UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

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

IMPLEMENTACIÓN DE ECUACIONES DINÁMICAS DE VUELO PARA CUADROROTOR EN UN TIVA TM4C123G COMO CONTROLADOR.

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

PRESENTA:
ING. CARLOS FERNANDO ORTEGA NAVA

TUTOR PRINCIPAL
M.I. JESÚS ÁLVAREZ CASTILLO

MÉXICO, D. F. ENERO 2016

JURADO ASIGNADO:

Presidente:	Dra. Navarrete Montesinos Margarita
Secretario:	Dr. Prado Molina Jorge
Vocal:	M. I. Álvarez Castillo Jesús
1 er. Suplente:	Dr. De La Rosa Nieves Saúl
2 do. Suplente:	Dra. Velasco Herrera Graciela

México D.F., Enero de 2016
Ciudad Universitaria
Facultad de Ingeniería

TUTOR DE TESIS:

M.I. Jesús Álvarez Castillo

FIRMA

RESUMEN

En este trabajo se presenta la implementación de algoritmos y ecuaciones dinámicas para el control de un cuadrorotor, utilizando una tarjeta de desarrollo Tiva C como controlador de vuelo y una tarjeta de expansión (Sensor Hub) como sensores de navegación.

Se presentan las ecuaciones: del modelo dinámico del cuadrorotor, del modelo del motor, del controlador de orientación para el vuelo estacionario, y del controlador PD como fundamentos de la parte teórica.

Para la implementación se parte de los requerimientos, especificaciones y restricciones. Incluye un diseño para la integración de todos los componentes del cuadrorotor; algoritmos para la programación y operación de los controladores electrónicos de velocidad y traducción de los diagramas de flujo a un programa en lenguaje C. Se realizan pruebas experimentales y se evalúa el desempeño del cuadrorotor.

Este trabajo es parte del desarrollo de un nuevo piloto automático el cual nos llevará a implementaciones alternativas en un cuadrorotor para futuras aplicaciones.

Dedicado a

mi querida familia Araceli, Agustín, Karla y Nadia.

AGRADECIMIENTOS

Quiero agradecer a Dios que me brinda salud y vida

En especial a mi mama Araceli Nava quien me ha inculcado valores, principios, disciplina con su ejemplo, me ha dado todo su amor, cuidado y confianza sin descanso durante toda mi vida. A mi papa Agustín Ortega quien me enseño liderazgo e inculco el orgullo y amor por la UNAM. A mis hermanas Areli y Nadia por ayudarme a ser mejor persona, y valoraré cada instante que compartimos juntos y mientras esté vivo cuidare de ustedes como prometí a papa.

Al M. I. Jesús Álvarez por las clases en el semestre en curso, por dirigir y apoyarme durante mi estancia en la maestría, así como impulsar mis logros académicos y motivar proyectos a futuro. También a la Dra. Graciela Velasco por el tiempo dedicado en cada clase, asesorías, aportaciones en este proyecto de tesis y apoyo la realización de un artículo y presentación para el congreso SOMIXXX.

Por supuesto quiero agradecer a la Dra. Margarita Navarrete, al Dr. Jorge Prado por compartir su experiencia, dedicarle el tiempo necesario revisando cuidadosamente y proponiendo mejoras en mi trabajo de Tesis. También gracias al Dr. Saúl de la Rosa por las asesorías, revisiones, consejos y aportes en este trabajo.

Finalmente quiero agradecer al Conacyt por el apoyo financiero para realizar mis estudios de maestría.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1 INTRODUCCIÓN

Los sistemas aéreos no tripulados (UAS Unmaned Aereal System), se define como aquel formado por el conjunto de uno o más vehículos aéreos no tripulados (UAV Unmaned Aereal Vehicle), elementos útiles para su operación (estación de recarga, sistema de comunicación, transporte, lanzamiento etc.) y uso para el que fue diseñado.

Actualmente está en curso una evolución respecto al desarrollo de UAS gracias a la tendencia de miniaturización de sistemas electrónicos como GPS (Global Positioning System), GPRS (General Packet Radio Services), sensores de navegación, cámaras, radiotransmisores, microcontroladores etc. El uso de estos sistemas se está difundiendo en todo el mundo con una amplia variedad de usos y va desde las aplicaciones militares a civiles.

Para el ámbito militar los usos se dividen en tres áreas (Tabla1-1) una acuática, una terrestre y una aérea.

Tabla 1-1 Aplicaciones militares de los UAVS

Naval
Detección de flotas enemigas.
Señuelo de misiles emitiendo señales artificiales.
Inteligencia electrónica.
Retransmisión de señales de radio.
Protección de los puertos de ataques en altamar.
Colocación y seguimiento de boyas de sonar y posiblemente otras formas de guerra antisubmarina.
Armada
Reconocimiento.
Inspección de la actividad enemiga.
Monitoreo de armas nucleares, biológicas o químicas contaminantes (NBC).
Inteligencia electrónica.
Detección y seguimiento de blancos.
Ubicación y detección de minas terrestres.
Fuerza aérea
Vigilancia de largo alcance y grandes alturas.
Sistema de interferencia de radar y destrucción del sistema.
Inteligencia electrónica.
Seguridad del aeródromo base.
Evaluación de los daños del aeródromo.
Eliminación de bombas sin explotar.

Para usos civiles entre los que destacan (Tabla 1-2):

Tabla 1-2 Aplicaciones civiles de los UAS

Fotografía aérea	Video y fotografía.
Agricultura	Monitoreo de cultivo y fumigación, vigilancia y dirección del rebaño.
Guardia costera	Monitoreo, búsqueda y rescate en las líneas de las costas
Conservación del medio ambiente	Vigilancia de la tierra y contaminación.
Aduanas	Vigilancia para las importaciones ilegales.
Compañías eléctricas	Inspección de las líneas de transmisión.
Servicios de bomberos y forestal	Detección de incendios y control de incidentes.
Pesca	Protección de pesca.
Compañías de gas y petroleras	Inspección de tierras y seguridad de tuberías.
Servicios de información	Información reciente de noticias, fotos etc.
Instituciones de bote salvavidas	Investigación de incidentes, guía y control.
Autoridades locales	Inspección y control de desastre.
Servicios meteorológicos	Muestreo y análisis de la atmosfera para la previsión del clima.
Agencias de tráfico	Monitoreo y control del tráfico en calles y avenidas.
Inspección de mapas	Fotografía aérea para cartografía.
Policía	Búsqueda de personas desaparecidas, vigilancia de incidentes.

Los términos que actualmente se usan para clasificar los UAV cubren una amplia gama de sistemas (R. Austin et al, 2010). Que son clasificados de acuerdo a su altitud, duración en vuelo continuo y peso (Tabla 3-1):

Tabla 1-3 Clasificación de los UAV

Altitud grande y larga duración (**HALE** High altitude long endurance). Sobre los 15 000 m de altitud y mayor a 24 hrs de vuelo continuo. Usados comúnmente por fuerzas armadas.
Altitud media y larga duración (**MALE** Medium altitud long endurance). Entre 5000 y 15000 metros de altitud y duración de vuelo 24hrs.
UAV táctico (**TUAV** Medium Range or Tactical UAV). Con intervalos de 100 y 300 Km. Usados por fuerzas navales o armada.
UAV de rango cerrado (**Close-Range UAV**). Usualmente opera en rangos cercanos a los 100Km. Tiene usos militares y civiles.

Mini UAV o **MUAV**. Por debajo de 20Kg capaz de operar en rangos de hasta 30 Km.
Micro UAV o **MAV**. Fue originalmente definida con hélices no más grandes de 150 mm de radio y un peso por debajo de los 2Kg. Su uso principalmente es para ambientes urbanos.
Nano vehículos aéreos (**NAV** Nano air vehicles). Del tamaño de una semilla.

Dentro de la categoría de los MUAV están los cuadrorotores que son aeronaves de ala rotatoria también conocidos como helicópteros de cuatro motores o multirotor (J. Anderson et al, 2000). Estos hacen despegues verticales, vuelos estacionarios y maniobras de vuelo en espacios reducidos por lo que son especialmente útiles en toma de video y fotografías, entrega de paquetes, vigilancia e inspección. Estas características en el desempeño del cuadrorotor son objeto de muchos estudios y los usos generalmente se implementan en la industria.

El único desarrollo que se conoce de las configuraciones en tamaño MUAV y MAV es por una serie de universidades. Entre las que destacan por sus avances en cuanto al desempeño en maniobras de vuelo, técnicas de navegación e inspección del entorno.

