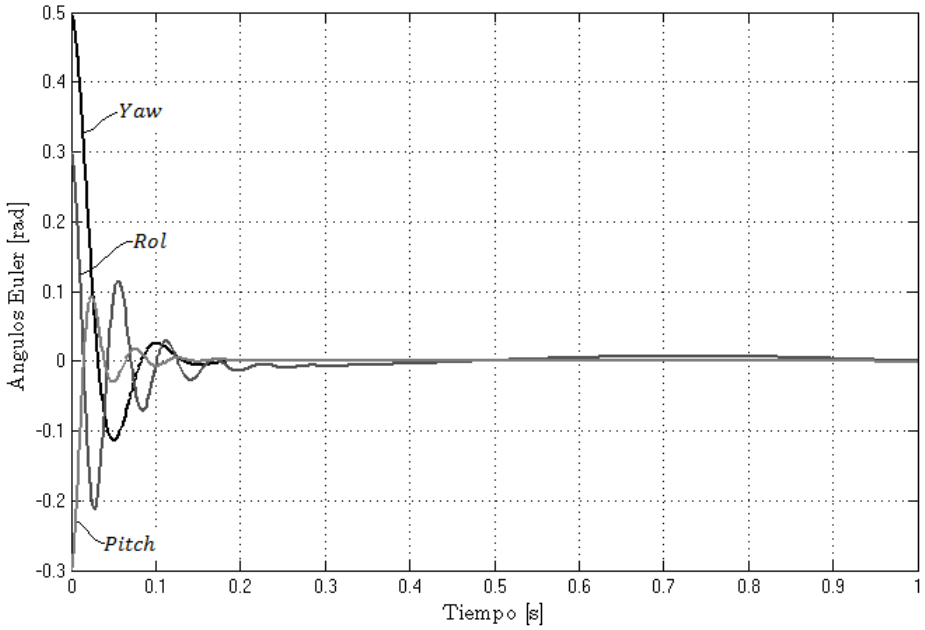

Fig. 4.1. Simulación del controlador de estabilización de los ángulos de Euler.

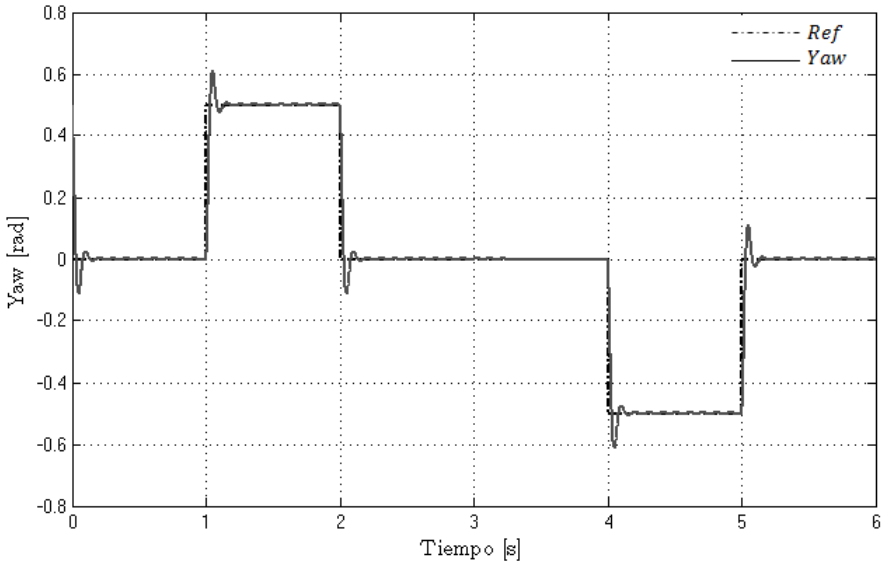

Fig. 4.2. Desempeño del controlador Yaw. Seguimiento de una referencia tipo escalón.

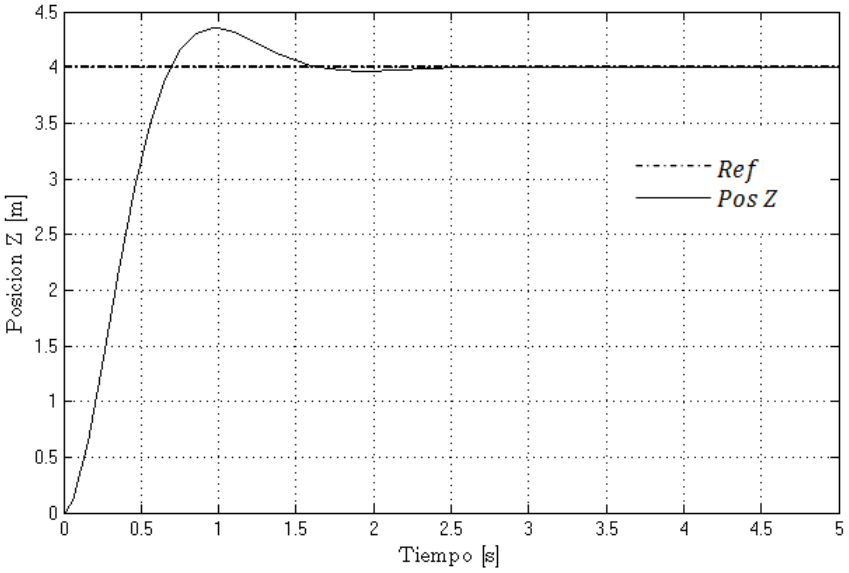

Fig. 4.3. Simulación del controlador de altura con $z_d = 4[m]$

Capítulo 5

Metodología

El controlador mostrado en la sección anterior requiere el conocimiento de los seis estados del sistema; las posiciones cartesianas (x, y, z) y las tres posiciones de orientación (ϕ, θ, ψ), además de los parámetros mostrados por la relación (4.11). Debido a que la electrónica incluída en el helicóptero *Draganfly* no permitía la modificación y/o extracción de datos del posicionamiento del helicóptero para su posterior aplicación al nuevo controlador, se decidió prescindir de dicha electrónica y construir un sistema electrónico que cumpliera con los requeriméntos mínimos del controlador propuesto.

Las partes principales que conforman el sistema se observan en la Figura 5.1. Consta de tres sensores infrarrojos serie *2Y0A02* del fabricante *Sharp* para medir las posiciones en el plano cartesiano (x, y, z) y el sensor IMU *3DM-GX1* de la compañía *MicroStrain* para medir los ángulos Euler (ϕ, θ, ψ). El procesador central, es un DSC (Controlador digital de señales, por sus siglas en inglés, Digital Signal Controller) *DSPIC33FJ64GP802* del fabricante *Microchip*, éste se encarga de adquirir las señales de todos los sensores, acondicionarlas, incluirlas en el procesamiento del algoritmo de control y envíar las señales de control a los actuadores. La etapa de potencia consiste en arreglos con MOSFET configurados en corte y saturación. Asímismo cuenta con una etapa de aislamiento la cual divide la etapa de potencia y la etapa digital. Finalmente, la parte de energía consta con dos baterías recargables Li-Po del fabricante *Thunder Power RC*, baterías de 3 celdas, con 11.1 V en su salida y una corriente de $1350mAh$ utilizadas para alimentar a todo el sistema.

En la primera parte de este capítulo se describirá el circuitería electrónica utilizada en esta tesis, se justificará la utilización de cada circuito utilizado. Después se explicarán algunas consideraciones específicas en cuanto a la fabricación del hardware. Posteriormente, se detallarán algunos conceptos de la programación en el DSC. Luego, se precisará una descripción respecto a la comunicación inalámbrica, donde se explicará en forma específica que se probó tanto el controlador en un sistema embebido (Tiempo de muestreo, $T_s \approx 4ms$ como en un

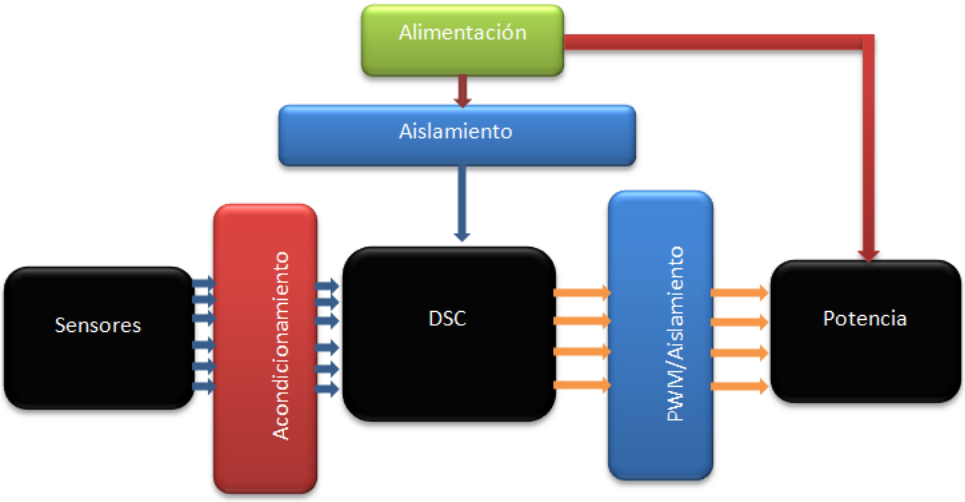

Fig. 5.1. Síntesis del sistema para el helicóptero cuatrirotor.

sistema controlado vía radiofrecuencia ($T_s \approx 25ms$). Finalmente, se expondrá la forma en que se obtuvieron los parámetros de la relación (4.11)

5.1. Descripción de la estructura del helicóptero

La estructura utilizada para construir el prototipo es de la serie *Draganfyier V Ti* de la compañía *Draganfly* se muestra en la Figura 5.2. El cuerpo está construido de una mezcla fibra de carbono-nylon lo que le proporciona rigidez a la estructura y lo hace resistente contra deformaciones en caso de impacto. En cada uno de sus brazos cuenta con motores corriente directa de 6V. Éstos cuentan con un juego de engranes fijo sobre su rotor que conectan con un par de hélices plásticas. La estructura tiene un diámetro de $76.20cm$.

La electrónica incluida con el helicóptero fue removida para proporcionar espacio al sensor IMU, la tarjeta de control y las baterías.

5.2. Diseño electrónico

En esta sección se detallarán cada una de las etapas del diseño electrónico: la etapa de acondicionamiento de señal, la etapa de aislamiento entrada-salida, la etapa de potencia, alimentación del sistema y consideraciones en el diseño del PCB.

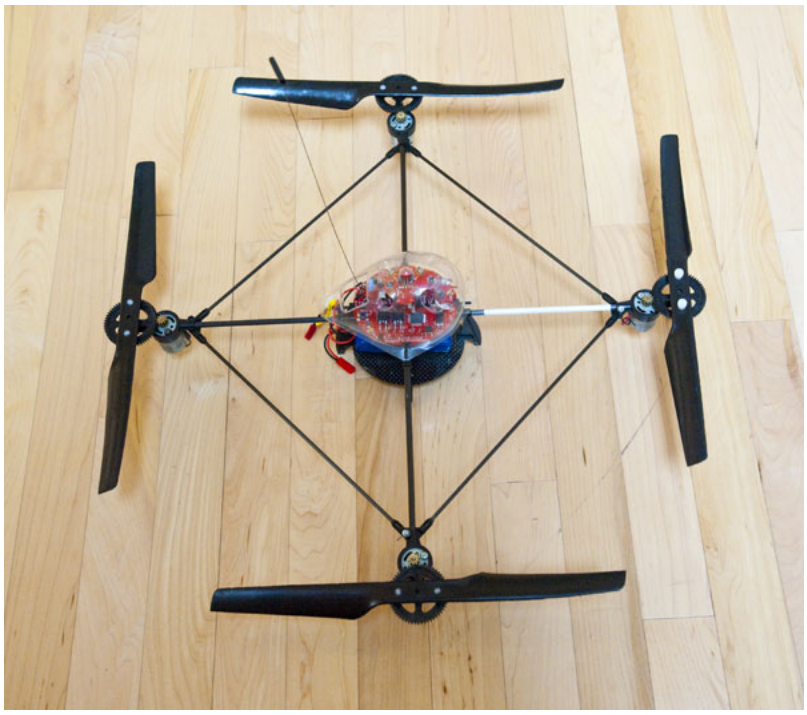

Fig. 5.2. Estructura del helicóptero de cuatro rotores.

5.2.1. Acondicionamiento de señales de los sensores de posicionamiento

El convertidor análogo-digital del DSC tiene un rango de operación entre 0 V y 3.3 V. Puesto que las señales análogas proporcionadas por el sensor de orientación (ϕ, θ, ψ) varían en el rango 0-5 V se requiere un acondicionamiento de la señal de voltaje para evitar un daño permanente en el dispositivo de procesamiento. Pueden implementarse varias soluciones, desde un arreglo pasivo divisor de voltaje hasta diseños con amplificadores operacionales (op. amps.), en este caso se optó por construir un divisor de voltaje pasivo por ser una solución sencilla de implementar y un mínimo uso de área en PCB. La opción del diseño con op. amps. se descartó desde la perspectiva energética pues se requiere al menos una fuente adicional para alimentar al circuito, además del área que ocupan los amplificadores es mayor a la de una arreglo de resistores SMD (Dispositivo de Montaje Superficial, del inglés, Surface Mount Device). El divisor de voltaje se muestra en la Figura 5.3, donde el voltaje de salida V_o se define como sigue:

$$V_{out} = \frac{R_2}{R_1 + R_2} V_{in} \tag{5.1}$$

La selección de los resistores para la divisora de voltaje debe sujetarse a dos condiciones: (1) mantener un valor mínimo para que no afecte el acoplamiento de impedancias y que (2) sean de un valor comercial. En experimentación los valores que se eligieron para esta tarea fueron $R_1 = 511\Omega$ y $R_2 = 1K\Omega$ resistencias de montaje superficial con $1\,\%$ de tolerancia.

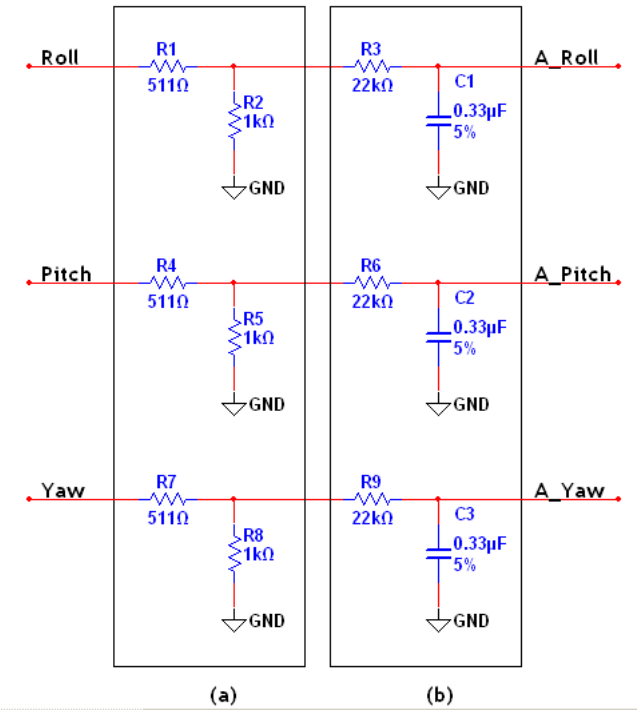

Fig. 5.3. Circuito de acondicionamiento de señal para el sensor de orientación. (a) Divisora de voltaje, (b) Filtro pasa bajas.

Con esos valores, el voltaje a la salida de la divisora es:

$$V_{out} = \frac{1K\Omega}{1K\Omega + 511\Omega} 5V = 3.3V \tag{5.2}$$

Después de que se acondicionó la señal de voltaje, el siguiente paso fue introducir una etapa de filtrado de modo que el ruido eléctrico producido por toda la circuitería afecte lo menos posible a la señal proveniente de los sensores para ello se implementó un filtro pasa bajas pasivo RC de primer orden, nuevamente con el fin de reducir al máximo la superficie del diseño. Los valores de los componentes utilizados en el filtro se eligieron con base en la experimentación siendo estos $R = 22K\Omega$ y $C = 0.33\mu F$ con lo cual se encuentra que la frecuencia de corte f_c es de:

$$f_c = \frac{1}{2\pi RC} = \frac{1}{2\pi(22K\Omega)(0.33\mu F)} \approx 22Hz \tag{5.3}$$

En las figuras 5.4 y 5.5 se muestra la señal proveniente del sensor de orientación con y sin filtrado. Se observa que la etapa de filtrado suprime el ruido de alta frecuencia en la señal del sensor. Esto minimizará los errores en el algoritmo de control.

Los sensores de posición (x, y, z), son dispositivos opto electrónicos cuya salida de voltaje es análoga en un rango de 0 - 2.7 V. Según las especificaciones del fabricante, estos dispositivos pueden medir distancias en un rango entre 15 cm hasta 150 cm. La curva carac-

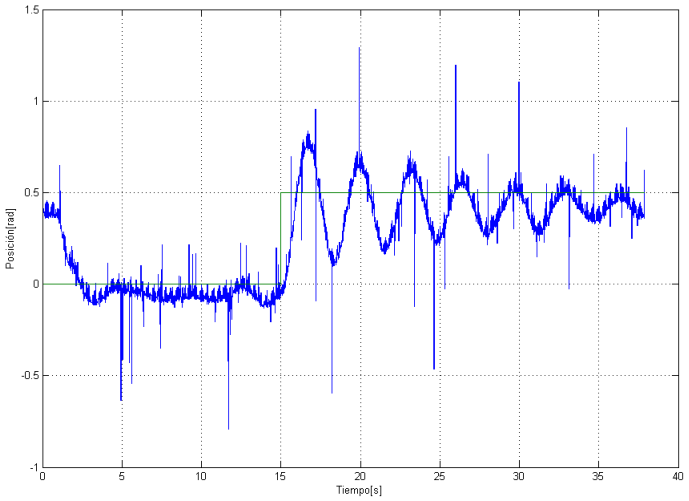

Fig. 5.4. Señal del sensor IMU sin proceso de filtrado.

terística se muestra en la Figura 5.6. Puesto que la salida de los sensores está dentro del rango del convertidor, no se requiere una etapa divisora de voltaje y debido a la baja impedancia de salida del sensor IR, no se modificaron los valores de los componentes que constituyen el filtro RC pasivo (Figura 5.7). Se conserva la misma frecuencia de corte.

5.2.2. Controlador digital de señales y aislamiento eléctrico

El procesador encargado de realizar todos los cálculos requeridos por el controlador es el *dspic33fj64gp802*. Este dispositivo es capaz de operar hasta 40 MIPS (Millones de Instrucciones Por Segundo) si se configura una frecuencia de operación a 80MHz. En la práctica, el máximo valor de cristal externo que se le puede colocar a este controlador es de 40MHz, por lo cual es necesario configurar los registros de oscilador para duplicar la frecuencia de operación. Cuando la máxima frecuencia de operación se ha programado, el DSC operará a su máxima capacidad, esto permitirá disminuir el tiempo de muestreo; esto es, disminuir el tiempo que le toma al dispositivo realizar todas las operaciones necesarias de control. La descripción detallada de programación se muestra en la siguiente sección.

El funcionamiento básico del sistema es el siguiente; primero, el convertidor analógico-digital del DSC transformará las señales análogas provenientes de los sensores de orientación y posición en señales digitales. Segundo, dichas señales digitales son utilizadas por los controladores PD que se requieren para la estabilización del helicóptero. Finalmente, las señales de control son escaladas para ser dirigidas a un PWM de 16 bits de resolución, a su vez, esas señales PWM son enviadas a la etapa de potencia, la cual distribuirá la energía disponible para alimentar a los cuatro rotores en función a su ciclo de carga con el fin de que cumplan

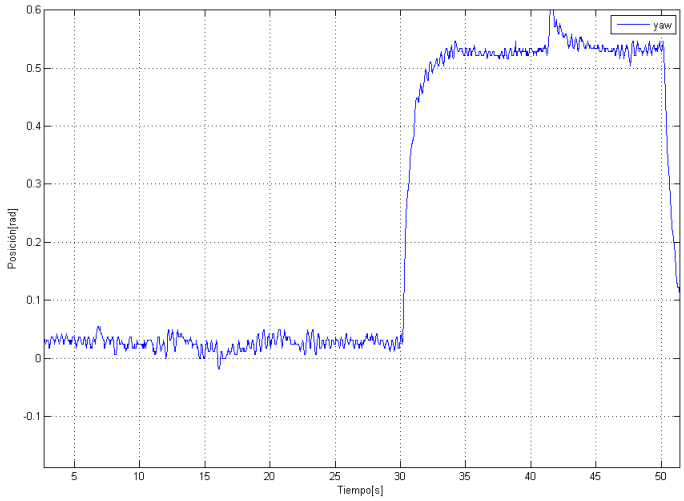

Fig. 5.5. Señal del sensor IMU con filtro pasa bajas.

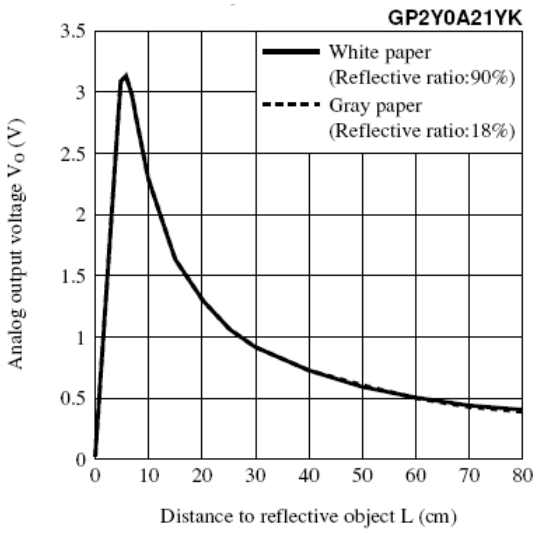

Fig. 5.6. Curva característica del sensor infrarojo.

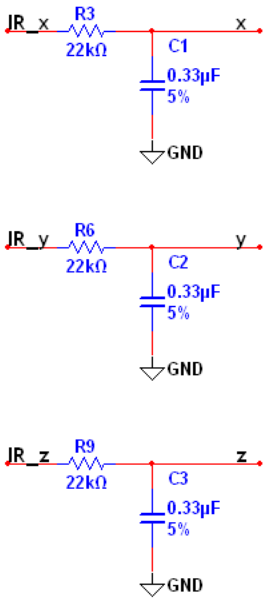

Fig. 5.7. Acondicionamiento de señal para los sensores infrarrojos.

con su función de estabilización (Figura 5.8).

En pruebas experimentales, llevadas a cabo no sólamente con este controlador sino también con el *PIC16F4431* del fabricante *Microchip*, ocurría un efecto no deseado debido a la acción de alta frecuencia de conmutación en la etapa de potencia. Esto provocaba que el procesador entrara en estado de reset, se apagara y se interrumpiera el control del helicóptero. Para resolver esta situación, se introdujeron dos componentes de aislamiento, uno es el convertidor DC/DC aislado *NCS12S1212C* de la compañía *Murata*. En el diseño, este dispositivo funciona como un regulador de voltaje aislado pues separa las tierras de la fuente de entrada de la referencia de la fuente de salida, esto minimiza el ruido por lazos de tierra debido a

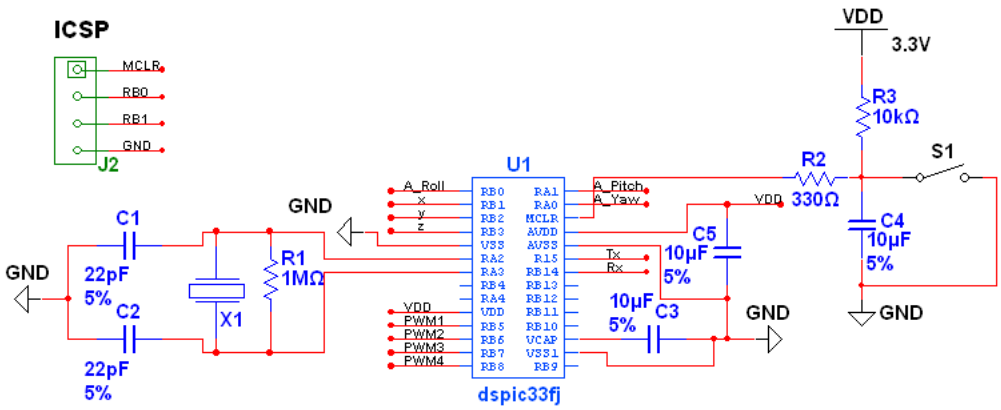

Fig. 5.8. Conexión del DSC. COnexión de programación In-Circuit (Parte superior izquierda).

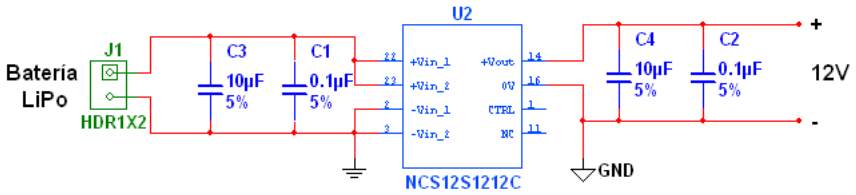

Fig. 5.9. Circuito para el aislamiento con el convertidor *NCS12S1212C*.

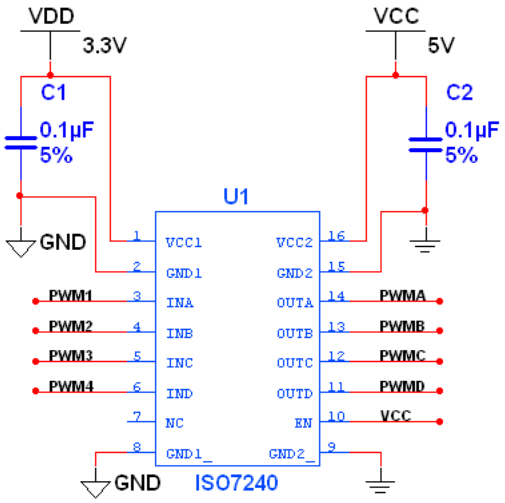

Fig. 5.10. Circuito de aislamiento de salida para el DSC.

la alta frecuencia de conmutación de los componentes de la etapa de potencia. Para mejor comprensión refiérase al diagrama simplificado de la Figura 5.9.

Así como se aisla la entrada de alimentación del DSC, también se debe realizar un aislamiento en la etapa de salida del mismo. En esta parte se utilizó una interfaz de aislamiento digital, el *ISO7240M* del fabricante *Texas Instruments*. Este dispositivo se conectó en las salidas configuradas como PWM del DSC para aislar al sistema digital de la etapa de potencia por la salida. El dispositivo está diseñado para duplicar señales cuadradas en su entrada en una señal de la misma forma pero con diferente voltaje -seleccionado por el usuario según su conveniencia-. En este caso, la salida del DSC enviaba señales PWM de 3.3 V. Después de la etapa de aislamiento, en la salida se reprodujo la misma señal PWM pero con un voltaje de 5 V. El sistema con asilamiento se muestra en la Figura 5.10.

5.2.3. Alimentación y regulación de voltaje

El sistema se alimenta con dos baterías recargables Li-Po de 11.1V de 3 celdas, este tipo de baterías son muy populares en el aeromodelismo por la principal característica de que ofrecen una gran cantidad de corriente por un periodo -razonablemente largo- de tiempo. Estas pilas usadas son capaces de ofrecer una salida de voltaje máximo de 12 V, con una corriente especificada de 1320 mAh. En pruebas experimentales, el tiempo de uso llegó hasta los 10 minutos a potencia máxima del helicóptero con un periodo de carga de la batería muy pequeño. En el diseño, se utilizan dos baterías debido a que, como se verá posteriormente, la intoducción del convertidor DC/DC exige un voltaje mínimo de 9.5 V en su entrada para operar correctamente. Inicialmente, se utilizó sólo una batería para alimentar todo el sistema pero en pruebas experimentales se observaba que unos instantes después de poner en funcionamiento el prototipo, el DSC entraba en reset y las señales de control se disparaban por lo cual los cuatro motores funcionaban a máxima velocidad, provocando que el helicóptero se descontrolara y dejara de funcionar adecuadamente. El diagnóstico realizado mostraba que cuando el controlador requería compensar un error muy grande en un ángulo de orientación, el valor de corriente exigido se incrementa por lo cual el sistema trataba de obtener la mayor corriente disponible para contrarestar el efecto, provocando así que el voltaje que proporcionaba la batería se decrementara hasta 7V lo cual traía como consecuencia que el convertidor DC/DC se apagara y, así mismo, todo el sistema digital dejara de funcionar. Debido a esta situación se tomó la decisión de usar dos baterías, una exclusivamente para alimentar a la etapa de potencia y la otra para suministrar energía al sistema digital. El uso de ambas baterías pareciera ser negativo dado que añade peso al helicóptero pero brinda la posibildad de introducir el convertidor de forma de que se aisla el sistema digital, disminuyendo el impacto del ruido por el switcheo de los MOSFET.

Para la fase de regulación de voltaje se utilizaron tres dispositivos para adaptar los voltajes de alimentación. Para un diseño eficiente se optó por utilizar reguladores del tipo LDO (baja caída de tensión en la salida, del inglés, Low Dropout Voltage), que como su nombre lo indica, tienen una baja caída de voltaje en su salida respecto a la tensión de entrada, generalmente cercano al 1 %. En esta sección se requirieron de tres diferentes circuitos; el primero es el *LD29080DT90R* de *ST Microelectronics*, suministra una salida de voltaje de 9 V con una corriente superior a los 800 mA, éste se empleó para alimentar al sensor IMU. El segundo dispositivo es el *MIC39100-5* de *Micrel* con una salida estable de 5 V y hasta 1 A, dedicado a proveer voltaje a los tres sensores infrarrojos y otro chip más que sirvió como parte del circuito de interfaz de aislamiento (*ISO7240MD*) mencionado en la sección anterior. Finalmente, un *LF33ABD* también de *ST Microelectronics* es utilizado para abastecer de energía al DSC y a los radios inalámbricos *Xbee* pues cuenta con una salida de tensión de 3.3 V. Los

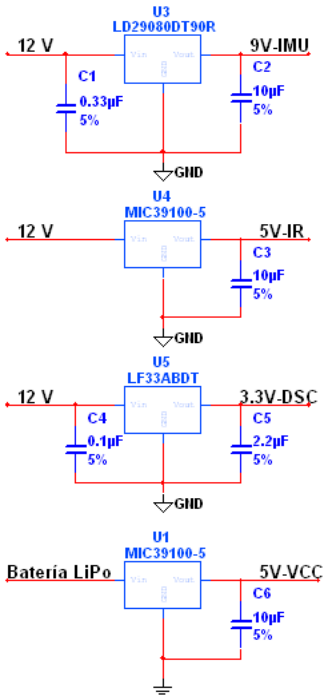

Fig. 5.11. Reguladores de voltaje para adaptación de voltajes de alimentación.

circuitos diseñados para la regulación de energía se muestran el las Figura 5.11.

5.2.4. Etapa de potencia

La etapa de potencia del sistema consiste en un arreglo de MOSFET en configuación corte y saturación. La señal PWM enviada por el DSC se introduce en la compuerta (GATE) del MOSFET como se ve en la Figura 5.12. El voltaje introducido en la compuerta hará que el MOSFET entre en su región activa proporcionando la energía a cada uno de los cuatro motores, calculada por el controlador.

Como se observa en la Figura 5.12, el MOSFET utilizado es el *STPL55NF06L* de *ST Mircroelectronics*, este es un mosfet canal n, de baja resistencia en canal drenaje-fuente R_{DS}, funciona con niveles lógicos de voltaje, lo cual previene el sobrecalentamiento en el MOSFET por un deficiente nivel de tensión en la compuerta, además de que es un dispositivo usado en aplicaciones de control en motores.

En la Figura 5.12 se muestra un arreglo típico para el control de un motor, el diodo usado es de la serie *MURS320* de *MURATA*, este diodo sirve como una protección pues se encarga de disipar la fuerza contraelectromotriz generada por la carga inductiva cuando el MOSFET conmuta al estado de apagado, y así proteger a dicho dispositivo de daños. El diodo elegido es de conmutación de alta frecuencia y puede proteger de picos de corriente inversa de hasta 100 μA.

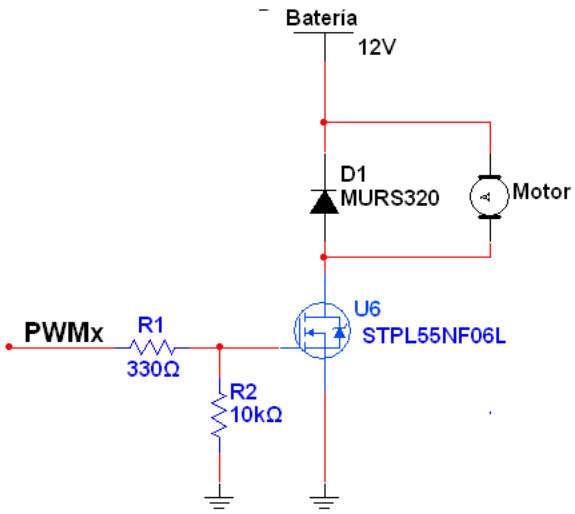

Fig. 5.12. Configuración para conmutación de MOSFET.

De la configuración mostrada, se ve que la resistencia de compuerta de 330Ω está en serie con una resistencia aterrizada con un valor nominal de $10k\Omega$, esto se hace con el fin de que cuando el voltaje de tensión en la compuerta del MOSFET esté en 0V, no se active, es decir, se asegura que la compuerta tenga un nivel bajo siempre que el controlador así se lo indique, evitando comportamientos no deseados, como la activación de un MOSFET por ruido generado por la propia conmutación de los otros MOSFET.

El funcionamiento de esta etapa se explica como sigue: El DSC envía una señal PWM con una frecuencia de $2KHz$ a la compuerta del MOSFET, esto provocará que el dispositivo entre en su región de corte o saturación dependiendo del ciclo de trabajo del PWM. De la situación anterior se desprenden dos casos: (1) Cuando la señal de PWM está en alto, el dispositivo entra en saturación produciendo que el voltaje de la batería -proporcional al ciclo de trabajo del PWM- sea redirigido al motor. (2) En el caso de que la señal PWM está en bajo, causará que el MOSFET entre en la región de corte impediendo que el voltaje proporcionado por la batería se transmita al motor conectado.

5.2.5. Hardware

El circuito electrónico donde se conforman todas las secciones anteriores se muestra en el apéndice E. Fue diseñado en el programa *Proteus 8 Profesional* de *Labcenter Electronics* en conjunto con *Multisim 11* de *National Instruments*. Para el diseño en PCB se uso el software *ARES 8* del mismo fabricante. Algunas recomendaciones y/o especificaciones respecto al diseño del hardware se mencionan a continuación:

a) Se debe tener especial cuidado cuando se hace el diseño del hardware pues muchos com-

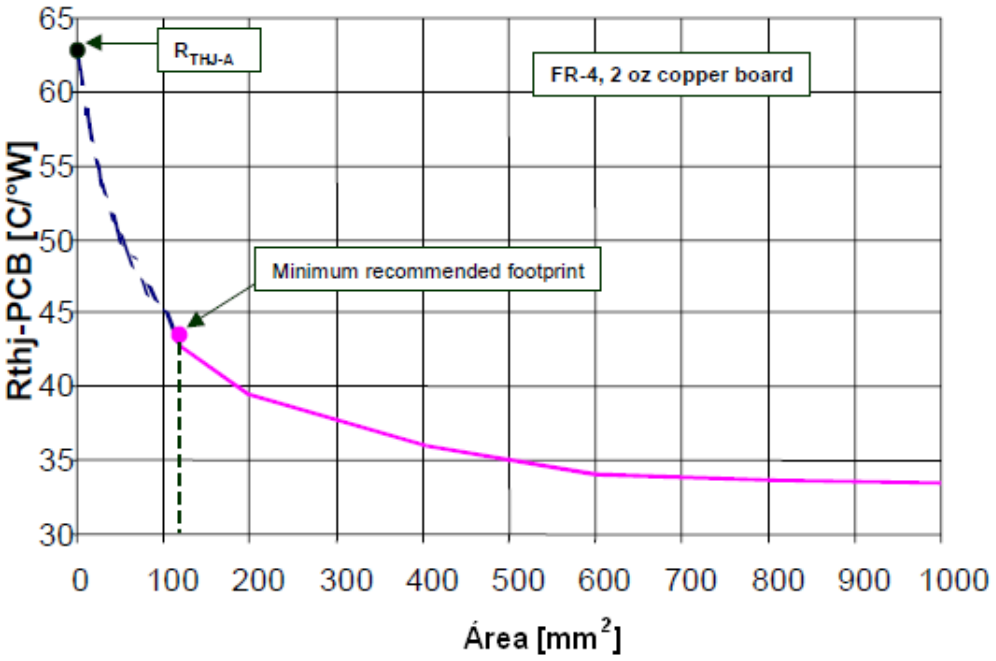

Fig. 5.13. Gráfica de $R_{THJ-PCB}$ contra área del pad drenaje (drain) para el empaquetado D^2PAK.

ponentes no están disponibles en librerías del software para su uso, por lo cual el usuario deberá de diseñar sus propios componentes mediante las especificaciones de las hojas de datos del fabricante, así como diseñar los “pads"para su fabricación en PCB.

b) Tener especial cuidado con algunos de los componentes usados, como por ejemplo, el asilador ISO7240MD. El pin de alimentación debe estar a máximo 3mm de distancia de las fuentes de alimentación, si esto no se respeta el integrado puede tener un mal funcionamiento.

c) Calcular el área de disipación de la etapa de potencia. El tipo de encapsulado del MOSFET usado, el DPAK, tiene la característica de que no se requiere un disipador externo sino que por su misma estructura, la placa PCB funcionará como su propio disipador ahorrando peso en el helicóptero, dependiendo de la potencia máxima que se vaya a disipar es el área que se tiene que resevar para el MOSFET, la hoja de especficaciones del fabricante menciona que hay una relación entre el área de disipación y la potencia mostrada en la Fig 5.13., la relación se muestra como sigue:

$$P_D = \frac{\triangle T}{R_{THJ} - PCB} \tag{5.4}$$

Donde P_D es el potencia de disipación en Watts, $\triangle T$ es la diferencia entre la temperatura máxima que soporta el encapsulado ($T_{MAX} = 175°C$) y la temperatura ambiente

($T_c = 25°C$), su unidad son los grados centrígrados, $R_{THJ} - PCB$ es la relación entre la temperatura y la potencia conforme a un área definida, este valor se determina por medio de la gráfica, sus unidades son el $°C/W$.