En la universidad de Pennsylvania trabajan en modelos matemáticos para cuadrorotores, hexarotores, y otros multirotores MUAV y MAV. Esto va desde análisis de modelos dinámicos para vuelos controlados, vuelos con acrobacias agresivas, navegación, vuelos en modo cooperativo (muchas unidades UAV en equipo) y otros tipos de control. La propuesta de su estudio está basada en la respuesta del cuadrorotor en base a un sistema de captura de movimiento formado por una constelación de varias cámaras detectoras de movimiento instaladas alrededor de un cuarto o área donde vuela el cuadrorotor para estimar su posición y velocidad. Es un sistema muy preciso y por esta razón sus cuadrorotores pueden hacer maniobras de vuelo agresivas. La desventaja es que su sistema es muy costoso con respecto a los que calculan su posición y velocidad con GPS y sensores para navegación. El uso del cuadrorotor debe ser dentro del cuarto con cámaras. Otra desventaja es que para aplicaciones civiles, este tipo de maniobras agresivas son peligrosas por lo que tendrían que ser reguladas con base en normas legales para un uso responsable y evitar accidentes. (D. Mellinger et al, 2010)

La universidad de München que tiene un grupo de visión por computadora. Desarrollaron un sistema MUAV que permite al cuadrorotor localizar y navegar de manera autónoma en lugares desconocidos sin utilizar GPS. Su propuesta utiliza una cámara monocular a bordo y no requiere de

marcadores o sensores externos. Su método consiste en tres componentes principales. El primero; un sistema para la estimación de posición conocido como localización simultánea y mapeo (SLAM Simultaneous localization and mapping). Segundo; emplean un filtro Kalman extendido el cual incluye un modelo completo del drone y la dinámica de control para fusionarlo y sincronizar todos los datos y compensar los retardos derivados de los procesos de comunicación y cálculos necesarios, para alimentar el siguiente proceso. En tercer lugar un controlador PID (Proporcional Integral Derivativo) para controlar la posición y la orientación del cuadrorotor. Con este método generan un mapa a escala real del entorno que visualiza el cuadrorotor (D. Cremers et al, 2014).

Para el estudio de un quadrorotor MUAV o MAV se requiere de conocimiento en muchas áreas como la electrónica, mecánica, control, dinámica, aerodinámica, comunicaciones etc. Las características y los requerimientos para el diseño e implementación de un cuadrorotor dependerán de las necesidades que se tengan para cada uso y del presupuesto con que se disponga.

En el Laboratorio de Transductores de la Facultad de Ingeniería se requiere monitorear lámparas de alumbrado público en zonas urbanas y rurales. De esta forma surge la necesidad de contar con un aparato que nos permita realizar esta tarea. Para este fin se propone el diseño de un cuadrorotor basado en un piloto automático construido mediante un algoritmo embebido en una tarjeta TivaC.

De esta forma no se depende de un piloto automático comercial en el cual se tienen características muy específicas que podría no concordar con nuestras necesidades. Con el fin de encaminar hacia esta meta se propone la implementación de ecuaciones dinámicas de vuelo para cuadrorotor en un TivaC como controlador de vuelo, que es parte esencial de un piloto automático.

1.1 Objetivo principal

Diseño e implementación de algoritmos y ecuaciones dinámicas para cuadrorotor en un TivaTM4C123G como controlador de vuelo.

1.2 Objetivos particulares

Proponer un algoritmo para el control de orientación del cuadrorotoSK450 para utilizarlo en pruebas experimentales.

Implementar la tarjeta de desarrollo TivaC Serie TM4C123G como controlador de vuelo, que forma parte de un piloto automático para el cuadrorotor elegido, utilizando dicha tarjeta ya que tiene un alto desempeño y accesorios adecuados para esta finalidad.

Comprobar el funcionamiento del sistema implementado en el cuadrorotor.

1.3 Justificación

En el Laboratorio de Transductores de la Facultad de Ingeniería de la UNAM, se genera una línea de estudio en el diseño de piloto automático, además de la formación de recursos humanos altamente calificados en esta área. La finalidad de este trabajo es entender la dinámica del cuadrorotor y sobre la base de esto, implementarla en un sistema embebido, teniendo un controlador de vuelo propio, no comercial, que es parte de un piloto automático, próximamente a implementar.

1.4 Metodología

Recopilar información de los temas y dispositivos más importantes y recientes que constituyen los cuadrorotores. Buscando en fuentes primarias como artículos de revista, publicaciones en congreso, tesis, libros, videos etc.

Estudiar y probar la tarjeta de desarrollo TivaC Serie TM4C123G.

Estudiar y seleccionar un modelo de ecuaciones dinámicas para cuadrorotor.

Diseñar algoritmos utilizando los modelos de ecuaciones dinámicas para cuadrorotor.

Codificar los algoritmos planteados en lenguaje C.

Depurar el software creado.

Probar el software en el cuadrorotor.

Despliegue del cuadrorotor.

Evaluar los resultados obtenidos.

1.5 Metas

Obtener un modelo dinámico para el cuadrorotor.

Implementar el programa para la tarjeta de desarrollo TivaC.

Obtener el controlador de orientación del piloto automático.

Despliegue del cuadrorotor.

Evaluación de resultados obtenidos.

1.6 Resultados esperados

Todos los elementos del cuadrorotor seleccionado se deberán integrar y acoplar de manera que no exista falla en la conexión entre ellos para un correcto funcionamiento del hardware.

La tarjeta de desarrollo TivaC Launc Pad deberá comunicarse y controlar cada uno de los elementos del cuadrorotor como son los sensores de navegación, barómetro y ESC (Controlador Electrónico de velocidad); tomar lecturas de los sensores: giróscopos, acelerómetros, magnetómetros y de presión por medio de protocolo I^2C (Inter Integrated Circuit); controlar los ESC por medio de señales PWM (Modulación por ancho de pulso); realizar operaciones matemáticas con base en las ecuaciones dinámicas y de control propuestas para el cuadrorotor.

Para comprobar las lecturas tomadas por los sensores de navegación y barómetro, la tarjera de desarrollo TivaC deberá enviar los datos por medio de una comunicación serial, utilizando una interfaz en la computadora para visualizar los datos.

Se harán mediciones experimentales para caracterizar los motores y hélices para modelar los motores en el cuadrorotor, ya que no se encontró información al respecto por parte del fabricante. Con base en estas mediciones experimentales, las ecuaciones dinámicas y de control se propondrá un algoritmo para estabilizar el cuadrorotor.

Se espera que las tareas y ecuaciones del algoritmo traducido a código fuente escrito en lenguaje C para el controlador del cuadrorotor, nunca rebase las capacidades de la tarjeta de desarrollo en cuanto a memoria de programa, memoria de datos y velocidad. Deberán sobrar recursos para expandir las capacidades del cuadrorotor en el futuro.

2 ESTADO DEL ARTE

De acuerdo a la metodología planteada, se presenta el estado del arte referente al estudio del modelo dinámico para cuadrorotores (MAV, MUAV), diseños de controladores PID (Proporcional Integral Derivativo), BS (Backsteeping), SMC (Control por modos deslizantes), FBC (Linealización por retroalimentación), visión por computadora y lógica difusa con base en los modelos dinámicos presentados. También algunos artículos con enfoque práctico para la implementación de algoritmos de controladores para determinar la posición y mantener la orientación en el espacio deseado, además de la generación de trayectorias 3D para el cuadrorotor.

En este trabajo (A. Salih et al, 2010) se presenta el modelo de un aeronave de despegue y aterrizaje vertical (VTOL vertical take-off and landing) conocido como cuadrorotor. Se presenta un modelo dinámico para cuadrorotor que es usado para diseñar un controlador estable y preciso. Se explica el desarrollo del controlador PID para obtener la estabilidad en vuelo. En las simulaciones que realizaron en MATHLAB® simulink se muestra como el controlador PID (Proporcional Integral Derivativo) estabiliza un cuadrorotor que lo lleva a una posición deseada considerando las rotaciones correspondientes de rotación de cada uno de los ejes, en un tiempo que va de 3 a 5 segundos.

En este trabajo (S. Bouabdallah et al, 2004) se presentan dos técnicas aplicadas a un cuadrorotor. Una aproximación clásica PID, asumiendo una dinámica simplificada y una técnica moderna cuadrática lineal (LQ linear cuadratic) basada en un modelo más completo. Se realizan numerosas simulaciones y experimentos en un banco de prueba. El controlador LQ era menos dinámico, no fue capaz de realizar un vuelo libre. Por el contrario, el uso del enfoque clásico PID era mejor en desempeño. Algunas perturbaciones eran introducidas por los cables de alimentación y por intervenciones para evitar que el cuadrorotor colisione con la pared. El siguiente objetivo es mejorar el control de posición y desarrollar un vehículo totalmente autónomo.

En este trabajo (C. Diao et al, 2011) se propone una nueva ley de control adaptable para cuadrorotor sujeto a incertidumbres asociadas con la masa del vehículo, matriz de inercia y coeficientes aerodinámicos amortiguados. A través de un análisis basado en Lyapunov, demuestran que la posición y los errores en la guiñada son forzados cerca del cero. Los resultados numéricos muestran el buen desempeño del controlador. El autor propone en trabajos futuros desarrollar controladores adaptables, que proporcionarán al cuadrorotor, la habilidad de rechazar la perturbación. El tiempo de respuesta es entre 3 a 5 segundos.

En este trabajo (T. Madani et al, 2006) se presenta un modelo dinámico no lineal para cuadrorotor. Un controlador backsteeping es propuesto para estabilizar todo el sistema y es capaz de controlar a un cuadrorotor a la trayectoria y orientación deseada. El tiempo de respuesta para los ángulos es de 5 a 15 segundos.