Por ejemplo, de la gráfica podemos obtener un gran valor de disipación sin requerir excesiva superficie en el PCB. Si tenemos un área de una pulgada cuadrada ($1in^2$) se obtiene que $R_{THJ} - PCB = 34°C/W$, por lo cual la potencia de disipación calculada sería de:

$$P_D = \frac{150°C}{34°C/W} = 4.4W$$

El área calculada ofrecería un margen de disipación muy holgado para esta aplicación. Es posible determinar áreas más pequeñas según sea el espacio que se tenga disponible en hardware.

d) Especificaciones de hardware del DSC. En la hoja de datos se sugiere el uso de capacitores con bajo ESR (Resistencia Serie Equivalente), así como también una resistencia de al menos $1M\Omega$ en paralelo con el cristal externo lo más cerca de los pines del DSC.

5.3. Programación

En este apartado se describirá el código implementado en el sistema embebido del helicóptero de cuatro rotores, se describirá toda la metodología de programación, se mencionarán algunos inconvenientes que se tuvieron y como se solventaron. El código completo se encuentra en el apéndice A.

Debido a que desde un inicio se pensó en construir un sistema embebido para reducir el peso del helicóptero y teniendo en cuenta de que la sintonización de los 6 controladores iba a consumir tiempo por contar con un gran número de pruebas, lo cual tenía como consecuencia reprogramar el DSC en cada ajuste a las ganancias del controlador, se optó por reprogramar el dispositivo usando el hardware In-Circuit Serial Programming (ICSP) de los programadores comunes de microcontroladores. Esto evita que se desmonte el DSC de su base en la tarjeta para reprogramarse, sino que con un simple conjunto de cables se puede modificar desde el programador. Para utilizar este tipo de hardware se tiene que especificar primero en la cabecera del archivo de programación, de otra manera el dispositivo no será reconocido, se deberá tener especial cuidado en los pines de programación marcados por el fabricante, el dispositivo DSC que se manejo tiene 3 pares de pines por donde se puede utilizar esta manera de programación. En la cabecera del archivo del programa también se debe especificar una serie de fusibles para el correcto funcionamiento. Algunos de los mismos se muestran a

continuación.

```
#include <stdlib.h>
#include <math.h>

#device ADC=12

#FUSES ICSP3      // Canal para programación ICSP
#FUSES NOIESO     // Desactiva el switch del reloj externo/interno
```

En el código anterior se declara un convertidor analógico a digital de 12 bits, se especifica el canal del ISCP por donde se programará el DSC, en este caso es el ICSP3 que corresponden a los pines 14 y 15 (para ICSP1, el 4-5 y para ICSP2, el 21-22). También se declara que no se requirió conmutación del oscilador externo al interno en caso de fallas. Esto se hizo debido a que hubo ocasiones en las que el DSC cambiaba del oscilador externo al oscilador interno sin aviso, trayendo como consecuencia que algunas interrupciones no funcionaran adecuadamente.

Como se describió en la introducción de este capítulo, se experimentaron con dos formas de control, el primero, programando el control en un sistema embebido, es decir, el control lo realiza el DSC y el otro, con transmisión inalámbrica, el control lo calcula una computadora y después lo transmite al helicóptero. En el caso de que se implementara la segunda opción, la adquisición de los seis estados de posición y orientación en el controlador, se requerirá que el helicóptero envíe los datos por medio de RF y dado que los radios utilizados (XBee)son compatibles con el protocolo *RS232*, se precisa de la configuración serial en el DSC, esto se puede ejemplificar como sigue:

```
#pin_select U1TX=PIN_B6
#pin_select U1RX=PIN_B7
#use rs232(UART1, baud=57600, PARITY=N,BITS =8, STOP=1)
```

Los pines de transmisión y recepción del puerto serial se pueden ubicar en cualquier pin reprogramable (consultar hoja de datos). El *baud rate* utilizado es de 57600 baudios por segundo, que es la tasa eficiente experimentalmente encontrada. La siguente tasa de transmisión, con 115200 baud/seg, no mejora la velocidad de transmisión debido a un retardo inherente a este tipo de comunicación.

El tiempo de muestreo se define como el tiempo total que le toma al sistema ejecutar todas las operaciones de control desde que captura los datos de los sensores hasta que el DSC manda las señales de control vía PWM, este tiempo de muestreo se ejecuta a través de una in-

terrupción por TIMER, por ello en el programa principal es necesario configurar los registros para ese TIMER de 16 bits, esto se hace de manera sencilla como se ve acontinuación:

```
setup_timer1(TMR_INTERNAL | TMR_DIV_BY_8);
set_timer1(59285);
```

Estas configuraciones están programadas de acuerdo al tiempo de muestreo en el que se desea que ocurra la interrupción, el valor que se le carga al registro del TIMER (S_{TMR} se relaciona con el tiempo de muestreo deseado T_{sd}, la frecuencia de reloj f_{cy} con la cual trabaja el DSC y con el registro de división del timer T_{div}), lo cual se ve en la siguiente formula:

$$S_{TMR} = (2^{16} - 1) - \frac{T_{sd} f_{cy}}{2T_{div}} \tag{5.5}$$

Aplicándolo a la configuración mostrada con un tiempo de muestreo $T_{sd} = 5ms$:

$$S_{TMR} = (2^{16} - 1) - \frac{(5ms)(20MHz)}{2(8)} = 59285$$

El TIMER se acciona cuando se desborda, quiere decir que como empieza en dicha cuenta, el tiempo que tarda en desbordarse es exactamente igual a 5 ms. Cuando el TIMER se desborda, entra a una interrupción que limpia ciertas banderas y vuelve actualizar su valor al calculado anteriormente. Cabe mencionar, que cuando se tiene el caso en el que se maneja el control por medio de una estación terrestre (computadora que calcula las señales de control), el tiempo de muestreo mínimo se selecciona como $T_s = 25ms$ -encontrado experimentalmente- y se observó que es el tiempo mínimo con el cual las xbee no generan errores de comunicación a una distancia no mayor de 10 metros.

Otro aspecto importante en la configuración del DSC es la de la definición del PWM, puesto que este dispositivo no tiene un modulo específico para PWM como otros dispositivo de la misma familia. Se requiere de unas configuraciones extra para simular la función, el PWM en este DSC se genera con comparación con un TIMER de 16 bits. Para configurarlo, primero se tienen que elegir los pines a usar como PWM, como el prototipo cuenta con cuatro motores, se requieren cuatro salidas, la especificación se hace con las siguientes líneas:

```
#pin_select OC1=PIN_B15
#pin_select OC2=PIN_B13
#pin_select OC3=PIN_B12
#pin_select OC4=PIN_B14
```

Después se precisa de la configuración de comparación, esto es:

```
setup_compare(1,COMPARE_PWM | COMPARE_TIMER2 ); //INICIALIZACION
   DEL PWM1
setup_compare(2,COMPARE_PWM | COMPARE_TIMER2 ); //INICIALIZACION
   DEL PWM2
setup_compare(3,COMPARE_PWM | COMPARE_TIMER2 ); //INICIALIZACION
   DEL PWM3
setup_compare(4,COMPARE_PWM | COMPARE_TIMER2 ); //INICIALIZACION
   DEL PWM4
```

Para seleccionar la frecuencia del PWM se utiliza la siguiente instrucción:

```
setup_timer2(mode,period);
```

En el valor *modo* se programan los diferentes parámetros que se requieren para hacer funcionar al TIMER (Ver ayuda del compilador CCS). El *periodo* se refiere a un valor relacionado con la frecuencia del PWM, para configurar correctamente estos parámetros es necesario conocer la expresión general para configurar el PWM:

$$PWM_{period} = 2[(PRy) + 1]T_{cy} * (Time\_Preescale\_Value) \tag{5.6}$$

Donde el periodo PWM_{period} se calcula con la relación del valor del registro del periodo PRy, el periodo del reloj usado por el DSC calculado como $T_{cy} = 1/f_{cy}$, y el $Time\_Preescale\_Value$ se refiere al valor del registro del preescaler perteneciente al TIMER. Para determinar la frecuencia, se tiene que:

$$PWM_{frequency} = \frac{1}{PWM_{period}} \tag{5.7}$$

Entonces, a modo de ejemplo, si se requiere una frecuencia del PWM de $1.2KHz$, para ello se necesita elegir los valores de los registros en función de la ecuación (5.6) teniendo cuidado en colocar valores aceptados por la instrucción como se maneja en el manual del compilador, para este ejemplo:

$$PWM_{period} = 2(8415 + 1)\frac{1}{20MHz}(1) \approx 8.33 \times 10^{-4}$$

Obteniendo la frecuencia:

$$PWM_{frequency} = \frac{1}{8.33x10^{-4}} \approx 1.2KHz$$

Con esta información se le pueden dar los argumentos requeridos por la función como se ve

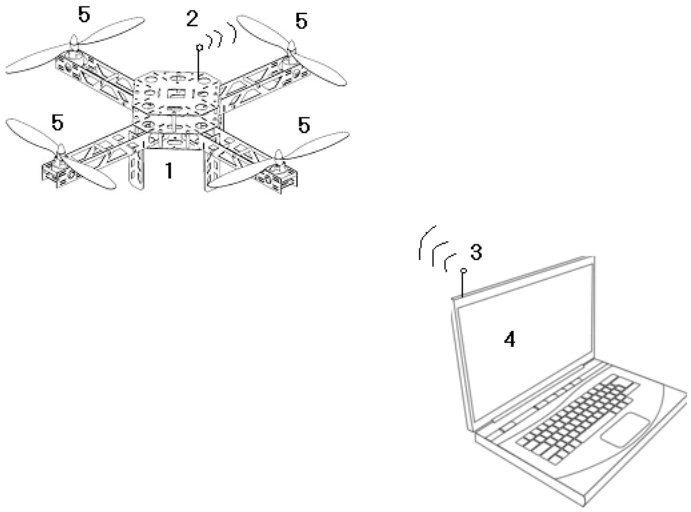

Fig. 5.14. Operación del cuatrirotor con estación en tierra.

acontinuación:

```
setup_timer2(TMR_INTERNAL | TMR_DIV_BY_1,8415);
```

Hasta este punto se conforma toda la parte de configuración del sistema, el siguiente paso sería introducir el controlador y realizar los ajustes pertinentes. En el apéndice A se puede ver el programa completo para el sistema embebido.

5.4. Descripción del caso del controlador en una estación en tierra (Transmisión RF)

La estación en tierra (computadora) calculará el algoritmo de control como se muestra en la Fig. 5.14. Aquí la tarjeta del helicóptero sólo funcionaría como adqusitora de señal pues recabaría la información de todos los sensores (1), después la transmitiría vía RF (2) hacia la estación terrestre (3), ahí se capturarían los datos de los sensores, se introducirían en el algoritmo de control, el cual calcularía el valor de voltaje en PWM para cada uno de los motores (4), esa información se enviaría nuevamente por RF hacia el helicóptero donde el DSC recibiría los comandos de control y los enviaría mediante PWM (5), controlando la aeronave.

El programa de control se ecribió en C usando el compilador C++Builder versión 6, el programa completo se muestra en el apéndice (C y D) donde se anexan las librerías y los códigos para establecer comunicación serial con el XBee. Las pruebas se realizaron con una

computadora DELL Inspiron 6400 con un procesador Intel Core 2 de 1.83GHz.

Si el lector desea implementar esta metodología, basta con seguir estos pasos:

1. El programa del apéndice A, copiarlo en un archivo con extensión «.cpp», el programa del apéndice B, guardarlo con una extensión de librería «.h» y compilarlo bajo la plataforma *CCS C Compiler*. Este programa fue probado con PCWHD Compiler versión 5.015.

2. El archivo «.hex» generado, grabarlo en el DSC con el programador de su preferencia, en este caso fue grabado con *Master Prog*.

3. Para el archivo del apéndice C, guardarlo como archivo ".cpp". El del apéndice D, como cabecera ".h", compilarlo con *C++ Builder Versión 6*

4. Grabar las configuraciones mostradas en la sección anterior en los radios *XBee* mediante "X-CTU". Si no se cuenta con el programador, éstos se pueden configurar mediante comandos (consultar hoja de datos del fabricante).

5. Conectar los radios en los canales seriales del DSC y de la computadora. En la computadora fijar la velocidad del puerto a 57600 baudios por segundo o la velocidad que se haya configurado.

6. Finalmente, una vez montado el helicóptero, encender el DSC, después correr el programa en la computadora y esperar por comunicación. Cuando se establezca comunicación, encender la etapa de potencia del helicóptero.

El inconveniente de esta metodología como ya se mencionó anteriormente es el incremento del tiempo de muestreo pues aunque el cálculo de control toma un par de milisegundos en realidad todo el retraso se debe al tiempo de latencia entre los dos radios, incrementándose hasta un tiempo confiable de $T_s = 25ms$ pero que a pesar de esto, en pruebas experimentales se logró controlar la posición del aeronave como se verá en la sección de resultados.

5.4.1. Comunicación inalámbrica

El sistema electrónico final del cuatrirotor, es un sistema totalmente embebido, esto quiere decir, que todo el controlador está programado dentro del DSC, como consecuencia, las ganancias de los controladores están definidas y no se pueden modificar hasta que se vuelva a reprogramar el sistema, lo cual dificulta la modificación de las ganancias en la etapa de sintonización del controlador. Por ello, se tomó la decisión de fragmentar el sistema en dos partes, la primera, el DSC funcionaría como una tarjeta de adquisición de señales montada en el helicóptero, adquiriría las lecturas de los seis sensores, para después enviarlas inalámbricamente.

La segunda, sería una estación en tierra con un transmisor-receptor que recibiría las señales enviadas por el helicóptero, las mandaría a una PC, donde se tendría el controlador, éste a su vez, obtendría los cálculos de estabilización y los mandaría nuevamente al helicóptero para que el DSC fijara las señales PWM a cada uno de los motores con el fin de poder controlar a la aeronave. Una desventaja de esta estrategia es que los sistemas inalámbricos tienen un retardo intrínseco en la transmisión de las señales, dificultando el control puesto que esto impactaría directamente sobre el tiempo de muestreo requerido para controlar la aeronave.

La estrategia descrita anteriormente, se aplicó bajo la suposición de que como el helicóptero es de dimensiones grandes, la dinámica del mismo es lenta y por ende es posible que el tiempo de muestreo se pueda extender para que la transmisión inalámbrica pueda llevarse a cabo.

Para la transmisión inalámbrica se usaron dos radios del tipo *XBee S1* del fabricante *MaxStream*, mediante la configuración por el software *X-CTU*. Con dicho software se programaron diferentes parámetros en cada uno de los radios de modo que hubiera una transimisión del tipo serial en modo AT, full duplex, es decir, transmisión bidireccional, cada radio funcionaría como un transmisor-receptor. En la tabla 5.1, se muestran y explican los parámetros comunes que se pueden reconfigurar en los radios *XBee*

Cabe mencionar que no todos los parámetros requieren ser modificados, por ejemplo, los canales de entrada y salida no se modifican excepto si se desea hacer uso del PWM o ADC integrados. Requieren especial atención aquellos parámetros que tengan que ver con la selección de red, seguridad de transmisión, interfaz serial, de diagnóstico y comandos AT. En la Tabla 5.2, se observan las configuraciones mínimas realizadas para los radios que se usaron en la experimentación y en la Figura 5.15 se muestra una captura del programa. Se pueden variar los párametros de ACK para intentar reducir el retraso en la transmisión.

Como ya se mencinó líneas arriba, la aplicación de la estrategia de control a través de sistemas inalámbricos conlleva a un tiempo de retardo intrínseco a la naturaleza del protocolo usado, es por ello que en la siguiente sección se establecerán las bases para la estimación de dicho retardo.

Cálculo del retardo en la transmisión inalámbrica

En esta etapa se realizaron diferentes pruebas de transmisión, concluyendo que el tiempo de muestreo T_s más pequeño que se obtuvo fue de $T_s = 25ms$, aunque teóricamente el tiempo de muestreo puede ser de hasta diez milisegundos, mediante este tipo de comunicación, ninguna configuración probada en los radios permitió modificar el tiempo de muestreo por debajo de los $25ms$. La razón por la cual el tiempo de muestreo alcanzado es de $25ms$ es debido a varios factores, como lo son el tiempo de latencia del protocolo $802.15.4$ usado

Tabla 5.1. Opciones de configuración comunes en *XBee*.