En el trabajo de (G. Ratto et al, 2008). Se presenta una estrategia de control robusta no lineal para resolver el problema de seguimiento de trayectorias en vehículos aéreos. El modelo dinámico se obtiene de las ecuaciones Euler-Lagrange. La estructura de control se realiza a través de un controlador no lineal para estabilizar los movimientos de rotación y una ley de control basada en backstepping para aproximar a la trayectoria de referencia. Se presentan los resultados simulados con incertidumbre para corroborar la efectividad del controlador propuesto. Las graficas muestran un buen desempeño y tiempo de respuesta por debajo de los tres segundos, permitiendo una muy buena aproximación en la trayectoria mostrada.

En este trabajo (R. Xu et al, 2006) se presenta un método de control de vuelo para cuadrorotor basado en modos deslizantes. El controlador permite al cuadrorotor realizar los movimientos de ángulo balanceo, cabeceo y guiñada, y los estabiliza a valores deseados. La ventaja de los modos deslizantes es su insensibilidad a incertidumbres y otras perturbaciones (N. Sinan et al, 2015)). Las simulaciones muestran las leyes de control que estabilizan un cuadrorotor en cualquier posición de ángulo de guiñada.

En este trabajo (D. Lee et al, 2009) se presentan dos tipos de controladores no lineales para UAV cuadrorotor. Un controlador por linealización retroalimentada (LF linealization feedback) fue derivado de una manera convencional con una dinámica simplificada para reducir el número de términos derivativos de alto orden involucrados en el proceso de diseño. Como alternativa se introduce una nueva aproximación para el controlador por modos deslizantes.

En este trabajo (Z. Zuo et al, 2012) se presenta el diseño de un controlador de vuelo capaz de estabilizar la posición y también una trayectoria con precisión para una aeronave cuadrorotor en presencia de incertidumbres y perturbaciones externas. Se propone un algoritmo de control basado en la relación entre la exactitud y la aceleración lineal, utilizando un aproximador adaptativo para estimar los parámetros aerodinámicos desconocidos, las perturbaciones externas y un diferenciador de seguimiento lineal para eliminar la supuesta separación de escala de tiempo entre la dinámica de la posición en el diseño del sistema de control. Se llevan a cabo extensas simulaciones que mostraron que el seguimiento de la trayectoria se puede realizar en el sistema de bucle cerrado a pesar de existir incertidumbres en los parámetros de aerodinámica y la perturbación exógena en un entorno complicado.

En este trabajo (G. Hoffmann et al, 2008), el banco de pruebas de Stanford, control de pruebas multiagente para cuadrorotor (STARMAC Stanford Testbed of Autonomous Rotorcraft for Multi-Agent Control), ha sido desarrollado para probar algoritmos novedosos que permiten el funcionamiento autónomo de vehículos aéreos. Se desarrolla un algoritmo para el seguimiento de trayectoria con ambientes desordenados para la plataforma STARMAC. El controlador de seguimiento de trayectoria está diseñado para visualizar dos problemas. Primero, al aceptar como entradas una ruta de puntos y velocidades deseadas, la entrada de control puede ser actualizada con frecuencia para rastrear con precisión la trayectoria deseada. Segundo, la planificación de una trayectoria se produce como un proceso separado en una escala temporal más lenta. El seguimiento de trayectoria fue usado experimentalmente demostrando la realización de las rutas con una precisión de 10 cm en interiores y 50 cm en exteriores. En un futuro se investigará la posibilidad de generar trayectorias con cambio de puntos, considerando obstáculos para modificar en vuelo la velocidad y

maniobrar en espacios cerrados. La propuesta del algoritmo será implementada en la plataforma STARMAC.

En el trabajo (D. Gurdan et al, 2007), se describe un cuadrorotor eficiente, fiable y robusto para la navegación. Actualmente las plataformas son controladas en bajas frecuencias (0 a 50 Hz) debido a limitaciones de hardware y software. Esto causa incertidumbre en el control de posición y un comportamiento inestable durante maniobras agresivas. Este robot volador ofrece una frecuencia de control a 1 KHz y motores sin escobillas de corriente directa. Con un enfoque de diseño minimalista, basado en un número pequeño de componentes de bajo costo. Su rendimiento robusto se consigue mediante el uso de algoritmos altamente optimizados pero simples y confiables. El cuadrorotor es pequeño, ligero y puede transportar cargas útiles de hasta 350 g. La clave de esta innovación es una plataforma capaz de generar una tasa de actualización alta con controladores optimizados. En el futuro se incluirán pruebas de la plataforma en combinación con sensores de aceleración para una dinámica y maniobras agresivas. La segunda generación debe ofrecer vuelos más largos con capacidades de carga más grandes.

En este trabajo (G. Scaglia et al, 2008), se propone el control de un robot móvil usando un modelo lineal con incertidumbres. Este controlador es diseñado con un control robusto para que el robot móvil muestre un buen desempeño alrededor de un punto. Las estructuras de control pueden ser diseñadas e implementadas sin gran dificultad. Las pruebas experimentales fueron realizadas en un robot móvil PIONER 2DX. La eficacia y viabilidad se ponen a prueba en la práctica a través de experimentos. Se aplica la formulación de algoritmos basada en la técnica presentada para evitar obstáculos inesperados. En trabajo futuro se aplicarán los algoritmos para una formación de robots móviles (la trayectoria de robots móviles así como la posición relativa de cada uno está establecida) que serán analizados. En estos sistemas, diversos robots se utilizan para llevar a cabo algunas tareas complejas como por ejemplo vigilancia, rescate y transporte de cargas etc.

En (E. Altug et al, 2005) se propone una estabilización de cuadrorotor basada en un sistema de visión y un método de control de trayectoria. Un método que usa dos cámaras es introducido para estimar un total de 6 grados de libertad del cuadrorotor. Una cámara abordo, y la otra se

encuentra en el suelo. La estimación del algoritmo se compara con otros métodos en una simulación y demuestra que es menos sensible a errores. Se estudian dos métodos de control, uno utilizando una serie de controladores retroalimentados y otro usando backsteeping como leyes de control. Los controladores propuestos son implementados en un sistema para pruebas y muestran la efectividad del sistema basado en visión para el cuadrorotor. El cuadrorotor puede hacer equipo con un robot móvil. El robot móvil será responsable del transporte, comunicación, asistencia para el aterrizaje y proporcionará los cálculos de información visual. Estas funciones serán útiles para la detección, caza y otras tareas de cooperación aeroterrestre.

En (S. Bouabdallah et al, 2004) este trabajo se describe un enfoque propio para microcuadrorotor de despegue y aterrizaje vertical (VTOL Vertical Takeoff andLanding), llevándolo hacia la total autonomía. Se presenta el diseño mecánico, el modelo dinámico, sensado y control. Se muestra un estudio de micro vehículos voladores existentes y se hace una comparación del micro VTOL en términos de miniaturización. Se introduce al proyecto SO4 (del Laboratorio de Sistemas Autónomos que se enfoca en los micro VTOL) y se discuten los detalles de robots voladores. Esta incluye el modelo dinámico, la optimización del diseño del vehículo y el control. Los resultados obtenidos en el desarrollo hacia la autonomía del micro VTOL refuerza su convicción de que estos sistemas tienen el potencial como candidatos para los micro vehículos de emergencia.

En este trabajo (J. Normey et al, 2001) presenta una solución sencilla y eficaz para el problema de seguimiento de trayectoria para robot móvil utilizando un controlador PID. El método propuesto utiliza un modelo linealizado simple del robot móvil, compuesto de un integrador y un retardo. El procedimiento de síntesis es simple y permite al controlador PID ser sintonizado considerando el rendimiento nominal y la robustez como especificaciones de control. El procedimiento de síntesis es fácil y las reglas obtenidas son similares al método de Ziegler/Nichols para controladores PID. El ajuste robusto del controlador puede ser tener en cuenta el uso de un solo parámetro de sintonía. El ajuste de este parámetro es intuitivo y permite un buen equilibrio entre rendimiento y robustez, principalmente cuando el proceso exhibe incertidumbres en los tiempos muertos. Finalmente los PD propuestos fueron probados usando un robot móvil y algunos resultaos experimentales han mostrado un buen desempeño a pesar de las incertidumbres del retardo.

En este trabajo (B. Xian et al, 2015), un banco de pruebas de bajo costo para simulación del cuadrorotor UAV e implementación de esquemas de control no lineales. El diseño y pruebas de algoritmos de control de vuelo para cuadrorotor no es una tarea sencilla, debido al riesgo de un posible peligro y daños durante la práctica de vuelo. Se desarrolla HILS (HILS Hardware in the loop simulation) a fin de mejorar la seguridad y eficiencia de la implementación del control de vuelo, a bajo costo y en tiempo real. Para realizar el banco de pruebas HILS, se usa un mini cuadrorotor como cuerpo principal, equipado con una micro unidad AHRS (Alttitude Heading Reference System) y una tarjeta con un DSP (Digital Signal Procesor). El HILS es implementado usando la trajeta xPC. Una computadora compacta PC/104 es utilizada como el equipo de destino y una PC portátil se usa como servidor. Una computadora de escritorio es usada para visualizar el vuelo en el cual corre el entorno Flight Gear (simulador de vuelo) y Google Earth, para visualizar los datos tales como la orientación y trayectoria de vuelo del cuadrorotor. Este banco de datos sirve para simular diversos algoritmos de control de vuelo sin perder seguridad y confianza. Se muestra la eficacia del banco de pruebas en un nuevo algoritmo de control basado en modos deslizantes. Para reducir el costo del banco de pruebas, integrarán algunos sensores nuevos giróscopos, acelerómetros y magnetómetros basados en MEMS (Micro-Electro-Mechanical Systems) en la tarjeta DSP y se desarrollaran algoritmos para fusionar los sensores y obtener mediciones de posición en lugar de la unidad AHRS. Esto proporcionará un buen enfoque para validar diferentes metodologías de fusión de sensores. Como trabajo futuro se investigará un novedoso controlador no lineal para el vuelo del cuadrorotor y se comprobarán esos algoritmos en el banco de pruebas HILS comparándolos contra una verdadera prueba de vuelo.