Comando	Significado y uso
CH	Channel: define el canal por el cual se transmitirá la información. Se usa para encontrar un canal limpio o para separar redes de xbee
ID	PAN ID: es el identificador de la red
DL	Destination Low Address: la dirección de destino de la transmisión. Define qué nodo recibe los datos
MY	Source Address: establece la dirección del nodo
NI	Node Identifier: define el nombre de un nodo
SM	Sleep Mode: utilizado para activar la función de bajo consumo
BD	Interface Data Rate: se selecciona el “baud rate"de la transmisión serial
AP	API Enable: el XBee conmuta de modo trasnparente (AT) a una versipin de datos estrecturados donde éstos deben ser manualmente modificados, como por ejemplo, la dirección y el “checksum"
RO	Packetization Timeout: es el tiempo de espera del xbee entre transmisión de caracteres
D0-D8	Establece la función de los pines de entrada y salida del XBee
P0,P1	Configura la función *PWM0* y *PWM1*
RP	Fija el tiempo total para la salida del RSSI
M0,M1	Establece el valor de PWM para las salidas PWM
IR	Sample Rate: el XBee puede ser configurado para que mande datos del sus puertos entrada-salida o del ADC cada cierto perido de tiempo. Sólo con transmisión API
DB	Received Signal Strength: el XBee consulta este registro para devolver el nivel de RSSI del último paquete recibido
EA	ACK Failures: si un paquete se transmite pero no recibe ningún reconocimiendo de que los datos llegaron a su destino, EA se incrementa. El XBee realiza dos intentos antes de mandar mensaje de fallo. Reintentos adicionales se pueden agregar mediante el uso de ajuste de RR
EC	CCA Failures: el protocolo reliza la evaluación del canal libre (CCA), es decir, revisa los niveles de RF antes de transmitir. Si no se puede conseguir una apertura, la parte de diagnóstico fallará y el contador CCA se incrementará
CT	AT Command Timeout: una vez en el modo de comandos, se fija la duración de demora antes de regresar a su funcionamiento normal
GT	Guard Tine: Cuando se conmuta en el modo transparente AT, aquí se define la duración del tiempo de guarda (ausencia de datos antes de la línea de comandos) para que no se realice el cambio de forma accidental
WR	Escribe la configuración de la memoria no volátil
RE	Restaura la configuración predeterminada en la memoria volátil

Tabla 5.2. Parámetros de configuración mínima en *XBee*.

Parámetro	Radio Helicóptero	Radio PC
PANID	9032	9032
DH	0	0
DL	5656	3434
MY	3434	5656
Baud Rate	57600	57600

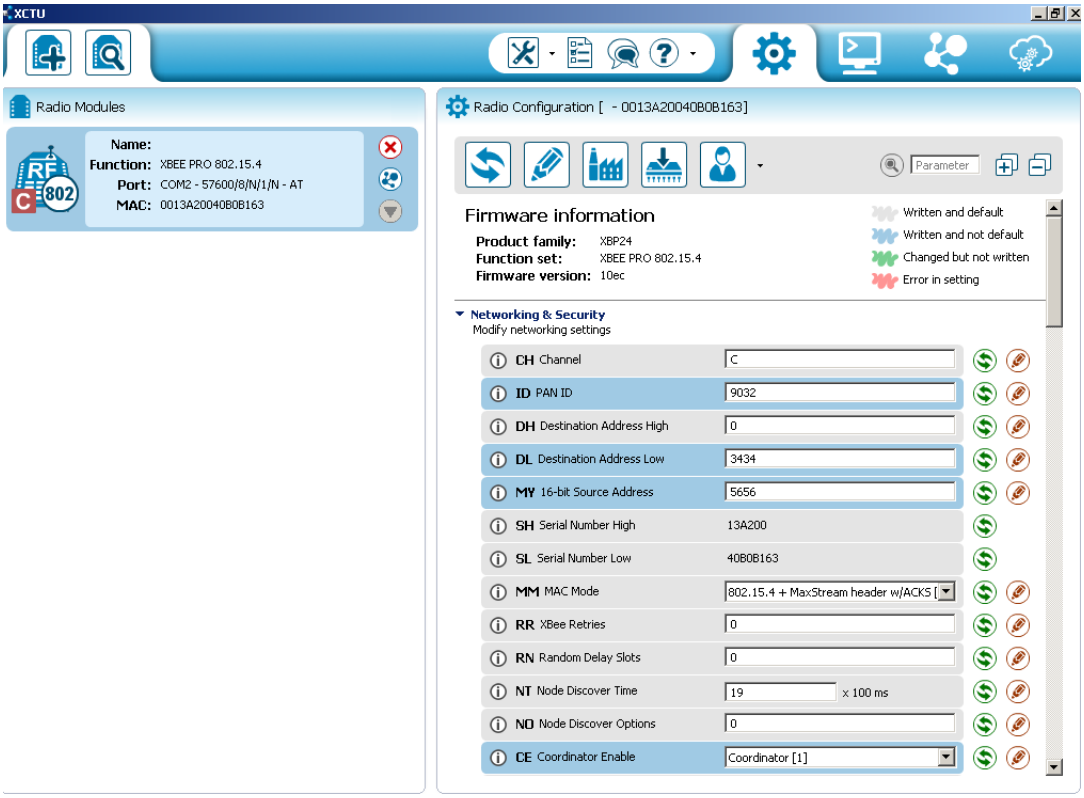

Fig. 5.15. Captura de pantalla de configuración de parámetros en software *X-CTU*.

por el radio, los factores ambientales, entre otros. Para determinar el retraso en el sistema inalámbrico es necesario introducir el concepto de tiempo de latencia y el cálculo del mismo. El tiempo de latencia -en explicación simple- es el tiempo de respuesta; esto es, la medida de tiempo que tarda un dato en ser transmitido y se compone por la suma del tiempo en aire más el tiempo para la función CSMA-CA (del inglés Carrier Sense Multiple Acces- Collision Avoidance) e intentos. La serie 1 del XBee maneja el protocolo de transmisión $802.15.4$ a una frecuencia de $2.4GHz$ con un baud rate máximo de 250 kbps, con lo cual se logra transmitir un bit cada $4\mu s$ o bien un byte cada $32\mu s$, con esta información es posible calcular el tiempo en aire, el cual se define en su versión simplificada como:

$$T_{air}(x) = (13 + x) * 32\mu s \tag{5.8}$$

Donde x es el numero de bytes por transmisión. Entonces, aplicado al sistema del helicóptero, para conocer el número de bytes que se envían en el proceso, se debe tomar en cuenta que si son seis sensores y una resolución de convertidor análogo-digital de 10 bits, por tanto, bajo el protocolo de transmisión serial (8 bits de información y 2 bits de paro) es necesario dividir cada lectura en dos bytes, por lo que se tendrían 12 bytes por lectura de todos los sensores más un byte de reconocimiento, por lo cual se tuvieron que enviar 13 bytes desde el transmisor en el helicóptero. Aplicando la formula (5.4) para calcular el tiempo en aire se tiene que:

$$T_{air}(13) = (13 + 13) * 32\mu s = 0.832ms \tag{5.9}$$

Para calcular el tiempo total de latencia se requiere también el cálculo del tiempo del CSMA-CA, este concepto se refiere, básicamente, a que antes de que un radio comience a transmitir en el aire, éste deberá detectar algún canal portador (llamado CCA del inglés, Clear Channel Assessment) para asegurarse de que no haya alguna interferencia. Si se detecta una fuerte interferencia en el canal que impida la transmisión, entonces, se llevará a cabo un retardo aleatorio (tiempo de retroceso / espera) y se intentará nuevamente con otro CCA. Para realizar un cálculo de un retardo introducido por este procedimiento es necesario analizar el algoritmo del CSMA-CA del protocolo de comunicación IEEE $802.15.4 - 2006$. Para una referencia rápida, a continuación se explica en resumen los pasos más básicos que se lleva el CSMA-CA:

1. Ejecutar un retraso aleatorio T_R.
2. Ejecutar la acción CCA.
3. Transmitir dato si el CCA se realizó con éxito. Si no hay un canal disponoble entonces se repiten los pasos 1 a 3 por lo menos 4 veces más.

4. Si se realiza con éxito hay transmisión. Si la comunicación es unicast:

   a. Espere por ACK del modulo destino (del otro radio)

   b. Si el ACK es recibido. Repita el paso 1-4 al menos tres veces más.

Entonces, para determinar el tiempo total que dura esta etapa, referente al módulo xbee, se tiene que:

1. Ejecutar un retraso aleatorio. Para este paso en las xbee la función de retraso está definida como $(0 : 2^{BE-1}) * 0.320ms$, donde BE empieza con el valor definido por RN y se incrementa cada vez (hasta el valor máximo de 5). El parámetro RN puede ser reconfigurado por el usuario.

2. Ejecutar la acción CCA. Este paso siempre toma 0.128 ms.

3. Transmitir dato si el CCA se realizó con éxito. Si no hay un canal disponoble entonces se repiten los pasos 1 a 3 por lo menos 4 veces más.

4. Si se realiza con éxito hay transmisión. Si la comunicación es unicast:

   a. Espere por ACK del modulo destino. Este paso toma arriba de 0.864 ms.

   b. Si el ACK es recibido. nó hay cálculos en este paso.

Con esta información ya se puede calcular el tiempo de latencia que tienen los modulos serie 1. Se analizarán los dos casos, el ideal o mejor caso y el peor de los casos que es cuando la transmisión se realiza varias veces.

Mejor Caso. Transmitir 13 bytes, $RN = 0$:

$$\begin{aligned} T_R &= (0 : 2^{0-1}) * 0.320 = 0ms \\ CCA &= 0.128ms \\ T_{air}(13) &= 0.832ms \\ T_{total}(13) &= 0 + 0.128 + 0.832 = 0.960ms \end{aligned}$$

Idealmente la latencia en los radios sería menos de un milisegundo. Para el peor caso, se tiene que a manera de ejemplo: Transmisión de 1 byte con $R_N = 0$

$$T_R = 0; CCA = 0.128ms$$

Asumiendo que este CCA no encontró canal, se vuelve al paso anterior y reintenta:

$$\begin{aligned} T_R &= (0:1) * 0.320 = 0.320ms; \\ CCA &= 0.128ms \\ T_R &= (0:3) * 0.320 = 0.960ms; \\ CCA &= 0.128ms \\ T_R &= (0:7) * 0.320 = 2.240ms; \\ CCA &= 0.128ms \\ T_R &= (0:15) * 0.320 = 4.800ms; \\ CCA &= 0.128ms \end{aligned}$$

El subtotal de tiempo del CSMA-CA es de 8.96 ms, por lo que:

$$\begin{aligned} T_{air}(13) &= 0.832ms \\ T_{total}(13) &= 8.96ms + 0.832ms = 9.792ms \end{aligned}$$

Para cálculos rápidos, se puede generalizar este caso de la siguiente manera:

$$T_{total}(bytes) = 9.376[ms] + 0.032 * bytes[ms] \tag{5.10}$$

Se pueden encontrar más casos al modificar los parámetros de reintento, que podrían incrementar el valor de la latencia dependiendo del número de bytes a transmitir, la distancia entre los radios y algunos factores ambientales. Tomaremos como base el último caso, siendo este el más cercano a la realidad. Entonces tenemos una latencia aproximada de $10ms$, sólo para la transmisión de datos del helicóptero hacia la base en tierra, efectuando los mismos cálculos pero ahora en sentido inverso; es decir, para la transmisión desde la base de tierra hacia el helicóptero, se podrá ver que aunque la cantidad de bytes es menor, son 5 (4 bytes de control y uno de reconocimiento), el tiempo de latencia es similar, pues aplicando la formula generalizada, observamos que:

$$\begin{aligned} T_{air}(13) &= 0.576ms \\ T_{total}(13) &= 8.96ms + 0.576ms = 9.536ms \end{aligned}$$

Se observa que la latencia para mandar los valores de control de la estación al helicóptero le toma al menos $10ms$, esto hace una latencia total de al menos $20ms$, por esta razón, los radios con este protocolo no son recomendables para usarse en prototipos de control, a menos

de que el controlador permita trabajar con ese tiempo de muestreo mínimo.

El prototipo fue probado con esta metodología, encontrando que el tiempo de muestreo mínimo era de 25 ms, los resultados y especificaciones de esta prueba se analizan en la sección de resultados.

5.5. Caracterización de motores

De acuerdo a la estrategia presentada, los controladores PD ofrecen una representación en torques para controlar los motores del helicóptero. Para relacionar los torques con las velocidades angulares de los motores se ha presentado la siguiente relación matricial:

$$\begin{bmatrix} k_1 & k_1 & k_1 & k_1 \\ -k_1 l & 0 & k_1 l & 0 \\ 0 & k_1 l & 0 & -k_1 l \\ k_2 & -k_2 & k_2 & -k_2 \end{bmatrix} \begin{bmatrix} \omega_{c_1}^2 \\ \omega_{c_2}^2 \\ \omega_{c_3}^2 \\ \omega_{c_4}^2 \end{bmatrix} = \begin{bmatrix} u \\ \tau_{xB} \\ \tau_{yB} \\ \tau_{zB} \end{bmatrix}$$

De esta relación se realiza un despeje para encontrar cada una de las relaciones para controlar al helicóptero. Antes de esto, se requiere encontrar el valor de las constantes definidad k_1 y k_2. De acuerdo a la definición de $u_i = k_1 \omega_{ci}^2$, se relaciona directamente a la constante con la fuerza producida por cada motor y la velocidad angular al cuadrado, esto es, hay una estrecha relación entre la velocidad y la fuerza de empuje. Con k_2 se encuentra que la relación $\tau_{M_i} = k_2 \omega_{ci}^2$, se vincula a dicha constante con el par producido por el motor. Con estas premisas se procederá a describir la metodología para el cálculo de estas constantes.

Los materiales usados para la experimentación se enlistan a continuación:

a) Báscula digital con resolución de al menos 5 gramos.

b) Una caja de cualquier material con suficiente espacio para meter las aspas del helicóptero completamente abiertas.

c) Fuente de voltaje variable de mínimo 2 Amperes.

d) 2 Multímetros (En caso de que la fuente esté bien calibrada se puede hacer sin multímetros)

e) Caimanes de grueso calibre.

f) Desarmadores.

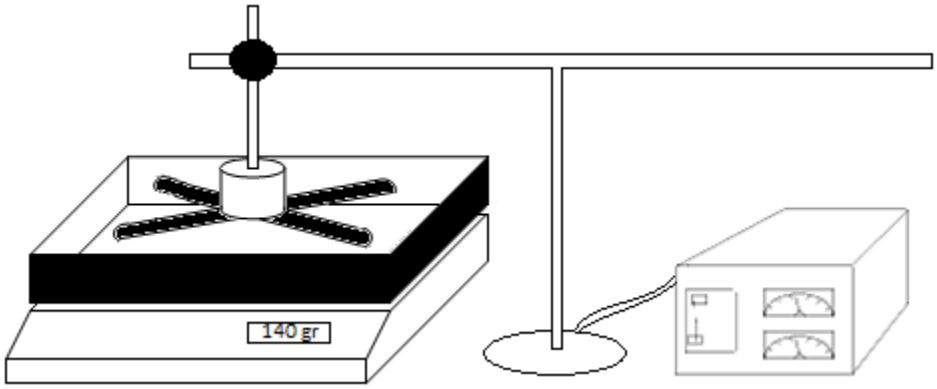

Fig. 5.16. Configuración experimental para k_1.

5.5.1. Medición de constante k_1

Para estimar el valor de la constante k_1, se describe el procedimiento seguido a detalle:

1. Teniendo el prototipo del helicóptero, las aspas de un motor son desmontadas y colocadas en el sentido contrario, con el fin de que la masa de aire que generan las aspas sea empujado en sentido contrario. Volver a atornillar las aspas (Ver Figura 5.16).

2. Colocar la caja encima de la báscula y restar el peso, haciendo uso de la función TARA de la báscula, esto hará que la medición de peso no considere la masa de la báscula.

3. Colocar el motor dentro de la caja para realizar la medición de la masa de aire que rechaza/genera cada motor.

4. Conectar el motor directamente sobre sus terminales de alimentación con la fuente de voltaje regulable, usando caimanes con un calibre grueso, esto evitará que a corrientes grandes los caimanes se calienten y así evitar pérdidas de energía.

5. Conectar un multímetro en serie con la fuente para medir la corriente consumida. De la misma manera conectar un multímetro en paralelo a las terminales del motor para supervisar que el voltaje de entrada al motor.

6. Alimentar el motor desde el nivel más bajo de voltaje, particularmente, este experimento se realizo en una relación de incremento de 0.5 V, por tanto la caracterización se empezó con 0.5 V.

7. Una vez alimentado el motor anotar el peso de la masa de aire producida por las aspas del motor marcado por la báscula.

8. Incrementar el voltaje en 0.5 V y obtener una tabulación de voltaje, corriente y masa de aire desplazada.

9. El experimento se realizó hasta los 8 V de alimentación, sin embargo, con el tipo de motor especificado no se sugiere esto, pues podría darse el caso de que el motor se dañe, de ahora en adelante, el voltaje máximo se mantendrá hasta los 7V, con un consumo máximo de 2 A por cada motor. Nota: Como no se cuenta, a pesar de búsqueda en la compañía del fabricante del motor, con la hoja de especificaciones del motor, no se sabe a con exactitud cual es el voltaje máximo permitido sin dañar el motor, esto se calculo de manera experimental, siguiendo la metodología anterior se aumento gradualmente el voltaje y cuando éste llego a los 8V el motor empezó a calentarse ligeramente, entonces, cuando el motor llegaba a calentarse, detener el procedimiento y tomar ese voltaje como máximo, el consumo en amperes fue de aproximadamente 2 Amperios.

10. Repetir el experimento para cada uno de los cuatro motores del helicóptero.

11. Una vez obtenida la tabulación de voltaje y corriente contra masa, procedemos a la graficación del mismo. Puesto que lo que nos interesa es una constante de motor definida en el controlador donde se refiere a la fuerza que producen cada uno de los i-motores, k_1 es una constante y es la velocidad angular de cada uno de los i-motores, necesitamos encontrar una relación para calcular la constante, por ende, se grafica la relación de voltaje al cuadrado [V^2] contra la masa desplazada de aire [m] multiplicada por la gravedad [g], esto se hace para encontrar una relación V^2-Fuerza, como se puede observar en la Figura 5.17, se ve que los puntos describen casi una línea recta a excepción de algunos puntos, que generalmente son los límites de alimentación, dada su naturaleza casi lineal, se procede a hacer una regresión lineal de los datos, en este caso se realizo en MATLAB mediante el comando *polyfit*, este comando ofrece una aproximación lineal o no lineal de los datos, lo que hace es aproximar los datos a un polinomio, en este caso, es suficiente que haga una aproximación lineal, es decir, un polinomio de grado 1. Una vez hecha la aproximación lineal, el programa genera una ecuación de la forma $y = mx + b$, aquí la constante que interesa es precisamente la pendiente. Esta constante m será el valor aproximado de la constante k_1.