En este trabajo (H. Xie et al, 2015) se propone una dinámica basada en control visual (IBVS Image-Based Visual Servoing) para un UAV de ala rotatoria. El objetivo del control de movimiento es seguir las líneas paralelas y está motivada por tareas de inspección de líneas eléctricas donde la posición relativa al UAV y la orientación a las líneas son controladas. El diseño está basado en una cámara cuyo movimiento sigue una cámara física abordo, pero que está limitada a apuntar hacia abajo independientemente de los ángulos de rotación y cabeceo. Un conjunto de imágenes es propuesto para las líneas proyectadas en la cámara virtual. Estas características son dadas para simplificar la matriz de interacción que a su vez conduce a un diseño de control IBVS simple. A pesar de que los resultados de la

simulación presentadas, demuestran que el diseño todavía funciona con líneas en un plano no horizontal, no se proporciona ninguna prueba rigurosa.

En este trabajo (J. Gomez et al, 2014) se presenta un enfoque basado en visión, es desarrollado para el vuelo de un mini cuadrorotor. Las cámaras se usan para estimar la posición y otra para estimar la velocidad de traslación del vehículo. El modelo dinámico del mini cuadrorotor se obtiene utilizando el enfoque Euler-Newton. Se presenta un controlador no lineal para un cuadrorotor miniatura basado en una estrategia de control de saturación. Las pruebas experimentales muestran que el cuadrorotor realiza el vuelo usando el sistema de control propuesto, y el potencial de las estrategias basadas en visión por computadora.

En este trabajo (H. Romero et al, 2012) se aborda el problema de la estabilización y posicionamiento local de un cuadrorotor utilizando la visión por computadora. Para estimar la posición del cuadrorotor, combinan las lecturas de la unidad de medición inercial (IMU) y un sistema de visión. En la primera etapa, el sistema de visión es utilizado para estimar el ángulo de posición y guiñada del cuadrorotor, mientras que en la segunda etapa, para estimar la velocidad de traslación. En ambos casos, la IMU proporciona los ángulos de cabeceo balanceo a una tasa de actualización menor a 10 ms por muestra. La técnica usada para estimar la posición del cuadrorotor en la primera etapa combina el enfoque de transformación homogéneo para el proceso de calibración de la cámara y el método de postura-plano para la estimación de la posición. En la segunda etapa, un sistema navegación mediante patrones de movimiento, se desarrolló para estimar la velocidad de traslación y localización de cuadrorotor. Los beneficios de la teoría propuesta en este trabajo es que emplea un sensor de bajo costo (cámara) para la estimación del estado del cuadrorotor. Esta estimación es suficiente para un vuelo estacionario del mismo.

Este trabajo (G. Scaglia et al, 2009) presenta el diseño de cuatro controladores para un robot móvil, de tal manera que el sistema pueda seguir una trayectoria preestablecida. El modelo cinemático del robot móvil se aproxima mediante métodos numéricos. A partir de esta aproximación, las acciones de control para obtener un error mínimo de seguimiento son

calculadas. La simulación y los resultados experimentales en un robot PIONNER 2DX son presentados mostrando el desempeño de los cuatro controladores propuestos. De los resultados experimentales se concluye que la metodología propuesta es muy simple para seleccionar los parámetros del controlador con el fin de lograr un buen rendimiento del sistema durante la navegación del robot móvil. La metodología propuesta para el diseño del controlador puede ser aplicada a otros tipos de sistemas. La precisión requerida del método numérico propuesto para la aproximación del sistema es más pequeña que la que se necesita para simular su comportamiento. La principal contribución es que los cuatro controladores son obtenidos con la misma metodología y no son necesarios cálculos complejos para obtener la señal de control.

En este trabajo (C. Rosales et al, 2014) se presenta el diseño de un controlador que permite a un cuadrorotor seguir una trayectoria en 3D. Se obtiene el modelo dinámico por medio de las ecuaciones de Euler-Lagrange. El modelo es representado con métodos numéricos, con el cual las acciones de control se obtienen para la operación del sistema. El controlador es fácil de manejar para la mayoría de los diseñadores comparado con otros controladores propuestos que necesitan más herramientas matemáticas complejas. La metodología se basa en definir la referencia de la trayectoria en términos de un juego de variables de estado y se determinan los valores deseados variando el tiempo para los estados restantes. La principal ventaja de esta aproximación es la simplicidad del controlador y el uso de ecuaciones discretas en el tiempo, lo cual es útil para su implementación en una computadora. El controlador 3D es probado en una simulación con MATLAB y muestra el buen desempeño, incluso bajo errores paramétricos y mediciones de ruido. Como trabajo futuro el controlador se propone para otros vehículos aéreos y se estudiará la robustez y otras propiedades del controlador.

En este trabajo (N. Sinan et al, 2015) se presenta la aplicación de técnicas PID (Proporcional Integral Derivativo), BS (Backsteeping), SMC (Control por modos deslizantes), FBC (Linealización por retroalimentación), y lógica difusa en un cuadrorotor, las cuales son frecuentemente usadas para probar estudios de investigación. Los métodos han sido evaluados con métricas de comparación y varios conjuntos de parámetros elegidos para cada enfoque. Los resultados ilustran los diferentes aspectos de las estrategias de control y las observaciones se tabulan de forma comparativa. En las contribuciones de

este trabajo se describe un conjunto de muestras de experimentos de control de retroalimentación para este tipo de cuadrorotor, para reforzar la noción lineal y no lineal con un estudio completo de literatura. Proporcionando a los investigadores, una visión general y algoritmos de control adaptativos para la estabilización y la trayectoria de seguimiento para el cuadrorotor, que describe un conjunto de criterios de rendimiento frecuentemente utilizados en un UAV.

En este trabajo (M. Arittanan et al, 2014) se presenta el diseño de un controlador para múltiples entradas y múltiples salidas (MIMO multi-input-multi-output), los sistemas no lineales con incertidumbres son lo más importante en este trabajo. La contribución se centra en el diseño y análisis de un control backsteeping adaptativo inteligente (IABC) para un cuadrorotor perturbado por incertidumbres y perturbaciones externas. El enfoque de diseño se basa en una técnica backsteeping y utiliza una red de función de base radial neuronal (RNFBR) como aproximador de perturbación. Se agrega un compensador de lógica difusa para eliminar el error de aproximación producido por la función de base radial. La efectividad del IABC es verificado con algunos resultados de simulación en MATLAB.

En este trabajo (D. Mellinger et al, 2010) se describe el diseño, control y metodologías para lograr una maniobra de enganche (similar a colocar la ropa en un perchero). Como plataforma base se usa un cuadrorotor Hummingbird de 55 cm y peso de aproximadamente 500 g incluyendo batería. Se usa un PID optimizado para el control de orientación y un sistema de captura de movimiento VICON para estimar la posición del cuadrorotor. El software de control es integrado con un sistema de captura de movimiento y accesado vía código abierto con un ROS (Sistema operativo para robot). Otra característica es que para sensar los movimientos del cuadrorotor utiliza IMU a una taza de 1 KHz como en Daniel Gurdan (2007) para disminuir la incertidumbre y hacer maniobras más rápidas. Una importante contribución en este trabajo es el de un sistema de estimación de estado y localización basado en captura de movimiento. Se plantea incorporar un sistema de reconocimiento visual que automáticamente identifique caminos y lugares para que realice el enganche.

En este trabajo (D. Mellinger et al, 2012) se estudia el problema de diseño factible de trayectorias y controladores que conducen a un cuadrorotor a una posición deseada en el espacio. Enfocándose en el desarrollo de una familia de trayectorias definidas como una secuencia de segmentos, con cada controlador parametrizado por una posición como objetivo. Cada controlador es desarrollado del modelo dinámico del cuadrorotor y después iterativamente refinado a través de experimentos sucesivos para dar cuenta de los errores en el modelo dinámico y ruido de los actuadores y sensores. Se muestra que este enfoque permite el desarrollo de trayectorias y los controladores realizan maniobras agresivas, como volar a través de huecos estrechos, verticales y posarse en superficies invertidas con alta precisión.

Se cree que mediante la representación de familias de trayectorias por controladores especializados, la alta dimensión del planteamiento del problema podría ser reducido, por lo que se está trabajando activamente en esta área de investigación. Finalmente, se están considerando métodos más avanzados para la optimización y adaptación de parámetros de trayectoria para hacer frente a las diferencias entre el modelo analítico y la realidad.

En este trabajo (V. Singh et al, 2013) se presenta un modelo matemático para el desempeño de un motor sin escobillas de corriente directa (BLDC brushless direct current). Se diseña un controlador PID para el modelo del motor BLDC y se encuentran las ganancias utilizando el método de Zieger/Nichols. Se muestra el desempeño del controlador utilizando MATLAB/SIMULINK. El modelo presentado es una aproximación a los vendidos comercialmente. El controlador diseñado refiere a una implementación donde se cuenta con la lectura a través de sensores Hall para determinar la posición del rotor.

En este trabajo (C. Powers et al, 2013) se modelan los efectos aerodinámicos de proximidad a superficies horizontales y cercano a otros vehículos (formaciones en equipo) para MAV cuadrorotor. Se explica la dependencia de la fuerza de sustentación generada por las hélices y empíricamente se determinan los coeficientes para el modelo del cuadrorotor presentado. Se muestra como se debe puede mejorar el desempeño de vuelo en un cuadrorotor sin sensores adicionales utilizando efectos aerodinámicos. Con equipos de vehículos que vuelan a 15,10, 6 cm de altura a una velocidad constante de 6 cm/s con una separación entre cada vehículo de 60 cm y grabando las velocidades requeridas de los rotores durante el experimento, se crea un mapa de la superficie del terreno usando el efecto de proximidad

de superficie. En el futuro se esperan aplicar los resultados de este trabajo para la estimación de problemas más complicados, el cual permita una operación de cuadrorotores en diferentes ambientes. El estudio de la influencia aerodinámica en la operación de los multirotores permitirán a los ingenieros hacer mejores diseños y por tanto, controles.

De los anteriores artículos se toman los que de manera explícita definen valores de frecuencia utilizados en sus trabajos experimentalmente, ya sea en bancos de prueba o vuelos reales (Tabla2-1). Donde se puede observar que para los sensores de navegación, los valores que normalmente se utilizan de frecuencia de muestreo están entre los 100 y 200 Hz. Los que aparecen con una frecuencia de 1 Kz es porque manejan microcontroladores más avanzados que pueden obtener esa taza de muestreo y procesarla gracias a sensores más rápidos y por tanto, costosos. También se concluye que uno de los controladores más utilizados en la práctica es el PID ya que consume menos recursos al implementarlo en una computadora o micocontrolador (N. Sinan et al, 2015). Los sistemas que utilizan cámara para la estabilización trabajan a frecuencias entre 15 y 30 Hz y necesitan computadoras o microcontroladores con una capacidad de almacenamiento superior a 32 KB con que cuenta la TIVAC para almacenar y procesar las imágenes capturadas.

Tabla 2-1 Cuadro de resumen

Referencia	Métodos de control	Prueba experimental	Frecuencia cámara	Frecuencia sensado navegación	Categoria UAV
(G.Hoffmann et al,20XX)	PID	si	no	100 Hz	MUAV
(D. Gurdan et al, 2007)	PD	si	no	1 KHz	MUAV
(E. Altug et al, 2005)	retroalimentación backsteeping visión	si	15-30 Hz	-	MUAV
(B. Xian et al, 2015)	PD y modos deslizantes	si	no	100 Hz	MAV
(H. Xie et al, 2015)	PID y backsteeping	no	30 Hz	200 Hz	MUAV
(J. Gomez et al, 2014)	visión retroalimentación	si	-	100 Hz	MUAV
(H. Romero et al, 2012)	visión	si	15 Hz	60 Hz	MUAV
(D. Mellinger et al, 2010)	PID	si	no	1 KHz	MAV

3 DINÁMICA DEL CUADROROTOR

En este capítulo se desarrolla el modelo dinámico del cuadrorotor utilizando las ecuaciones de Euler.

Definimos los sistemas de referencia; el móvil de coordenadas $\left(e_x^2, e_y^2, e_z^2\right)$, fijado al cuadrorotor en un punto cualquiera P (polo), el sistema de referencia de centro de masa c de coordenadas $\left(e_x^3, e_y^3, e_z^3\right)$, y el inercial 0 de coordenadas $\left(e_x^1, e_y^1, e_z^1\right)$, (figura. 3-1).

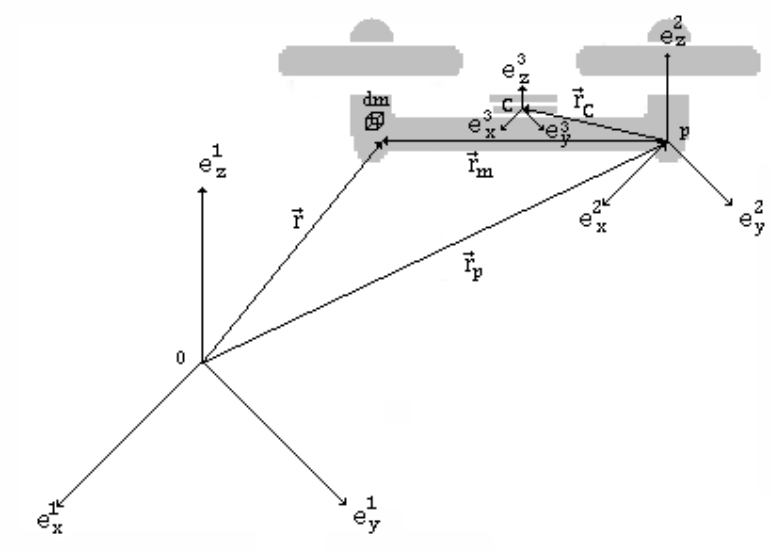

Figura 3.1 Sistemas de referencia

Se define un vector $\vec{r}$ que va dirigido a un diferencial de masa dm del cuadrorotor, un vector $\overrightarrow{r_p}$ que define un punto p como referencia al cuadrorotor y un vector $\overrightarrow{r_m}$ (masa) que va del punto p a un diferencial de masa dm.

3.1 Momento angular

El momento angular $\overrightarrow{L_0}$ de nuestro cuadrorotor se define:

$$\overrightarrow{L_0} = \int_m \vec{r} \times \dot{\vec{r}} dm \tag{3.1}$$

Donde

$\vec{r}$ - vector de posición.

$\dot{\vec{r}}$ - velocidad absoluta.

De la figura 3-1 se obtienen las ecuaciones (3.2) y (3.3).

$$\vec{r} = \overrightarrow{r_p} + \overrightarrow{r_m} \tag{3.2}$$

$$\dot{\vec{r}} = \dot{\overrightarrow{r_p}} + (\vec{\omega} \times \overrightarrow{r_m}) \tag{3.3}$$

Donde

ω - velocidad angular absoluta del cuerpo.

Sustituyendo (3.2) y (3.3) en (3.1) tenemos.

$$\overrightarrow{L_0} = \int_m \left[(\overrightarrow{r_p} + \overrightarrow{r_m}) \times \left(\dot{\overrightarrow{r_p}} + (\vec{\omega} \times \overrightarrow{r_m}) \right) \right] dm \tag{3.4}$$

Desarrollando los términos dentro de la ecuación.

$$\overrightarrow{L_0} = \int_m \left[\overrightarrow{r_p} \times \left(\dot{\overrightarrow{r_p}} + (\vec{\omega} \times \overrightarrow{r_m}) \right) + \overrightarrow{r_m} \times \dot{\overrightarrow{r_p}} + \overrightarrow{r_m} \times (\vec{\omega} \times \overrightarrow{r_m}) \right] dm \tag{3.5}$$

Del triple producto vectorial dado en el último término de la expresión.

$$\overrightarrow{L_0} = \int_m \left[\overrightarrow{r_p} \times \left(\dot{\overrightarrow{r_p}} + (\vec{\omega} \times \overrightarrow{r_m}) \right) + \overrightarrow{r_m} \times \dot{\overrightarrow{r_p}} + (\overrightarrow{r_m} \cdot \overrightarrow{r_m})\vec{\omega} - (\overrightarrow{r_m} \cdot \vec{\omega})\overrightarrow{r_m} \right] dm \tag{3.6}$$

Al integrar se obtiene la siguiente igualdad:

$$\overrightarrow{L_0} = \overrightarrow{r_p} \times \left(\dot{\overrightarrow{r_p}} + (\vec{\omega} \times \overrightarrow{r_c}) \right) M + \left(\overrightarrow{r_c} \times \dot{\overrightarrow{r_p}} \right) M + [(\overrightarrow{r_c} \cdot \overrightarrow{r_c})\vec{\omega} - (\overrightarrow{r_c} \cdot \vec{\omega})\overrightarrow{r_c}] \tag{3.7}$$

Donde M es la masa total que coincide con la masa del cuadrorotor.

La expresión $\left(\dot{\vec{r_p}} + (\vec{\omega} \times \vec{r_c})\right)$ es la velocidad absoluta del centro de masa, por tanto, $\left(\dot{\vec{r_p}} + (\vec{\omega} \times \vec{r_c})\right)M$ representa el momento lineal absoluto del cuadrorotor con respecto al sistema de referencia inercial.