    Regresión lineal:

    $$y = 0.0481x + 0.1357$$

12. El procedimiento se extiende a cada uno de los cuatro motores y se calcula la pendiente de cada una de las rectas aproximadas, en teoría esa constante debería ser igual para cada uno de los cuatro motores, esto no sucede en la práctica pero como se podrá observar, en los datos graficados en el anexo, las constantes difieren poco entre sí, para mejorar la

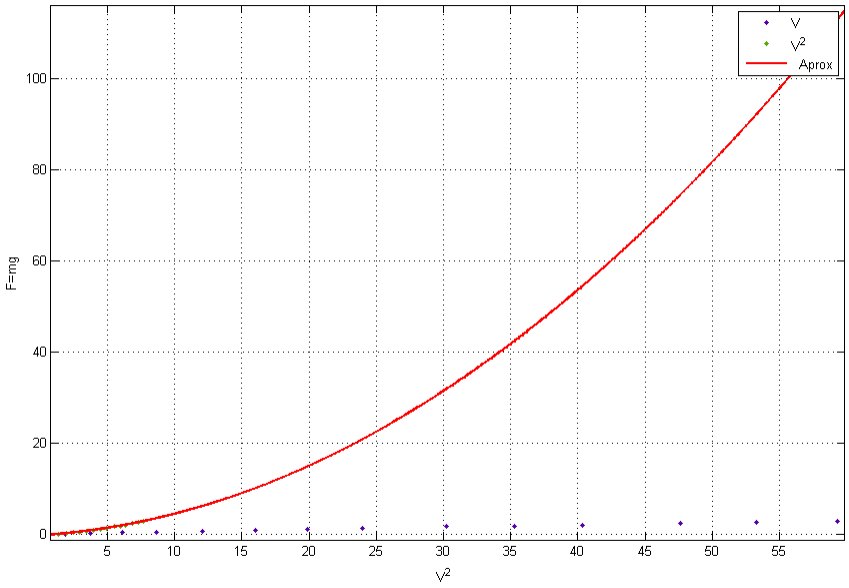

Fig. 5.17. Aproximación cuadrática de F vs V^2.

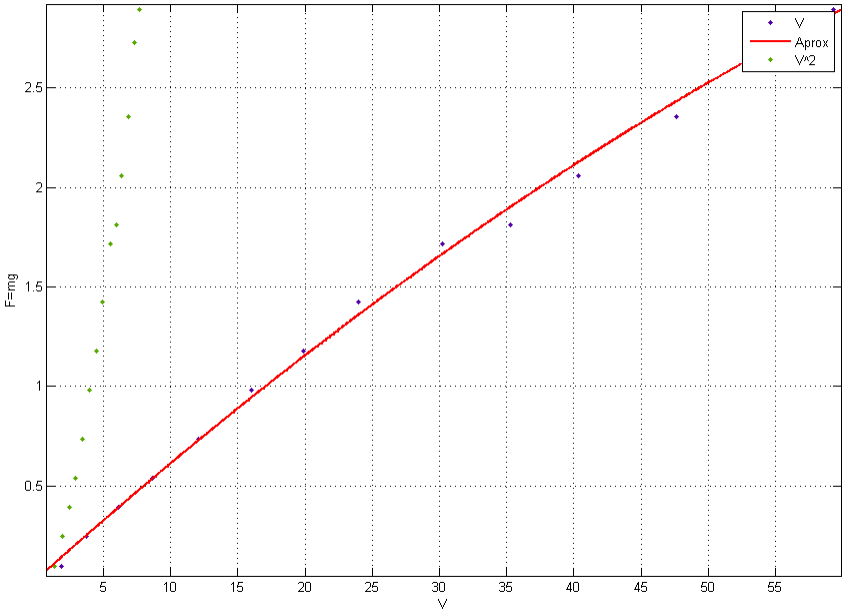

Fig. 5.18. Aproximación lineal de F vs V.

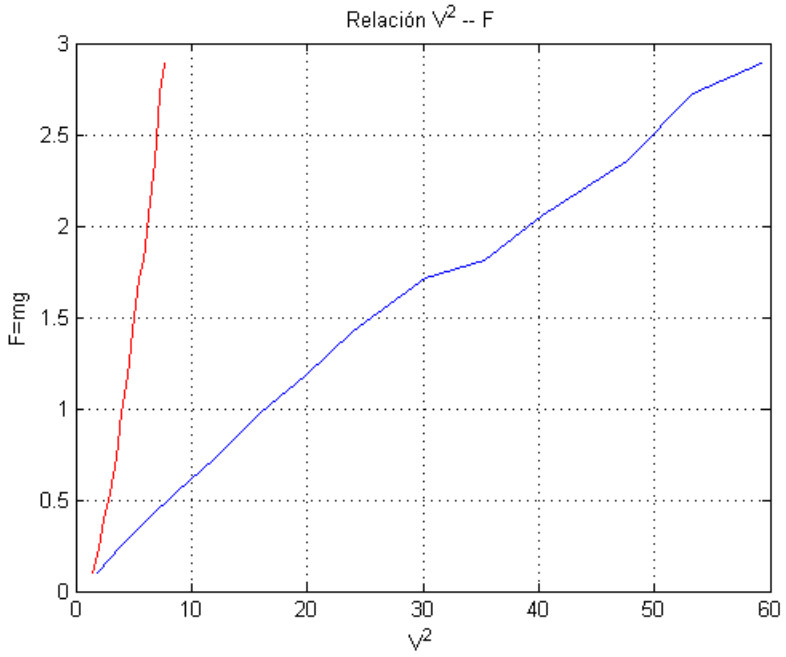

Fig. 5.19. Las relaciones F vs V^2 y F vs V en un solo gráfico.

medición, se hace un promedio de las cuatro constantes, esto es:

$$k_1 = \frac{k_{m1} + k_{m2} + k_{m3} + k_{m4}}{4}$$

13. Regresar las aspas a su posición original y montar todo nuevamente.

5.5.2. Medición de constante k_2

Para estimar el valor de la constante k_2 se realizó el siguiente experimento (Ver Figura 5.20).

1. De la estructura del helicóptero se desmonta el motor con el rotor en conjunto con el poste que lo sostiene.

2. Medir los siguientes parámetros: la distancia desde un extremo del poste hasta el motor y la masa del motor junto con sus rotores

3. En el borde de una superficie, colocar un transporatdor que servirá de guía en la medición angular.

4. Del poste que sostiene al motor se le anexa un alambre de cobre que coadyuvará en la medición del desplazamiento angular.

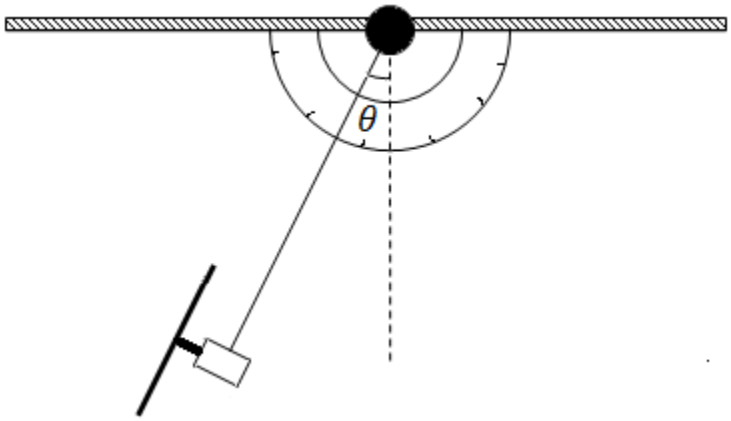

Fig. 5.20. Configuración experimental para k_2.

Tabla 5.3. Valor de constantes k_1 y k_2

k_1	k_2
0.0481	0.0371

5. Una vez colocado el arreglo como se recrea en la Figura R, se alinea el motor en una posición que se tomará como referencia.

6. Posteriormente el motor se alimenta con el voltaje mínimo que gradualmente se incrementa hasta que haya un desplazamiento angular visible.

7. Se realiza el experimento con diferentes niveles de voltaje y se anota con la relación angular.

8. Mediante la relación angular y la distancia del poste al rotor, se determina la fuerza producida por el motor y las aspas.

9. Se grafica la relación Fuerza contra Voltaje y Fuerza contra Voltaje cuadrado (Figura 5.21)

10. Mediante una aproximación polinomial se encuentra el valor de la constante k_2.

11. El experimento se vuelve a repetir con cada uno de los cuatro motores y se realiza un promedio con las constantes encontradas.

El valor encontrado de las constantes se muestra en la Tabla 5.3. Las constantes son adimensionales.

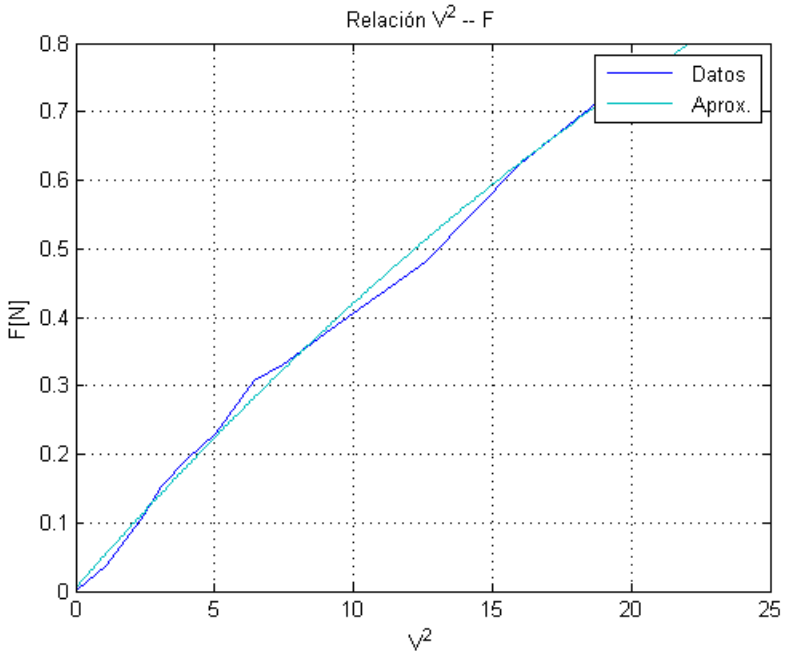

Fig. 5.21. Relación F vs V^2 y su aproximación polinomial (línea color claro).

Capítulo 6

Resultados Experimentales y Discusión

En esta sección se muestran los resultados experimentales obtenidos mediante la aplicación del control PD mostrado en el capítulo 4. El controlador propuesto en (4.1)-(4.7) está compuesto por seis controladores PD, tres pertenecen al control de los ángulos de orientación que son los que se encargan de la estabilización de la aeronave, otro controlador se encarga de regular la posición en z y los otros dos controladores restantes modifican la posición de $x - y$, éstos están relacionados directamente con el control de orientación.

Una vez descrito el hardware y software del sistema se procederá a describir la forma en la que se llevaron los experimentos de control de estabilización de la aeronave.

Como se vio en el Capítulo 3, la descripción matemática del helicóptero es compleja por ser un sistema de seis grados de libertad, subactuado y no lineal, por esta razón se pensó que en vez de programar los seis controladores en una sola prueba sería una buena idea dividir la experimentación en cuatro partes. En la primera se prueba sólo el controlador Yaw que es el encargado de proporcionar el giro de la aeronave. En la segunda se verifican los contoladores de Roll y Pitch en conjunto con el de Yaw para lograr la estabilización del helicóptero. En la tercera parte, se programa el controlador de altura en Z en conjunto con los tres controladores anteriormente mencionados. En la cuarta y última parte, los controladores de desplazamiento en el eje $x-y$ son probados. En los párrafos siguientes se describirá el procedimiento seguido para llevar a cabo la experimentación.

6.1. Prueba del controlador Yaw

Para esta prueba, primero se montó todo el prototipo experimental colgado desde una estructura mediante un hilo grueso de forma que estuviera separado a 30 centímetros de distancia del suelo para que en caso de que el helicóptero se llegase a desbalancear por la acción de los motores, las hélices no impacten contra el suelo o contra la persona que realiza

el experimento. Posteriormente, se colocaron un par baterías cuidando de que el prototipo quedase balanceado, esto es, que la estructura quede paralela a la superficie. Después, se posicionó el sensor de orientación de tal forma que la lectura del ángulo Yaw marcara los cero radianes. Finalmente se enciende el sistema de control.

Como se observa en la Figura 6.1, se introduce una señal de referencia tipo escalón, el punto de referencia (setpoint) permanece en un ángulo de 0 radianes durante 15 segundos para posteriormente desplazarse 0.5 radianes. En la figura se observa que la señal de control alcanza a la señal de referencia en tres segundos, pero cuando ocurre el cambio en la señal de referencia, al controlador de Yaw le toma más tiempo en seguir a dicha señal, pues se perciben oscilaciones en la señal de control que se minimizan conforme se llega al estado estacionario. Inicialmente se pensó que las oscilaciones eran producto de una mala sintonización por lo que se probaron diferentes combinaciones de ganancias para la sintonización de los controladores con el objetivo de minimizar dichas oscilaciones, no obstante, este comportamiento prevalecía. Otra hipótesis planteada, es que a partir de la observación de la Figura 6.1, fue la de introducir un filtro pasa bajas encargado minimizar el impacto del ruido de alta frecuencia, esto es debido a que como se está usando un control con parte derivativa es sabido que el ruido inherente en las señales de control puede ser amplificado por el mismo controlador. A partir de esta información se construyó un filtro RC (Resistivo-Capacitivo) de primer orden pasivo a una frecuencia de corte de 22 Hz -como se describión en el capítulo 5-, esto trajo como consecuencia que las señales de los sensores suprimieran el ruido de alta frecuencia expandiendo el rango seleccionable de ganancias del controlador, el resultado de la introducción del filtro se muestra en la Figura 6.2. En la figura mencionada aún se observan las oscilaciones en estado estacionario, esto llevo a reformular la hipótesis de que los controladores se podrían probar por separado, ya que como se pudo comprobar en este experimento las perturbaciones al sistema -por ejemplo, las vibraciones por el funcionamiento de los motores- afectaban no sólo al ángulo yaw, sino a también a los ángulos roll y pitch, por tanto, se pensó en introducir los tres controladores de orientación para verificar que el sistema se estabilice.

Parar realizar esta prueba se debe tener en consideración que no hay interacción de los sensores de posición (x, y, z). De la relación de pares mostrada en (4.11), se requiere el conocimiento tanto de los torques aplicados a los ejes como también la fuerza total aplicada por el cuatrirotor u. Los torques aplicados en los ejes se encuentran mediante las lectura del sensor IMU, haciendo $(\tau_{xB} = 0, \tau_{yB} = 0)$ y τ_{zB} dependiente del cálculo del controlador. La fuerza $u = 1$ se especifica un valor debido a que no se introduce el sensor de medición de altura.

Las ganancias usadas para el controlador de Yaw se muestran en la Tabla 6.1:

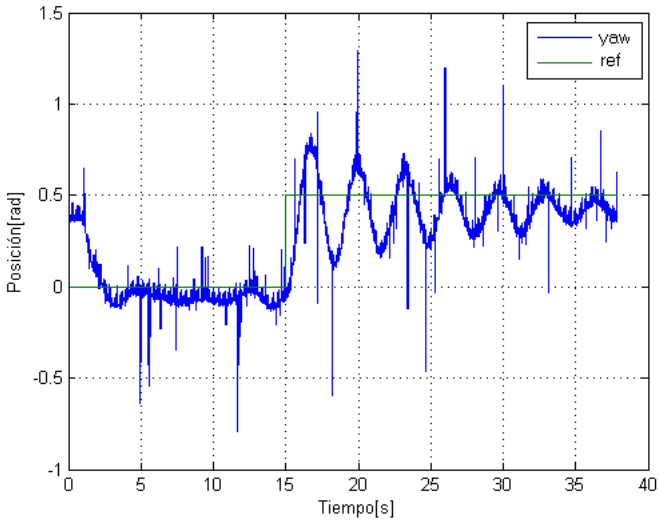

Fig. 6.1. Prueba de controlador Yaw sin filtro.

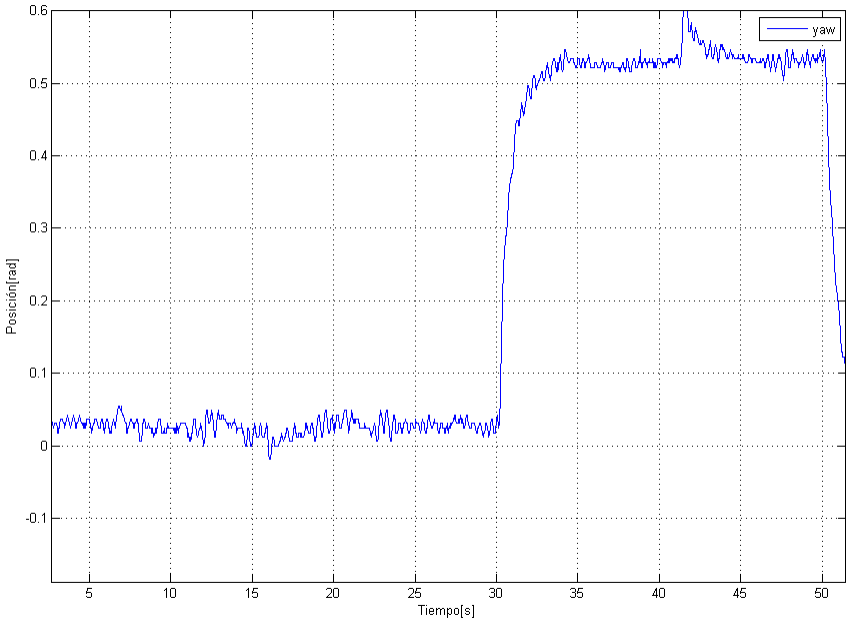

Fig. 6.2. Prueba de controlador Yaw con etapa de filtrado.

Tabla 6.1. Ganancias para el controlador Yaw

Ganancia	Símbolo	Valor
Proporcional	K_{pyaw}	0.7
Derivativa	K_{dyaw}	0.5

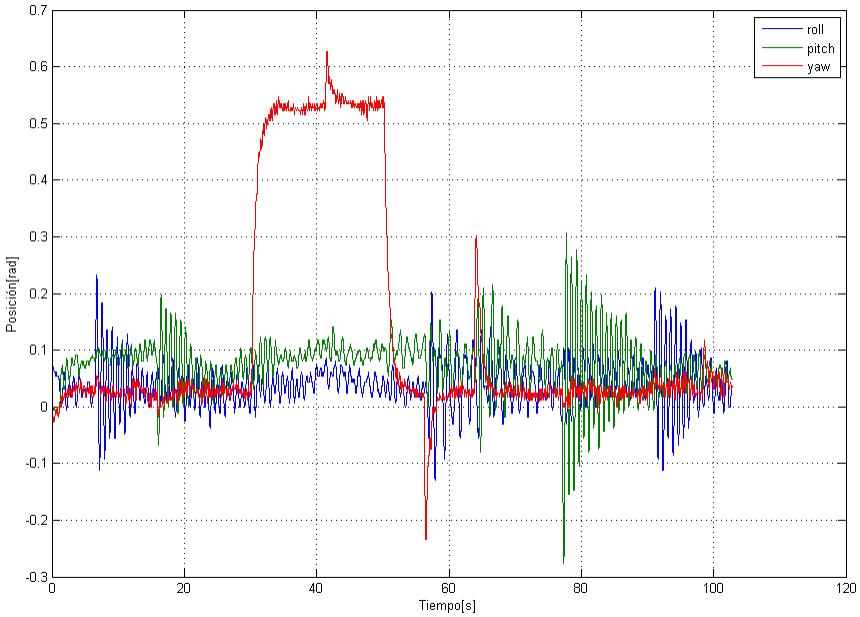

Fig. 6.3. Prueba del control de orientación, sometido a perturbaciones.