El primer término de la ecuación (3.7) representa el momento angular con respecto a 0 debido a la traslación al centro de masa del cuadrorotor y el último término representa el momento angular causado por la rotación del cuadrorotor. El último término $[(\vec{r_c} \cdot \vec{r_c})\vec{\omega} - (\vec{r_c} \cdot \vec{\omega})\vec{r_c}]M$ corresponde a la diada de un tensor y obtenemos la siguiente igualdad.

$$[(\vec{r_c} \cdot \vec{r_c})\vec{\omega} - (\vec{r_c} \cdot \vec{\omega})\vec{r_c}]M = J \cdot \vec{\omega} \tag{3.8}$$

Donde J corresponde al tensor de inercia.

La expresión final de momento angular para el cuadrorotor es el siguiente:

$$\vec{L_0} = \vec{r_p} \times \left(\dot{\vec{r_p}} + (\vec{\omega} \times \vec{r_c})\right)M + \left(\vec{r_c} \times \dot{\vec{r_p}}\right)M + J \cdot \vec{\omega} \tag{3.9}$$

Para la ecuación (3.9) los términos $\vec{r_p} \times \left(\dot{\vec{r_p}} + (\vec{\omega} \times \vec{r_c})\right)M + \left(\vec{r_c} \times \dot{\vec{r_p}}\right)M$ nos expresan el movimiento de traslación y los términos $J \cdot \vec{\omega}$ nos expresan el movimiento de rotación en donde J es el tensor de inercia, definido por:

$$J = \begin{bmatrix} I_{11} & -I_{12} & -I_{13} \\ -I_{21} & I_{22} & -I_{23} \\ -I_{31} & -I_{32} & I_{33} \end{bmatrix} \tag{3.10}$$

3.2 Momento de fuerza

Se utiliza el teorema de momento angular (J. Witenburg (2007)). Para obtener las ecuaciones de movimiento. *En un sistema de masa arbitraria. La derivada en el tiempo del momento angular absoluto con respecto a un punto de referencia 0 fijo en un espacio inercial, equivale al momento resultante con respecto al mismo punto de referencia.*

$$\dot{\overrightarrow{L^0}} = M^0 \tag{3.11}$$

Donde:

$\dot{\overrightarrow{L^0}}$ - La derivada con respecto al tiempo.

M^0 - Momentos de fuerzas totales.

Derivando $\overrightarrow{L_0}$ con respecto del tiempo para la ecuación (3.1).

$$\frac{d\overrightarrow{L_0}}{dt} = \frac{d}{dt}\left[\int_m \vec{r} \times \dot{\vec{r}} dm\right] \tag{3.12}$$

$$\dot{\overrightarrow{L^0}} = \int_m \left[\frac{d}{dt}\left(\vec{r} \times \dot{\vec{r}}\right)\right] dm \tag{3.13}$$

$$\dot{\overrightarrow{L^0}} = \int_m \left[\dot{\vec{r}} \times \dot{\vec{r}} + \vec{r} \times \ddot{\vec{r}}\right] dm \tag{3.14}$$

El término $\dot{\vec{r}} \times \dot{\vec{r}}$ es igual a 0 ya que es el producto cruz del mismo vector.

$$\dot{\overrightarrow{L^0}} = \int_m \vec{r} \times \ddot{\vec{r}}\, dm \tag{3.15}$$

Derivando $\dot{\vec{r}}$ de la ecuación (3.3).

$$\ddot{\vec{r}} = \ddot{\overrightarrow{r_p}} + \left(\dot{\vec{\omega}} \times \overrightarrow{r_m}\right) + \left(\vec{\omega} \times \dot{\overrightarrow{r_m}}\right) \tag{3.16}$$

Sustituyendo la ecuación (3.2) y (3.16) en (3.15).

$$\dot{\overrightarrow{L^0}} = \int_m \left[\left(\overrightarrow{r_p} + \overrightarrow{r_m}\right) \times \left(\ddot{\overrightarrow{r_p}} + \left(\dot{\vec{\omega}} \times \overrightarrow{r_m}\right) + \left(\vec{\omega} \times \dot{\overrightarrow{r_m}}\right)\right)\right] dm \tag{3.17}$$

$$\dot{\overrightarrow{L^0}} = \int_m \left[\overrightarrow{r_p} \times \left(\ddot{\overrightarrow{r_p}} + \left(\dot{\vec{\omega}} \times \overrightarrow{r_m}\right) + \left(\vec{\omega} \times \dot{\overrightarrow{r_m}}\right)\right) + \overrightarrow{r_m} \times \left(\ddot{\overrightarrow{r_p}} + \left(\dot{\vec{\omega}} \times \overrightarrow{r_m}\right) + \left(\vec{\omega} \times \dot{\overrightarrow{r_m}}\right)\right)\right] dm \quad (3.18)$$

Desarrollando términos, así como la integral, queda.

$$\dot{\overrightarrow{L^0}} = \left[\overrightarrow{r_p} \times \ddot{\overrightarrow{r_p}} + \overrightarrow{r_p} \times \left(\dot{\vec{\omega}} \times \overrightarrow{r_c}\right) + \overrightarrow{r_p} \times \left(\vec{\omega} \times \dot{\overrightarrow{r_c}}\right) + \overrightarrow{r_c} \times \ddot{\overrightarrow{r_p}}\right]M + \left[\overrightarrow{r_c} \times \left(\dot{\vec{\omega}} \times \overrightarrow{r_c}\right) + \overrightarrow{r_c} \times \left(\vec{\omega} \times \dot{\overrightarrow{r_c}}\right)\right]M \quad (3.19)$$

$$\dot{\overrightarrow{L^0}} = \left[\overrightarrow{r_p} \times \ddot{\overrightarrow{r_p}} + \overrightarrow{r_p} \times \left(\dot{\vec{\omega}} \times \overrightarrow{r_c}\right) + \overrightarrow{r_p} \times \left(\vec{\omega} \times \dot{\overrightarrow{r_c}}\right) + \overrightarrow{r_c} \times \ddot{\overrightarrow{r_p}}\right]M + \left[\overrightarrow{r_c} \times \left(\dot{\vec{\omega}} \times \overrightarrow{r_c}\right) + \overrightarrow{r_c} \times \left(\vec{\omega} \times \left(\vec{\omega} \times \overrightarrow{r_c}\right)\right)\right]M \quad (3.20)$$

$$\dot{\overrightarrow{L^0}} = \left[\overrightarrow{r_p} \times \ddot{\overrightarrow{r_p}} + \overrightarrow{r_p} \times \left(\dot{\vec{\omega}} \times \overrightarrow{r_c}\right) + \overrightarrow{r_p} \times \left(\vec{\omega} \times \dot{\overrightarrow{r_c}}\right) + \overrightarrow{r_c} \times \ddot{\overrightarrow{r_p}}\right]M + \left[\overrightarrow{r_c} \times \left(\dot{\vec{\omega}} \times \overrightarrow{r_c}\right) + \vec{\omega} \times \left(\overrightarrow{r_c} \times \left(\vec{\omega} \times \overrightarrow{r_c}\right)\right)\right]M \quad (3.21)$$

$$\dot{\overrightarrow{L^0}} = \left[\overrightarrow{r_p} \times \ddot{\overrightarrow{r_p}} + \overrightarrow{r_p} \times \left(\dot{\vec{\omega}} \times \overrightarrow{r_c}\right) + \overrightarrow{r_p} \times \left(\vec{\omega} \times \dot{\overrightarrow{r_c}}\right)\right]M + \overrightarrow{r_c} \times \ddot{\overrightarrow{r_p}}M + J \cdot \dot{\vec{\omega}} + \vec{\omega} \times J \cdot \vec{\omega} \quad (3.22)$$

Momento de fuerza con respecto a un punto P en el cuadrorotor.

$$M^P = \overrightarrow{r_c} \times \ddot{\overrightarrow{r_p}}M + J \cdot \dot{\vec{\omega}} + \vec{\omega} \times J \cdot \vec{\omega} \quad (3.23)$$

Fuerza resultante en el cuadrorotor.

$$M\ddot{r_p} = F \quad (3.24)$$

Sustituyendo (3.23) y (3.24) en (3.11) obtenemos la forma final del Momento de fuerza con respecto al sistema inercial.

$$M^0 = M^P + \overrightarrow{r_p} \times F \quad (3.25)$$

En el caso particular de que el polo sea el mismo que el origen del sistema inercial. El único cambio que se percibe es respecto al polo.

$$M^P = J^P \cdot \dot{\omega} + \omega \times J^P \cdot \omega \tag{3.26}$$

Usando los ejes principales de inercia como direcciones para los vectores base, esta ecuación matricial equivale a:

$$\begin{aligned} J_x\dot{\omega} - (J_y - J_z)\omega_y\omega_z &= M_{x,} \\ J_y\dot{\omega} - (J_z - J_x)\omega_z\omega_x &= M_y, \\ J_z\dot{\omega} - (J_x - J_y)\omega_x\omega_y &= M_z \end{aligned} \tag{3.27}$$

Donde M^0J para el estudio del cuadrorotor es de la siguiente forma.

$$J = \begin{bmatrix} I_{11} & 0 & 0 \\ 0 & I_{22} & 0 \\ 0 & 0 & I_{33} \end{bmatrix} \tag{3.28}$$

3.3 Ángulos de Euler

La orientación de un cuerpo de base $\bar{e}^2$ es el resultado de tres rotaciones sucesivas. La primera rotación es en el eje e_z^1 y genera un ángulo ψ. Se lleva una orientación intermedia denotada como $\bar{e}^{2''}$ Figura 3-2. La segunda rotación para el ángulo θ es en el eje $e_x^{2''}$, resultando otra orientación intermedia denotada $\bar{e}^{2'}$. La tercera rotación para el ángulo ϕ es a través del eje $e_z^{2'}$ produce la orientación final de la base que se denota $\bar{e}^2$. La secuencia de rotación que se utilizó para los ángulos Euler fue: e_z, e_x y e_z. Al final se busca obtener la matriz $\bar{\boldsymbol{A}}$ en términos de los ángulos ψ, θ, ϕ, dicha matriz se encuentra tomando en cuenta las rotaciones individuales.