6.2. Prueba del controlador de orientación

Siguiendo la metodología anterior, en esta etapa se programaron los tres controladores de orientación mostrados en la serie de ecuaciones (4.1)-(4.7). Primero, se posicionó el sensor IMU con la medición del ángulo yaw en cero radianes. Después, se posicionó al helicóptero de forma paralela al suelo y balanceado, esto trajo como consecuencia que los ángulos de pitch y roll medidos por el sensor estuviesen cercanos a los cero radianes. Nuevamente, se introdujo una referencia del tipo escalón al ángulo yaw que durante los primeros 30 segundos permanecería en cero radianes, luego la referencia cambiará a 0.5 radianes donde se mantendría por otros 20 segundos más, por último, la referencia volvería a su origen de cero radianes hasta el término del experimento.

Como se puede ver en la Figura 6.3. En el experimento, los ángulos de roll y pitch permanececieron en cero radianes. En esta prueba se introdujeron perturbaciones en cada uno de los ángulos de orientación, dichas perturbaciones consistieron en empujar los motores o la misma estructura del helicóptero. Dentro de los primeros 30 segundos se perturba toda la estructura, posteriormente se manipula uno de los motores relacionado con el ángulo Pitch. La siguiente perturbación se realizó después de que el ángulo yaw regresara a su referencia cero. Cuando la aeronave se estabilizó,entonces, se procedió a empujar con la mano el motor relacionado con el movimiento del ángulo roll.

De la Figura 6.3 podemos observar que se mejoró notablemente la respuesta al controla-

Tabla 6.2. Ganancias para el controlador de orientación

Ganancia	Símbolo	Valor
Proporcional Yaw	K_{pyaw}	0.7
Derivativa Yaw	K_{dyaw}	0.5
Proporcional Roll	K_{pr}	0.1
Derivativa Roll	K_{dr}	0.07
Proporcional Pitch	K_{pp}	0.1
Derivativa Pitch	K_{dp}	0.07

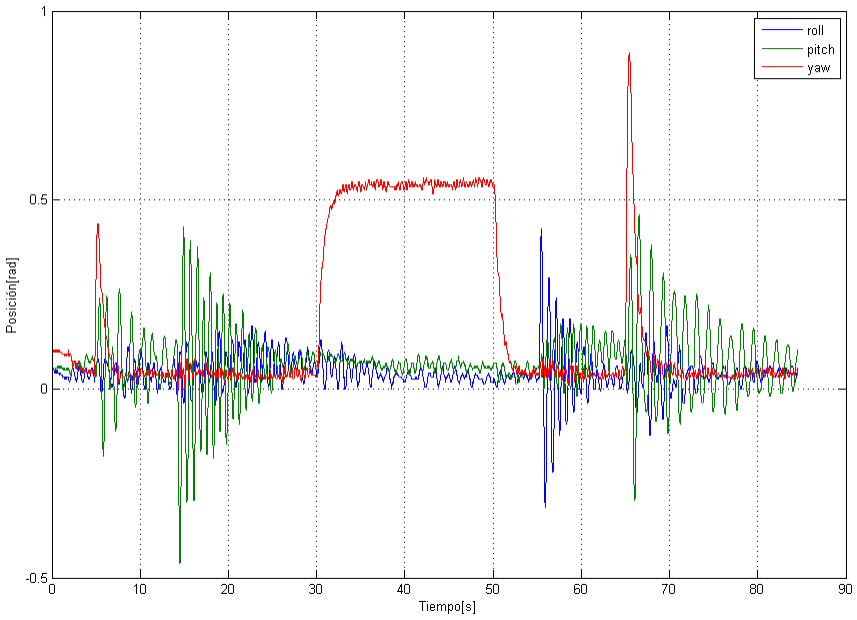

Fig. 6.4. Prueba del control de orientación, sometido a perturbaciones (segunda prueba).

dor yaw, pues no se visualizaron las oscilaciones constantes como se observó en la Figura 6.1 del apartado anterior, esto indica que la introducción de los tres controladores en conjunto es necesaria para poder estabilizar la aeronave. Las oscilaciones en los ángulos roll y pitch duraron por un tiempo más largo en comparación con las de yaw, esto posiblemente se debería a que en la descripción del controlador realizado para el modelo se consideraban perturbaciones de pequeña magnitud, por lo cual se supone que la variación en los ángulos debería de ser minúscula. Pese a esta condición el controlador tiende a minimizar las oscilaciones conforme transcurre el tiempo.

Las ganancias encontradas para este controlador se muestran en la Tabla 6.2.

En las figuras 6.4 y 6.5 se muestra el mismo experimento pero introduciendo las perturbaciones en diferentes partes de la prueba, todo esto con el fin de mostrar repetibilidad.

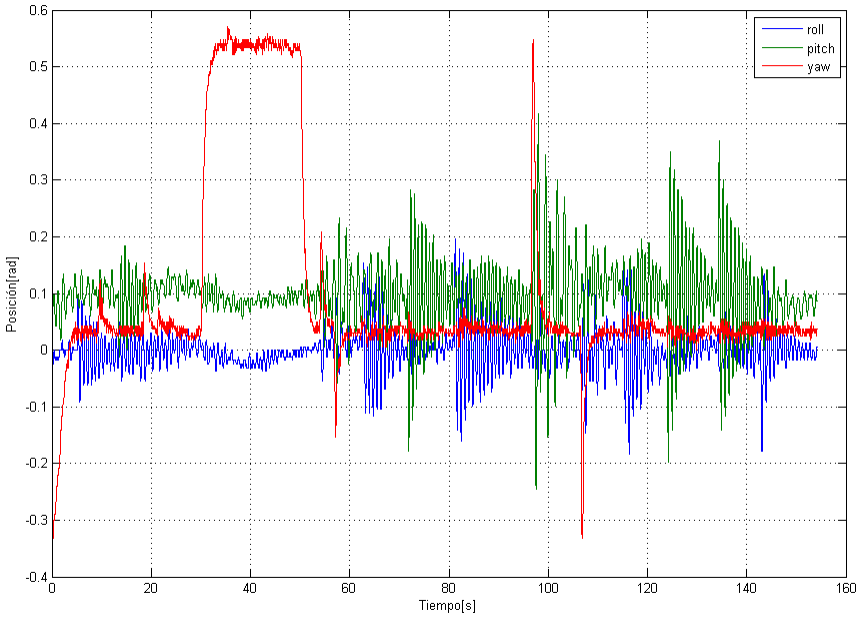

Fig. 6.5. Prueba del control de orientación, sometido a perturbaciones (tercer prueba).

6.3. Prueba del controlador de altura

Una de las pruebas adicionales que se realizaron fue la introducción del controlador en Z. Este controlador regula la posición de la nave respecto del suelo y se relaciona con los ángulos de pitch y roll como se ve en la ecuación (4.1). Para esta prueba se sigue la metodología anterior, con un ligero cambio en el banco de pruebas, el hilo grueso que sostenía al prototipo fue colocado por dentro de un cilindro hueco que sirvió como guía con lo cual se utilizó como prevención en caso de que el controlador arrojase valores erróneos, el helicóptero no se desbalancearía, ni colapsaría contra alguna estructura cercana. Posteriormente, se programó el controlador de altura (eje Z) en conjunto con los tres controladores de orientación. La señal de referencia, en este caso, se refiere a la posición vertical de la aeroanve. La referencia que se introdujo fue una de tipo escalón, donde los primeros 10 segundos permaneció a 27 centímeros de distancia respecto al suelo, pasando estos segundos, la referencia se movió a 32 cm - recuérdese que el eje z disminuye si el helicóptero se aleja del suelo- durante otros 10 segundos y finalmente volvió a la posición inical por el resto del experimento. Los resultados se muestran en la Figura 6.6 y las ganancias que se usaron se muestran en la Tabla 6.3.

De la Figura se observa que la respuesta del controlador es oscilatoria pero que aún así, éste sigue a la señal de referencia con cierto error de seguimiento, una de las posibles causas por las cuales esto sucede, es debido al tipo de sensor utilizado para medir distancia, el sensor usado es del tipo infrarojo y que como se vio en el capítulo anterior su respuesta no es lineal.

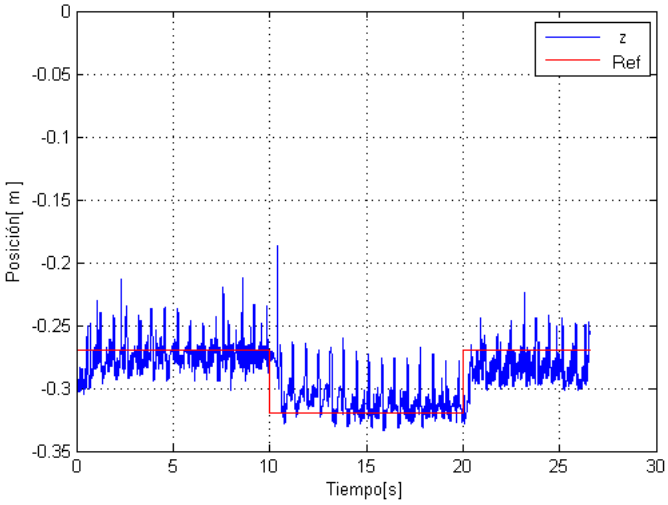

Fig. 6.6. Prueba del control de orientación junto con el controlador en z.

Tabla 6.3. Ganancias para el controlador en z

Ganancia	Símbolo	Valor
Proporcional	K_{pz}	2.0
Derivativa	K_{dz}	0.1

Aunque se realizó una aproximación mediante una función exponencial, esta estimación no es perfecta y como se observó produjo variaciones en la lectura debido a la resolución del dispositivo. Se probaron diferentes combinaciones en las ganancias de controlador y el efecto oscilatorio seguía presente.

Capítulo 7

Conclusiones

En esta tesis se propuso un controlador PD con compensación de gravedad para la estabilización de un helicóptero de cuatro rotores. El controlador propuesto está diseñado para estabilizar el prototipo aún en presencia de perturbaciones, la eficiencia del controlador fue verificado tanto con simulación como con pruebas experimentales. En cuanto a la verifiación por medio de simulación, el controlador propuesto mostró tener un desempeño eficiente para el control de los ángulos de orientación. Las simulación mostrada sirve como referencia del comportamiento del prototipo con el control y las ganancias encontradas en simulación no deberán tomarse como las óptimas, pues se debe mencionar que la simulación trabaja con base en un modelo matemático propuesto y que en dichos modelos -generalmente- hay efectos físicos que por simplificar el aspecto matemático no se consideran o se ignoran.

Por otra parte, la eficiencia del controlador fue mostrada al probarse en un prototipo experimental. En experimentación se observó que cuando se prueban los tres controles estabilizadores el sistema sigue a la referencia deseada con un mínimo error de regulación y que además es robusto ante perturbaciones, pues como se describió en la sección anterior, la estructura del helicóptero fue manipulada mediante ligeros golpes para simular perturbaciones y observar su comportamiento ante éstas.

Otro punto importante a resaltar es respecto al diseño de hardware, la electrónica del helicóptero de cuatro rotores fue completamente diseñada por el autor, la cual brinda una ventaja sobre los prototipos comerciales que en general tienen un diseño cerrado que no permiten la modificación de las distintas etapas del hardware de acuerdo a las necesidades del usuario. Por ello, con la fabricación de un diseño propio se propicia un mejoramiento o adaptación de nuevos módulos o dispositivos electrónicos según la necesidad del controlador propuesto o las funciones a futuro del helicóptero. Así, por ejemplo, pueden usarse otras plataformas diferentes al DSC, como un microcontrolador o bien un DSP (procesador digital de señales, del inglés Digital Signal Processing) que pueden simplificar el sistema, potenciar

el cálculo del controlador o reducir el tiempo de muestreo del controlador. Otra de las ventajas que tiene el diseño de un sistema propio es que permite describir a modo de tutorial; la construcción del sistema, los cálculos correspondientes en las diferentes etapas del diseño electrónico, describir los problemas que se presentaron, las soluciones que se le dieron y que cualquier lector afin a la electrónica -sin importar su nivel- podrá fácilmente reproducir el experimento.

7.1. Trabajo a futuro

En el presente trabajo se mostró la aplicación del control de estabilización de los ángulos de Euler y del controlador de altura en Z. Como trabajo futuro, resta la prueba de los controladores en $x - y$, así como también la prueba de seguimiento de trayectorias del controlador propuesto.

Apéndice A

Programa C para DSC embebido

```
#include "main.h"
#include <stdlib.h>
#include <math.h>

#pin_select OC1=PIN_B15
#pin_select OC2=PIN_B13
#pin_select OC3=PIN_B12
#pin_select OC4=PIN_B14

#DEFINE Ts        0.005
#DEFINE iTs       200.0
#DEFINE maxCADC   4096.0
#DEFINE maxang    6.2832
#DEFINE vo        10.0
#DEFINE offsetdig 2048.0
#DEFINE Kpyaw     0.7
#DEFINE Kdyaw     0.5
#DEFINE Kpr       0.0
#DEFINE Kdr       0.07
#DEFINE Kpp       0.0
#DEFINE Kdp       0.07
#DEFINE Kpx       1.0
#DEFINE Kdx       1.0
#DEFINE Kpy       1.0
#DEFINE Kdy       1.0
#DEFINE Kpz       1.0
```

```
#DEFINE Kdz       1.0
//------------Declaracion de variables------------//
int16 roll,pitch,yaw,x,y,z,inter;
int8 ban,ban3,rollH,rollL,pitchH,pitchL,yawH,yawL,xH,xL,yH,yL,zH,zL
   ;
unsigned int16 m1,m2,m3,m4,i=0;
float rollf,pitchf,yawf,xf,yf,zf,t=0,esc=maxang/maxCADC,escp=3.0/
   maxCADC,escs=8415/vo;
float yawf_1,pitchf_1,rollf_1,xf_1,yf_1,zf_1,yawfp,rollfp,pitchfp,
   xfp,yfp,zfp,taux,tauy,tauz,ux,uy,r1;
float yawde=0.0,rollde=0.0,pitchde=0.0,xde=0.0,yde=0.0,zde=0.0;
float u=1;
float w1_2,w2_2,w3_2,w4_2,w1,w2,w3,w4;


// Interrupción al Timer. Tiempo de muestreo
#INT_TIMER1
void TIMER1_isr(void)
{
    ban3=0;
    output_toggle(pin_b8);
    set_timer1(59285); //Tiempo de muestreo de 5ms TMR=(Ts)*(fxtal)
       /(2*Tdir)
}

//------------Programa principal------------//
void main(void)
{
   setup_adc_ports(sAN0 | sAN1 | sAN2 | sAN3 | sAN4 | sAN5, VSS_VDD
      );
   setup_adc(ADC_CLOCK_DIV_4 | ADC_TAD_MUL_4);

   setup_compare(1,COMPARE_PWM | COMPARE_TIMER2 );            //
      INICIALIZACION DEL PWM1
   setup_compare(2,COMPARE_PWM | COMPARE_TIMER2 );            //
      INICIALIZACION DEL PWM2
   setup_compare(3,COMPARE_PWM | COMPARE_TIMER2 );            //
      INICIALIZACION DEL PWM3
```

```
57      setup_compare(4,COMPARE_PWM | COMPARE_TIMER2 );             //
           INICIALIZACION DEL PWM4
58      setup_timer2(TMR_INTERNAL | TMR_DIV_BY_1,8415);      //
           CONFIGURACION DEL TIMER 2 CON 2047 CICLOS
                                                                    //ES
           DECIR 11 BITS DA 19.500Hz
59
60      setup_timer1(TMR_INTERNAL | TMR_DIV_BY_8);           //PWM con
           rango de 0 a 8332, falta escalamiento
61      set_timer1(59285);
62
63      setup_wdt(WDT_OFF);
64      enable_interrupts(INT_TIMER1);
65      enable_interrupts(GLOBAL);
66
67      while (1)
68       {
69         yawf_1   = yawf;
70         pitchf_1 = pitchf;
71         rollf_1  = rollf;
72         xf_1 = xf;
73         yf_1 = yf;
74         zf_1 = zf;
75
76         set_adc_channel(0);  //seleccionando canal 0
77         delay_us(10);        //retardo para estabilizar entrada
78         yaw=read_adc();       //leyendo yaw
79         inter=(yaw)&(0xFF00);
80         yawH=inter>>8;
81         yawL=(yaw)&(0x00FF);
82
83         set_adc_channel(1);  //seleccionando canal 1
84         delay_us(10);        //retardo para estabilizar entrada
85         roll=read_adc();       //leyendo pitch
86         inter=(roll)&(0xFF00);
87         rollH=inter>>8;
88         rollL=(roll)&(0x00FF);
89
90         set_adc_channel(2);  //seleccionando canal 2
```

```
91          delay_us(10);         //retardo para estabilizar entrada
92          pitch=read_adc();       //leyendo roll
93          inter=(pitch)&(0xFF00);
94          pitchH=inter>>8;
95          pitchL=(pitch)&(0x00FF);
96
97          set_adc_channel(3);  //seleccionando canal 3
98          delay_us(10);         //retardo para estabilizar entrada
99          x=read_adc();       //leyendo x
100         inter=(x)&(0xFF00);
101         xH=inter>>8;
102         xL=(x)&(0x00FF);
103
104         set_adc_channel(4);  //seleccionando canal 4
105         delay_us(10);         //retardo para estabilizar entrada
106         y=read_adc();       //leyendo y
107         inter=(y)&(0xFF00);
108         yH=inter>>8;
109         yL=(y)&(0x00FF);
110
111         set_adc_channel(5);  //seleccionando canal 5
112         delay_us(10);         //retardo para estabilizar entrada
113         z=read_adc();       //leyendo z
114         inter=(z)&(0xFF00);
115         zH=inter>>8;
116         zL=(z)&(0x00FF);
117
118         putc(0xAA);       //mandando dato de reconocimieno al puerto
               serial
119         putc(rollH);       //mandando roll al puerto serial
120         putc(rollL);
121         putc(pitchH);     //mandando pitch al puerto serial
122         putc(pitchL);
123         putc(yawH);       //mandando yaw al puerto serial
124         putc(yawL);
125         putc(xH);         //mandando x al puerto serial
126         putc(xL);
127         putc(yH);         //mandando y al puerto serial
128         putc(yL);
```

```
129         putc(zH);          //mandando z al puerto serial
130         putc(zL);
131
132         yaw= yaw - offsetdig;
133         yawf= (signed int16)yaw;
134         yawf   =  esc*yawf;
135
136         roll= roll - offsetdig;
137         rollf= (signed int16)roll;
138         rollf   =  esc*rollf;
139
140         pitch= pitch - offsetdig;
141         pitchf= (signed int16)pitch;
142         pitchf   =  -esc*pitchf;
143
144           xf=escp*x;
145           xf=1.7115*exp(-0.896*xf);
146
147           yf=escp*y;
148           yf=1.7115*exp(-0.896*yf);
149
150           zf=escp*z;
151           zf=-1.7115*exp(-0.896*zf);
152
153           yawfp  = (yawf  - yawf_1)*iTs;
154           pitchfp= (pitchf-pitchf_1)*iTs;
155           rollfp = (rollf - rollf_1)*iTs;
156           xfp = (xf - xf_1)*iTs;
157           yfp = (yf - yf_1)*iTs;
158           zfp = (zf - zf_1)*iTs;
159
160           //inicia controlador
161
162           taux = Kpr*(rollde - rollf) + Kdr*(-rollfp);
163           tauy = Kpp*(pitchde - pitchf) + Kdp*(-pitchfp);
164           tauz = Kpyaw*(yawde - yawf) + Kdyaw*(-yawfp);
165
166           if(taux>30)
167                   taux=30;
```

```
168         if(taux<-30)
169                 taux=-30;
170
171         if(tauy>30)
172                 tauy=30;
173         if(tauy<-30)
174                 tauy=-30;
175
176         if(tauz>30)
177                 tauz=30;
178         if(tauz<-30)
179                 tauz=-30;
180
181         w1_2= 5.1975*u - 47.25*taux + 80.6452*tauz;
182         w2_2= 5.1975*u + 47.25*tauy - 80.6452*tauz;
183         w3_2= 5.1975*u + 47.25*taux + 80.6452*tauz;
184         w4_2= 5.1975*u - 47.25*tauy - 80.6452*tauz;
185
186         if(w1_2<0)
187                 w1_2=0;
188         if(w2_2<0)
189                 w2_2=0;
190         if(w3_2<0)
191                 w3_2=0;
192         if(w4_2<0)
193                 w4_2=0;
194
195
196         w1=sqrt(w1_2);
197         w2=sqrt(w2_2);
198         w3=sqrt(w3_2);
199         w4=sqrt(w4_2);
200
201        //Saturación en voltaje
202
203         if(w1>vo)
204                 w1=vo;
205         if(w1<0)
206                 w1=0;
```

```
207          if(w2>vo)
208                  w2=vo;
209          if(w2<0)
210                  w2=0;
211          if(w3>vo)
212                  w3=vo;
213          if(w3<0)
214                  w3=0;
215          if(w4>vo)
216                  w4=vo;
217          if(w4<0)
218                  w4=0;
219          //Conversión voltaje a cuentas
220          m1=escs*w1;
221          m2=escs*w2;
222          m3=escs*w3;
223          m4=escs*w4;
224          //Enviando valores a PWM
225
226         set_pwm_duty(1,m1);
227         set_pwm_duty(2,m2);
228         set_pwm_duty(3,m3);
229         set_pwm_duty(4,m4);
230         ban3=1;
231
232         //Setpoints de control
233          if(t<20.0){
234         set_pwm_duty(1,500);
235         set_pwm_duty(2,500);
236         set_pwm_duty(3,500);
237         set_pwm_duty(4,500);
238         }
239          if(t>=20.0){
240                  yawde=0.0;
241                  rollde=0.0;
242                  pitchde=0.0;}
243          if(t>=30.0){
244                  yawde=0.5;
245                  rollde=0.0;
```

```
246                     pitchde=0.0;}
247             if(t>=40.0){
248                     yawde=0.0;
249                     rollde=0.0;
250                     pitchde=0.0;}
251             while(ban3);
252             t=t+Ts;
253         }
254     }//cierre del main
```

Apéndice B

Cabecera Programa principal

```
#include "C:\Archivos de programa\PICC\Devices\33FJ64GP802.h"
#device ADC=12

#FUSES ICSP3
#FUSES PR              // Primary Oscillator with PLL
#FUSES HS             // Crystal osc <= 4mhz for PCM/PCH , 3mhz to
   10 mhz for PCD
#FUSES NOIESO           // Internal External Switch Over mode
   enabled
#FUSES CKSNOFSM       // Clock Switching is enabled, fail Safe
   clock monitor is disabled
#FUSES NOWDT          //  Watch Dog Timer
#FUSES NOCKSFSM         // Clock Switching is enabled, fail Safe
   clock monitor is enabled
#FUSES PUT2           // Power On Reset Timer value 2ms
#FUSES NOJTAG         // JTAG disabled
#FUSES NODEBUG        // No Debug mode for ICD
#FUSES NOPROTECT
#FUSES WINDIS

#use delay(clock=20M)

#pin_select U1TX=PIN_B6
#pin_select U1RX=PIN_B7
#use rs232(UART1, baud=57600, PARITY=N,BITS =8, STOP=1)
```

Apéndice C

Programa en Builder versión 6 para estación terrestre

```
//Programa para adquisición de datos
//
   ---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop
#include "Main.h"
#include <math.h>
#include <stdio.h>
#include <share.h>
#include <conio.h>
#include <dos.h>
#include <time.h>
#include <stdlib.h>
//
   ---------------------------------------------------------------------------

#pragma package(smart_init)
#pragma link "CSPIN"
#pragma resource "*.dfm"
//
   ---------------------------------------------------------------------------
```

```
18 #define Ts              0.025      //cada cuenta del TMR0 vale (4/
      FXtal)*256 seg
19 #define maxCADC         4096.0   //1023.0
20 #define maxang          6.2832
21 #define vo              10.0
22 //#define ret             80
23 FILE *ptrMonit;
24 TMainForm *MainForm;
25 unsigned char flagcom=0,flagfile=0,signo_sal,sel,m1,m2,m3,m4; //de
      8 bits
26 unsigned short int roll,pitch,yaw,x,y,z;    //de 16 bits
27 float rollf,pitchf,yawf,xf,yf,zf,t=0,esc=maxang/maxCADC,escp
      =3.3/1023.0;
28 float escs=255/vo,vp;
29 float yawf_1,pitchf_1,rollf_1,yawfp,rollfp,pitchfp,tauyb,tauxb,
      tauzb,taux,tauy,tauz;
30 float Kpr=0.0,Kdr=0.07;             //Kdr=0.07;  // Ganancias ROLL
31 float Kpp=0.0,Kdp=0.07;            //Kdp=0.07;  // Ganancias PITCH
32 float Kpy=0.7,Kdy=0.5;  // Ganancias YAW Kpy=0.7,Kdy=0.5;
33 float yawde=0.0,rollde=0.0,pitchde=0.0;
34 float u=1;
35 //double w1_2,w2_2,w3_2,w4_2;
36
37 float w1_2,w2_2,w3_2,w4_2;
38 float w1,w2,w3,w4;
39 float wa,wb,wc,wd;
40 //--------- Inicia adquisición de bytes
      ------------------------------//
41 void ProcessByte(BYTE byte)
42 {
43 if(flagcom!=0)
44         flagcom++;
45 if(((byte==0xAA))&&(flagcom==0))
46    {
47         roll=0;
48         pitch=0;
49         yaw=0;
50         x=0;
51         y=0;
```

```
52          z=0;
53          flagcom=1;
54      }
55  if(flagcom==2)
56      {
57          yaw=byte;
58          yaw=yaw<<8;
59      }
60  if(flagcom==3)
61      {
62          yaw=yaw+byte-2048; //512;
63      }
64  if(flagcom==4)
65      {
66          roll=byte;
67          roll=roll<<8;
68      }
69  if(flagcom==5)
70      {
71          roll=roll+byte-2048;
72      }
73  if(flagcom==6)
74      {
75          pitch=byte;
76          pitch=pitch<<8;
77      }
78  if(flagcom==7)
79      {
80          pitch=pitch+byte-2048;
81      }
82  if(flagcom==8)
83      {
84          x=byte;
85          x=x<<8;
86      }
87  if(flagcom==9)
88      {
89          x=x+byte;
90      }
```

```
91  if(flagcom==10)
92     {
93          y=byte;
94          y=y<<8;
95     }
96  if(flagcom==11)
97     {
98          y=y+byte;
99     }
100 if(flagcom==12)
101    {
102         z=byte;
103         z=z<<8;
104    }
105 if(flagcom==13)
106    {
107         z=z+byte;
108
109 //-------- Escalamiento para los sensores
       ------------------------------------//
110
111         yawf_1   = yawf;
112         pitchf_1 = pitchf;
113         rollf_1  = rollf;
114
115         yawf   = esc*(signed short int)yaw;
116         pitchf = -esc*(signed short int)pitch;
117         rollf  = esc*(signed short int)roll;
118
119         yawfp  = (yawf  - yawf_1)/Ts;
120         pitchfp= (pitchf-pitchf_1)/Ts;
121         rollfp = (rollf - rollf_1)/Ts;
122
123         //inicia controlador
124
125         taux = Kpr*(rollde - rollf) + Kdr*(-rollfp);
126         tauy = Kpp*(pitchde - pitchf) + Kdp*(-pitchfp);
127         tauz = Kpy*(yawde - yawf) + Kdy*(-yawfp);
128
```

```
129         if(taux>30)
130                 taux=30;
131         if(taux<-30)
132                 taux=-30;
133
134         if(tauy>30)
135                 tauy=30;
136         if(tauy<-30)
137                 tauy=-30;
138
139         if(tauz>30)
140                 tauz=30;
141         if(tauz<-30)
142                 tauz=-30;
143
144         w1_2= 5.1975*u - 47.25*taux + 80.6452*tauz;
145         w2_2= 5.1975*u + 47.25*tauy - 80.6452*tauz;
146         w3_2= 5.1975*u + 47.25*taux + 80.6452*tauz;
147         w4_2= 5.1975*u - 47.25*tauy - 80.6452*tauz;
148
149         if(w1_2<0)
150                 w1_2=0;
151         if(w2_2<0)
152                 w2_2=0;
153         if(w3_2<0)
154                 w3_2=0;
155         if(w4_2<0)
156                 w4_2=0;
157
158
159         w1=sqrt(w1_2);
160         w2=sqrt(w2_2);
161         w3=sqrt(w3_2);
162         w4=sqrt(w4_2);
163
164        //Saturación en voltaje
165
166         if(w1>vo)
167                 w1=vo;
```

```
168         if(w1<0)
169                 w1=0;
170         if(w2>vo)
171                 w2=vo;
172         if(w2<0)
173                 w2=0;
174         if(w3>vo)
175                 w3=vo;
176         if(w3<0)
177                 w3=0;
178         if(w4>vo)
179                 w4=vo;
180         if(w4<0)
181                 w4=0;
182         //Conversión voltaje a cuentas
183
184         m1=escs*w1;
185         m2=escs*w2;
186         m3=escs*w3;
187         m4=escs*w4;
188
189         if(t<30){
190                 yawde=0.0;
191                 rollde=0.0;
192                 pitchde=0.0;}
193         if(t>=30){
194                 yawde=0.5;
195                 rollde=0.0;
196                 pitchde=0.0;}
197         if(t>=50){
198                 yawde=0.0;
199                 rollde=0.0;
200                 pitchde=0.0;}
201
202         MainForm->Edit1->Text = FloatToStr (rollf);
203         MainForm->Edit2->Text = FloatToStr (pitchf);
204         MainForm->Edit3->Text = FloatToStr (yawf);
205         MainForm->Edit4->Text = FloatToStr (xf);
206         MainForm->Edit5->Text = FloatToStr (yf);
```

```
207          MainForm->Edit6->Text = FloatToStr (zf);
208          MainForm->Edit7->Text = FloatToStr (t);
209          MainForm->Edit8->Text = FloatToStr (w1);
210          MainForm->Edit9->Text = FloatToStr (w2);
211          MainForm->Edit10->Text = FloatToStr (w3);
212          MainForm->Edit11->Text = FloatToStr (w4);
213
214
215          //MainForm->Acknowledge();
216          ///MainForm->send_byte(0x0D);
217
218          //mandando primer byte
219          MainForm->Acknowledge();
220          MainForm->send_byte(m1);
221
222          //mandando segundo byte
223          MainForm->Acknowledge();
224          MainForm->send_byte(m4);
225
226          //mandando tercero byte
227          MainForm->Acknowledge();
228          MainForm->send_byte(m2);
229
230          //mandando cuarto byte
231          MainForm->Acknowledge();
232          MainForm->send_byte(m3);
233
234          /*abrir/crear un archivo*/
235          if(flagfile==0)
236             if((ptrMonit=fopen("MONIT.TXT", "w"))==NULL){}
237          flagfile=1;
238          /*escribir algunos datos en el archivo*/
239          fprintf(ptrMonit,"%3.3f\t%3.3f\t%3.3f\t%3.3f\t%3.3f\t%3.3f\
                t%3.3f\t%3.3f\t%3.3f\t%3.3f\t%3.3f\n",t,rollf,pitchf,
                yawf,xf,yf,zf,w1,w2,w3,w4);
240          flagcom=0;
241          t=t+Ts;
242      }
243  }
```

```
244 //
    ---------------------------------------------------------------------------

245 __fastcall TMainForm::TMainForm(TComponent* Owner)
246         : TForm(Owner), SerialPort(1, ProcessByte), fAcknowledge(
               true)
247 {
248         if (SerialPort.IsReady() != TRUE)
249                 MessageBox(NULL, "Problemas con el puerto", "Error"
                       , MB_OK);
250 }
251 //
    ---------------------------------------------------------------------------

252 void TMainForm::send_byte(unsigned char byte_sal)
253 {
254         if (fAcknowledge == false)
255                 return;
256         SerialPort.WriteByte(byte_sal);
257         fAcknowledge = false;
258 }
259 //
    ---------------------------------------------------------------------------

260 void TMainForm::Acknowledge()
261 {
262         fAcknowledge = true;
263 }
264 void __fastcall TMainForm::Button1Click(TObject *Sender)
265 {
266 /* close the file */
267 fclose(ptrMonit);
268 Close();
269 }
270 //
    ---------------------------------------------------------------------------
```

Apéndice D

Configuración del puerto RS232 para Builder versión 6

```
#include "SerialPort.h"
#include <strstream>
unsigned long __stdcall SerialPortThread(LPVOID pParam)
{
        TSerialPort *SerialPort = static_cast<TSerialPort *>(pParam
           );
        BYTE byte;

        while (1) {
                SerialPort->WaitCommData();
                if (SerialPort->ReadByte(byte))
                     SerialPort->ProcessByte(byte);
        }
        ExitThread(0);
        return 0;
}
TSerialPort::TSerialPort(UINT uPortNumber, void (*lpProcessByte)(
   BYTE byte))
        : m_bPortReady(FALSE), m_lpProcessByte(lpProcessByte)
{
        char DeviceName[128];
        std::ostrstream ostrDeviceName(DeviceName, 128);
        DCB dcb;
        ostrDeviceName << "Com" << uPortNumber << '\0';
```

```
23          m_hCom = CreateFile(DeviceName, GENERIC_READ |
               GENERIC_WRITE,
24                  FILE_SHARE_READ | FILE_SHARE_WRITE, NULL,
25                  OPEN_EXISTING, FILE_FLAG_OVERLAPPED, NULL);
26          m_bPortReady = SetupComm(m_hCom, 128, 128);
27          if (m_bPortReady == 0)
28                  return;
29          GetCommState(m_hCom, &dcb);
30          dcb.BaudRate = CBR_57600;
31          dcb.ByteSize = 8;
32          dcb.Parity = NOPARITY;
33          dcb.StopBits = ONESTOPBIT;
34          dcb.fAbortOnError = TRUE;
35          m_bPortReady = SetCommState(m_hCom, &dcb);
36          m_hThread = CreateThread(NULL, 0, SerialPortThread,
37                  static_cast<LPVOID>(this), 0, &m_dwThreadID);
38  }
39  TSerialPort::~TSerialPort()
40  {
41          TerminateThread(m_hThread, 0);
42          if (m_hCom != INVALID_HANDLE_VALUE)
43                  CloseHandle(m_hCom);
44  }
45  BOOL TSerialPort::IsReady() const
46  {
47          return m_bPortReady;
48  }
49  BOOL TSerialPort::WriteByte(BYTE byte)
50  {
51
52          OVERLAPPED osWrite = CreateOverlappedStructure();
53          DWORD dwBytesWritten;
54          BOOL fRes;
55
56          if (osWrite.hEvent == NULL)
57                  return FALSE;
58          if (!WriteFile(m_hCom, &byte, 1, &dwBytesWritten, &osWrite)
               ) {
59                  if (GetLastError() != ERROR_IO_PENDING) {
```

```
60                          fRes = FALSE;
61                  } else {
62                          if (!GetOverlappedResult(m_hCom, &osWrite,
                               &dwBytesWritten, TRUE))
63                                  fRes = FALSE;
64                          else
65                                  // Write operation completed
                                       successfully.
66                                  fRes = TRUE;
67                  }
68          } else
69                  fRes = TRUE;
70          CloseHandle(osWrite.hEvent);
71          return fRes;
72  }
73  BOOL TSerialPort::ReadByte(BYTE &byte)
74  {
75          OVERLAPPED osRead = CreateOverlappedStructure();
76          DWORD dwBytesRead;
77          BOOL fRes = FALSE;
78
79          if (osRead.hEvent == NULL)
80                  return FALSE;
81
82          if (!ReadFile(m_hCom, &byte, 1, &dwBytesRead, &osRead)) {
83                  if (GetLastError() != ERROR_IO_PENDING) {
84                          fRes = FALSE;
85                  } else {
86                          if (!GetOverlappedResult(m_hCom, &osRead, &
                               dwBytesRead, TRUE))
87                                  fRes = FALSE;
88                          else
89                                  // Read operation completed
                                       successfully.
90                                  fRes = TRUE;
91                  }
92          } else
93                  fRes = TRUE;
94          CloseHandle(osRead.hEvent);
```

```
95          return fRes;
96  }
97  BOOL TSerialPort::WaitCommData()
98  {
99          OVERLAPPED osWait = CreateOverlappedStructure();
100         DWORD dwEvtMask = EV_RXCHAR;
101         DWORD dwBytes;
102         BOOL fRes;
103
104         if (osWait.hEvent == NULL)
105                 return FALSE;
106         if (!WaitCommEvent(m_hCom, &dwEvtMask, &osWait)) {
107                 if (GetLastError() != ERROR_IO_PENDING) {
108                         fRes = FALSE;
109                 } else {
110                         if (!GetOverlappedResult(m_hCom, &osWait, &
                            dwBytes, TRUE))
111                                 fRes = FALSE;
112                         else
113                                 // Wait operation completed
                                    successfully.
114                                 fRes = TRUE;
115                 }
116         } else
117                 fRes = TRUE;
118         CloseHandle(osWait.hEvent);
119         return fRes;
120 }
121 OVERLAPPED TSerialPort::CreateOverlappedStructure()
122 {
123         OVERLAPPED o;
124
125         o.hEvent = CreateEvent(NULL, TRUE, FALSE, NULL);
126         o.Internal = 0;
127         o.InternalHigh = 0;
128         o.Offset = 0;
129         o.OffsetHigh = 0;
130         return o;
131 }
```