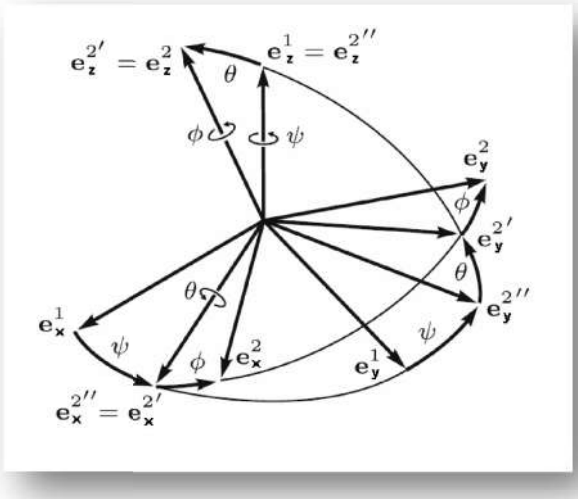

Figura 3.2 Ángulos de Euler

$$\bar{e}^2 = \bar{\boldsymbol{A}}_\phi \bar{e}^{2'} \quad , \qquad \bar{e}^{2'} = \bar{\boldsymbol{A}}_\theta \bar{e}^{2''} \quad , \qquad \bar{e}^{2''} = \bar{\boldsymbol{A}}_\psi \bar{e}^1 \tag{3.29}$$

$$\bar{\boldsymbol{A}}_\phi = \begin{bmatrix} cos\phi & sen\phi & 0 \\ -sen\phi & cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.30}$$

$$\bar{\boldsymbol{A}}_\theta = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\theta & sen\theta \\ 0 & -sen\phi & cos\theta \end{bmatrix} \tag{3.31}$$

$$\overline{\boldsymbol{A}}_{\psi} = \begin{bmatrix} cos\psi & sen\psi & 0 \\ -sen\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.32}$$

$$\overline{\boldsymbol{A}} = \overline{\boldsymbol{A}}_{\phi}\overline{\boldsymbol{A}}_{\theta}\overline{\boldsymbol{A}}_{\psi} \tag{3.33}$$

Se abrevia $s\psi$, $s\theta$, $s\phi$ para $sen\psi$, $sen\theta$, $sen\phi$ y $c\psi$, $c\theta$, $c\phi$ para $cos\psi$, $cos\theta$, $cos\phi$ respectivamente. Multiplicando las matrices de cosenos directores individuales se obtiene la siguiente expresión.

$$\begin{bmatrix} e_x^2 & e_y^2 & e_z^2 \end{bmatrix} = \begin{bmatrix} c\psi c\phi - s\psi c\theta s\phi & s\psi c\phi + c\psi c\theta s\phi & s\theta s\phi \\ -c\psi s\phi - s\psi c\theta c\phi & -s\psi s\phi + c\psi c\theta c\phi & s\theta c\phi \\ s\psi s\theta & -c\psi s\theta & c\theta \end{bmatrix} \begin{bmatrix} e_x^1 & e_y^1 & e_z^1 \end{bmatrix} \tag{3.34}$$

En forma reducida se expresa:

$$\bar{e}^2 = \overline{\boldsymbol{A}}\bar{e}^1 \tag{3.35}$$

3.4 Fuerza de sustentación y momentos en el cuadrorotor

Se usa un sistema de coordenadas fijo al cuerpo con origen en el centro de masa del cuadrorotor. Los ejes x_B y y_B son paralelos a los brazos del cuadrorotor y son ortogonales entre sí, y el eje z_B es perpendicular al plano formado por x_B y y_B. Los ejes fijos a la tierra están definidos por x_W, y_W y z_W. La fuerza de sustentación se representa por los vectores $F_1, F_2, F_3\ y\ F_4$. Cada motor produce también un momento perpendicular $M_1, M_2, M_3\ y\ M_4$ al plano de rotación de la hélice. Los momentos producidos M_1 y M_3 actúan en la dirección z mientras que M_2 y M_4 rotan en la dirección $-z$. La matriz de momento de inercia tiene referencia al centro de masa del cuadrorotor (D. Mellinger et al, 2010).

La figura 3-3 describe el modelo del cuadrorotor que se usa para su análisis.

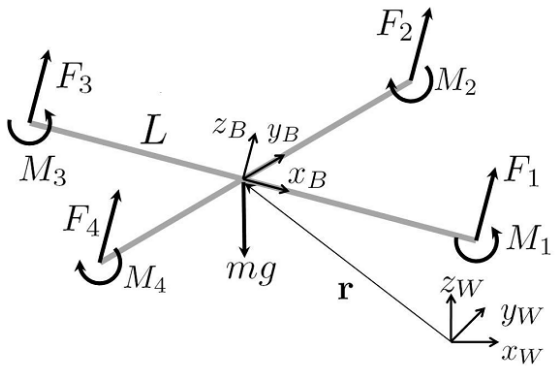

Figura 3.3 Sistema de coordenadas, fuerzas y momentos en el cuadrorotor (D. Mellinger et al, 2010)

La ecuación que gobierna la aceleración del centro de masa considerando la gravedad.

$$m\ddot{r} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + A \begin{bmatrix} 0 \\ 0 \\ \sum \bar{F}_i \end{bmatrix} \tag{3.36}$$

Las componentes de velocidad angular fijas al cuadrorotor son p, q, y r. Estos valores están relacionados a las derivadas de los ángulos de rotación, cabeceo y guiñada de acuerdo a:

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} c\theta & 0 & -c\phi s\theta \\ 0 & 1 & s\phi \\ s\theta & 0 & c\phi c\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{3.37}$$

La aceleración angular determinada por la ecuación de Euler es:

$$J \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} a(\bar{F}_2 - \bar{F}_4) \\ a(\bar{F}_3 - \bar{F}_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times J \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{3.38}$$

3.5 Modelo del motor

Cada motor tiene una velocidad angular ω_i y produce una fuerza F_i de acuerdo a:

$$\bar{F}_i = k_F \omega_i^2 \tag{3.39}$$

Donde:

$\bar{F}_i$ = Fuerza de sustentación $[N]$.

ω = Velocidad angular $[rpm]$.

k_F = Constante $\left[\frac{N}{rpm}\right]$.

Los motores también producen un momento de acuerdo a:

$$M_i = k_M \omega_i^2 \tag{3.40}$$

Donde:

M_i = Momento de fuerza $[Nm]$.

k_M = Constante $\left[\frac{Nm}{rpm}\right]$.

Considerando que los ángulos de rotación y cabeceo son pequeños entonces $c\phi \approx 1, c\theta \approx 1, s\phi \approx \phi$, y $s\theta \approx \theta$. Si tomamos en cuenta que la suma de las fuerzas de sustentación en los 4 motores para un vuelo en suspensión debe ser proporcional a la fuerza de gravedad que actúa en el cuadrorotor y de forma ideal las constantes k_F son iguales, debe satisfacer.

$$mg = \sum_{i=1}^{4} k_F \omega_h^2 \tag{3.41}$$

y las velocidades de los motores están dadas por

$$\omega_h = \sqrt{\frac{mg}{4k_F}} \tag{3.42}$$

De acuerdo a (D. Mellinger et al, 2010), se presenta un controlador de orientación, diseñado para el vuelo en suspensión. Las velocidades deseadas

de los motores pueden ser escritas como una combinación lineal de cuatro términos.

$$\begin{bmatrix} \omega_1^{des} \\ \omega_2^{des} \\ \omega_3^{des} \\ \omega_4^{des} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 & 1 \\ 1 & 1 & 0 & -1 \\ 1 & 0 & 1 & 1 \\ 1 & -1 & 0 & -1 \end{bmatrix} \begin{bmatrix} \omega_h + \Delta\omega_F \\ \Delta\omega_\phi \\ \Delta\omega_\theta \\ \Delta\omega_\psi \end{bmatrix} \tag{3.43}$$

Donde la velocidad del motor requerida para estar en vuelo suspendido es ω_h, y las desviaciones de este vector nominal son: $\Delta\omega_F$, $\Delta\omega_\phi$, $\Delta\omega_\theta$ y $\Delta\omega_\psi$. El incremento $\Delta\omega_F$ resulta de una fuerza total a lo largo del eje z_B, mientras $\Delta\omega_\phi$, $\Delta\omega_\theta$ y $\Delta\omega_\psi$ producen momentos causando rotación, cabeceo y guiñada respectivamente.