Apéndice E

Circuito electrónico del cuatrirotor

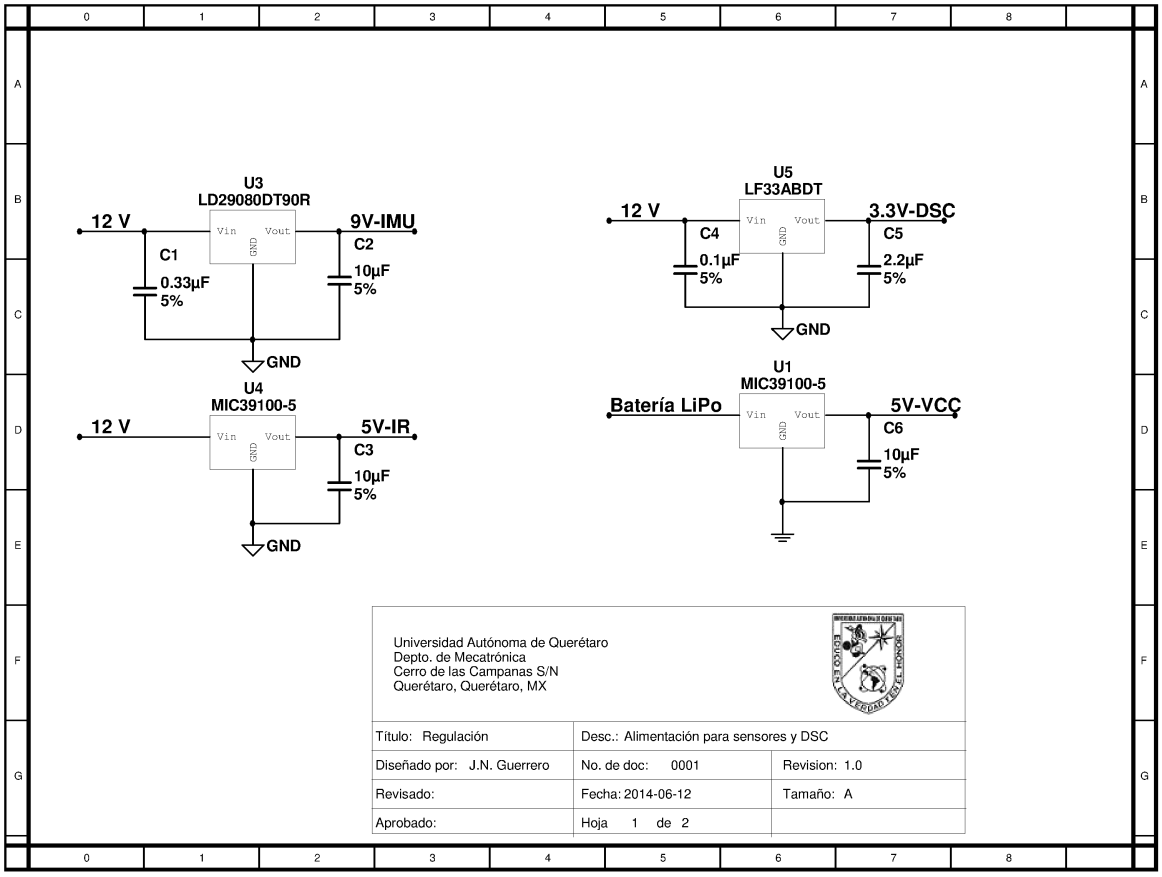

Fig. E.1. Circuito de Regulación de voltaje.

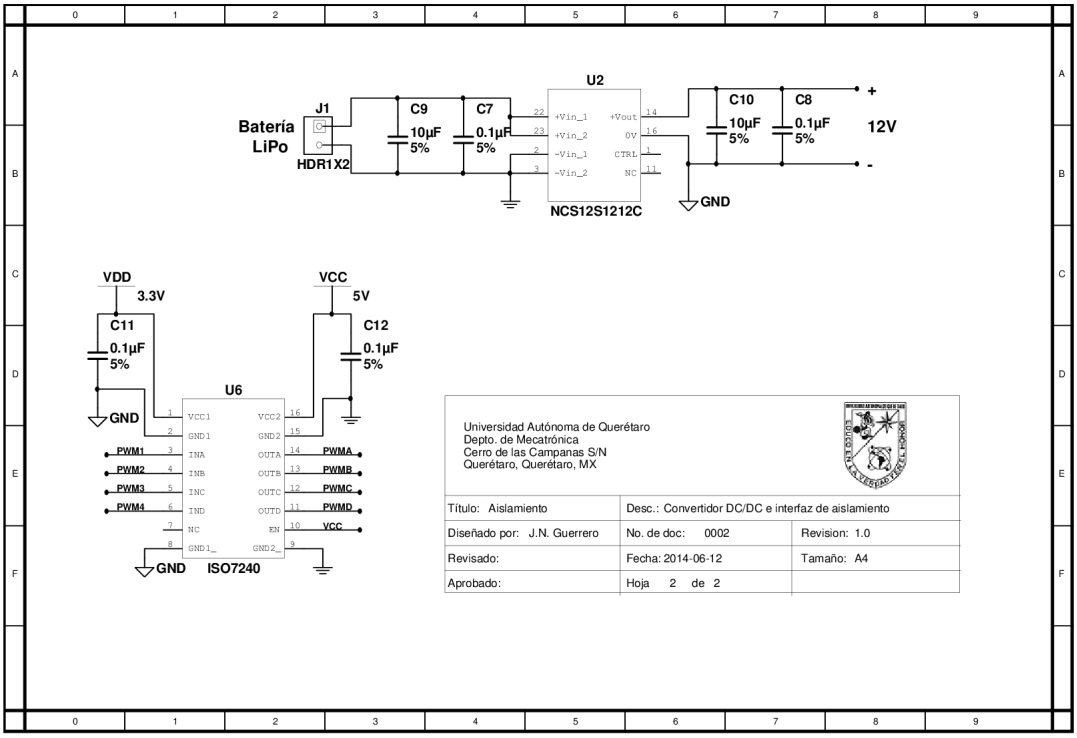

Fig. E.2. Circuito de aislamiento entrada/salida.

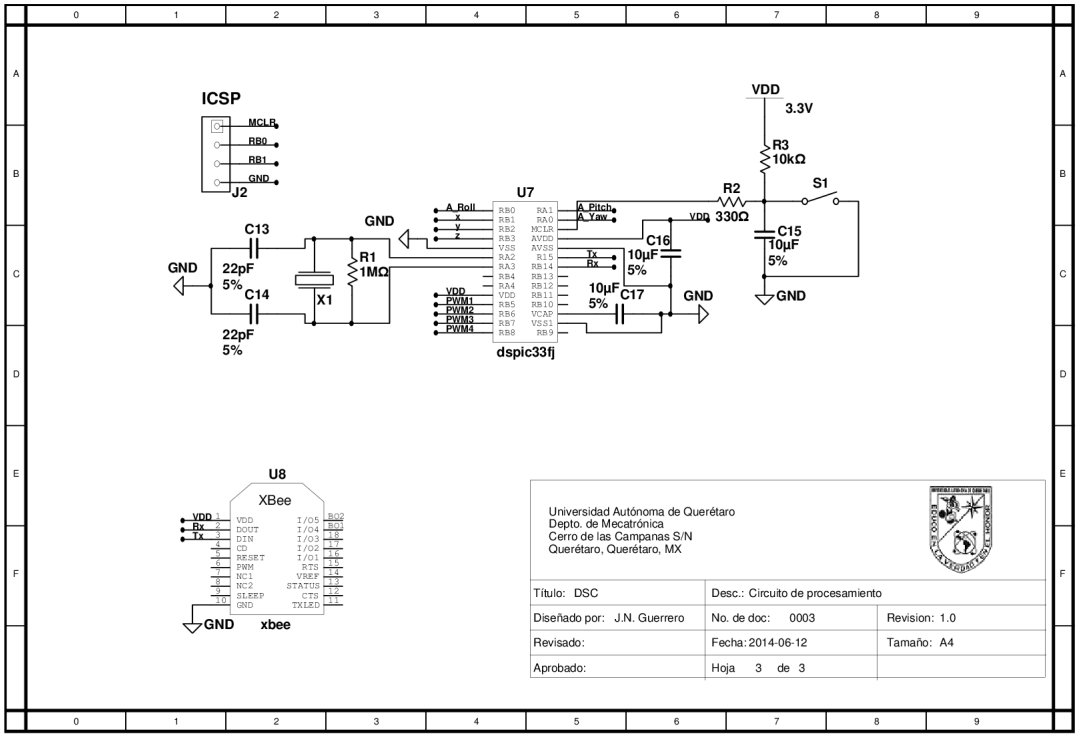

Fig. E.3. Intrumentación del DSC con XBee.

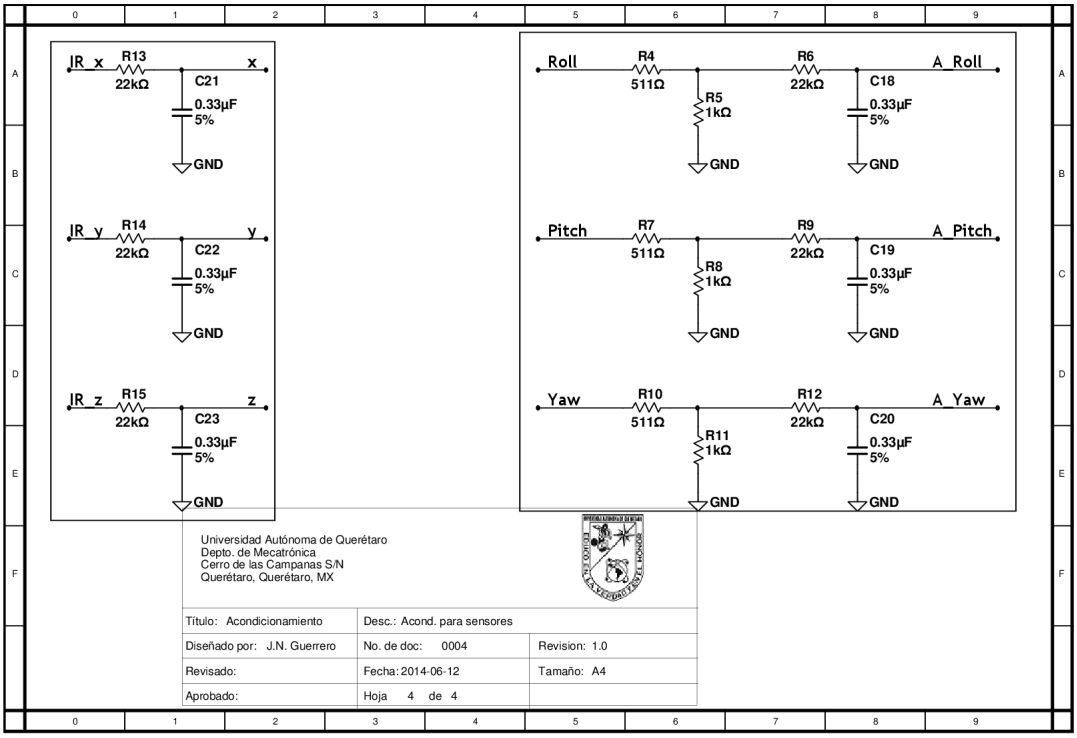

Fig. E.4. Circuito de acondicionamiento de señal.

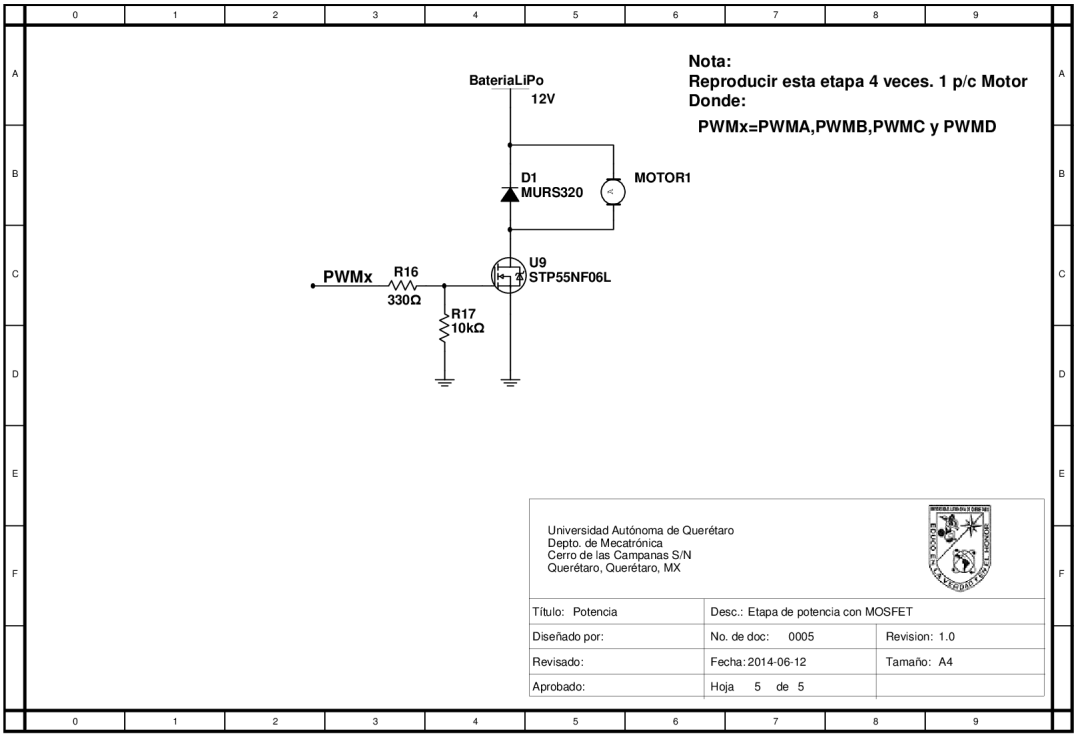

Fig. E.5. Etapa de potencia con MOSFET.

Bibliografía

Al-Hiddabi, S. a. (2009). Quadrotor control using feedback linearization with dynamic extension. *2009 6th International Symposium on Mechatronics and its Applications*, (1), 1–3.
URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5164788

Belkheiri, M., Rabhi, A., Hajjaji, A. E. L., y Egard, C. P. (2012). Different Linearization Control Techniques for a Quadrotor System. *2nd International Conference on Communications Computing and Control Applications, CCCA 2012*, (pp. 0–5).

Coza, C., y Macnab, C. J. B. (2006). A new robust adaptive-fuzzy control method applied to quadrotor helicopter stabilization. In *Annual Conference of the North American Fuzzy Information Processing Society - NAFIPS*, (pp. 475–479).

Erginer, B., y Altug, E. (2007). Modeling and PD Control of a Quadrotor VTOL Vehicle. *2007 IEEE Intelligent Vehicles Symposium*, (pp. 894–899).
URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4290230

Fang, S., Xu, Y., Jiang, J., Hu, B., y Que, X. (2011). The Analysis on Posture Control of Micro Quadrotor Based on PID. *2011 Fourth International Symposium on Computational Intelligence and Design*, (1), 283–286.
URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6079793

Fang, Z., Zhi, Z., Jun, L., y Jian, W. (2008). Feedback Linearization and Continuous Sliding Mode Control for a Quadrotor UAV. In *Proceedings of the 27th Chinese Control Conference*, (pp. 349–353).

Inversion, D. M., Control, A., y Network, O.-l. N. (2013). Analysis of Adaptive Control Using On-line Neural Networks for a Quadrotor UA V. Iccas, (pp. 1840–1844).

Khelfi, M. F., y Kacimi, a. (2012). Robust control with sliding mode for a quadrotor unmanned aerial vehicle. *Industrial Electronics (ISIE), 2012 IEEE International Symposium on*, (pp. 886–892).

Li, J., y Li, Y. (2011). Dynamic analysis and PID control for a quadrotor. *2011 IEEE International Conference on Mechatronics and Automation*, (pp. 573–578).
URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5985724

Madani, T., y Benallegue, A. (2006a). Backstepping Control for a Quadrotor Helicopter. (pp. 3255–3260).

Madani, T., y Benallegue, A. (2006b). Control of a Quadrotor Mini-Helicopter via Full State Backstepping Technique. *Proceedings of the 45th IEEE Conference on Decision and Control*, (pp. 1515–1520).
URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4177802

Naidoo, Y., Stopforth, R., y Bright, G. (2011). Quad-Rotor Unmanned Aerial Vehicle Helicopter Modelling & Control. *INTECH International Journal of Advanced Robotic Systemsbotic Systems*, *8*, 139–149.

Nicol, C., Macnab, C., y Ramirez-Serrano, A. (2008). Robust Neural Network Control of a Quadrotor Helicopter. In *Proceedings of CCECE 2008, the Canadian Conference on Electrical and Computer Engineering*, 1, (pp. 1233–1238). IEEE.
URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4564736

Rabhi, a., Chadli, M., y Pegard, C. (2011). Robust fuzzy control for stabilization of a quadrotor. *2011 15th International Conference on Advanced Robotics (ICAR)*, (pp. 471–475).
URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6088629

Runcharoon, K. (2013). Sliding Mode Control of Quadrotor. (1), 552–557.

Saif, A.-W. a., Dhaifullah, M., Al-Malki, M., y Shafie, M. E. (2012). Modified backstepping control of Quadrotor. *International Multi-Conference on Systems, Sygnals & Devices*, (pp. 1–6).
URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6197975

Santos, M., y López, V. (2010). Intelligent Fuzzy Controller of a Quadrotor. (pp. 0–5).

Santos, O., Castillo, P., y Fantoni, I. (2013). Energy-based nonlinear control for a quadrotor rotorcraft. *1*, 1177–1182.

Voos, H. (2007). Nonlinear and neural network-based control of a small four-rotor aerial robot. *2007 IEEE/ASME international conference on advanced intelligent mechatronics*, (pp. 1–6).
URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4412470

Voos, H. (2009). Nonlinear Control of a Quadrotor Micro-UAV using. (April).

Xu, R. (2006). Sliding Mode Control of a Quadrotor Helicopter. In *Proceedings of the 45th IEEE Conference on Decision & Control*, (pp. 4957–4962).