Usando un control proporcional derivativo (PD) con las velocidades diferenciales como entradas de control.

$$\Delta\omega_\phi = k_{p,\phi}(\phi^{des} - \phi) + k_{d,\phi}(p^{des} - p) \tag{3.44}$$

$$\Delta\omega_\theta = k_{p,\theta}(\theta^{des} - \theta) + k_{d,\theta}(q^{des} - q) \tag{3.45}$$

$$\Delta\omega_\psi = k_{p,\psi}(\phi^{des} - \psi) + k_{d,\psi}(r^{des} - r) \tag{3.46}$$

El diagrama a bloques de la figura 3-4 muestra como se relaciona la dinámica del motor, del cuerpo rígido y el control de altitud.

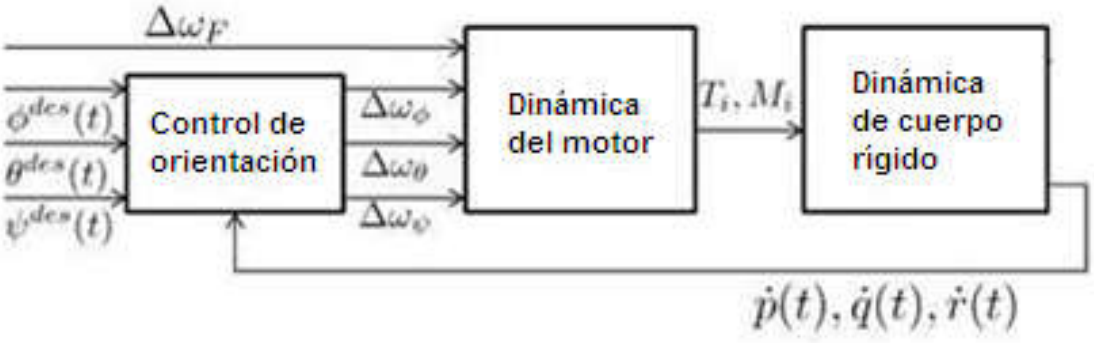

Figura 3.4 Control de orientación (D. Mellinger et al, 2010)

4 CONTROLADOR PID

El controlador PID de acuerdo a (E. García et al, 2010) está definido mediante la siguiente ecuación.

$$u(t) = K_P e(t) + \frac{K_p}{T_i}\int_0^t e(t)dt + K_p T_d \frac{de(t)}{dt} \quad (4.1)$$

Donde

$u(t)$: Entrada de control del proceso.

$e(t)$: Error de la señal.

K_p: Ganancia proporcional.

T_i: Constante de tiempo integral.

T_d: Constante de tiempo derivativa.

En el dominio S el controlador PID se puede escribir como:

$$U(S) = K_p\left[1 + \frac{1}{T_i S} + T_d S\right]E(S) \quad (4.2)$$

Para ajustar los parámetros K_p, T_i y T_d se utiliza el método de Ziegler/Nichols. En lazo abierto este proceso se puede definir según la siguiente función de transferencia.

$$G(S) = \frac{K_0 e^{-S\tau_0}}{(1+S\gamma_0)} \quad (4.3)$$

Donde los parámetros K_p, T_i y T_d se obtienen de la respuesta del sistema en lazo abierto de acuerdo a las ecuaciones (4.7), (4.8), (4.9) y la Figura 4-1.

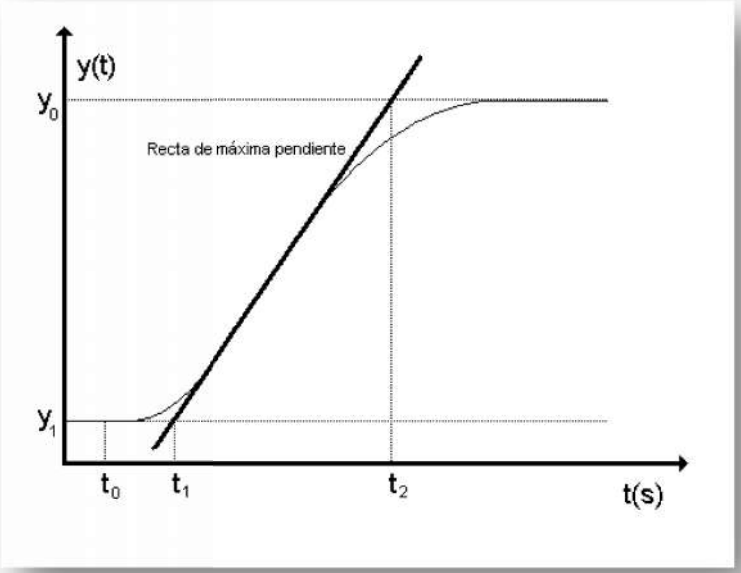

Figura 4.1 Respuesta de salida en lazo abierto.

Donde:

$$\tau_0 = t_1 - t_2 \tag{4.4}$$

$$\gamma_0 = t_2 - t_1 \tag{4.5}$$

$$K_0 = \frac{y_1 - y_0}{u_1 - u_0} \tag{4.6}$$

Según Ziegler/Nichols, las relaciones de estos coeficientes con los parámetros del controlador PID son:

$$K_p = \frac{1.2\gamma_0}{K_0\tau_0} \tag{4.7}$$

$$T_i = 2\tau_0 \tag{4.8}$$

$$T_d = 0.5\tau_0 \tag{4.9}$$

Un controlador PID discreto viene dado por la transformada Z para hacer la implementación en la tarjeta de desarrollo Tiva Launch Pad:

$$U(z) = E(z)K_p\left[1 + \frac{T}{T_i(1-z^{-1})} + T_d\frac{(1-z^{-1})}{T}\right] \tag{4.10}$$

Agrupando términos:

$$\frac{U(z)}{E(z)} = a + \frac{b}{(1-z^{-1})} + c(1 - z^{-1}) \tag{4.11}$$

Donde:

$$a = K_p \tag{4.12}$$

$$b = \frac{K_p T}{T_i} \tag{4.13}$$

$$c = \frac{K_p T_d}{T} \tag{4.14}$$

Se utiliza un diseño de controlador PID en paralelo como se muestra en la figura 4-2.

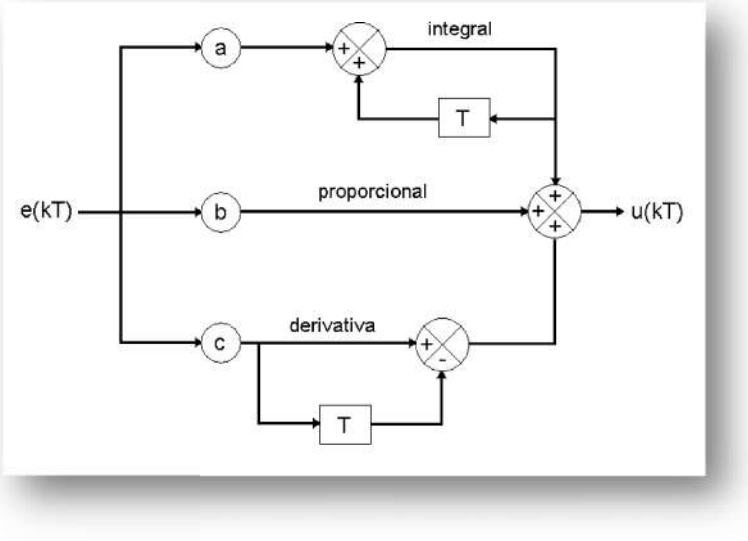

Figura 4.2 Diseño paralelo de un controlador PID.

El muestreo debe ser menor que el tiempo de establecimiento del sistema en lazo abierto. El modelo Ziegler/Nichols toma un valor $T < \tau_0/4$.

Existe un problema asociado a este diseño llamado “integral windup”, el cual provoca largos períodos de sobre impulsos, a causa de los valores excesivos que alcanza la señal de control debido a la acumulación en el integrador. Esto se evita limitando la señal de control entre un valor mínimo y un máximo, es decir estableciendo un intervalo.

Un método empírico se utiliza para afinar el controlador PID (J. Valvano et al, 2012). Dicho método inicia con la constante proporcional K_p. Con ésta se ajusta la rapidez o suavidad de la respuesta del controlador a los cambios en la carga. El siguiente paso es ajustar la constante integral K_i un poco cada vez para mejorar la estabilidad del controlador sin afectar el tiempo de respuesta. No se debe cambiar ambas constantes K_p y K_i a la vez. Si el tiempo de respuesta, sobre impulso y precisión están dentro de límites aceptables entonces el PI es aceptable. Si la precisión y respuesta están bien pero el sobre impulso es inaceptable, se ajusta la constante K_d para reducirlo.

El método empírico es la alternativa que se utilizó ya que el fabricante de los motores sin escobillas BL (Brushless) no proporciona un manual con las características nominales que sirven para la obtención de la gráfica de la función de transferencia con la que se aplica el método de Ziegler/Nichols para sintonizar de forma matemática las constantes del PID.

5 DESCRIPCIÓN DEL SISTEMA

Para los propósitos de este trabajo se escoge un cuadrorotor que este dentro de la categoría MAV. Los requerimientos para entrar en esta categoría son: un peso menor a 2 Kg, hélices de longitud menor a 150mm de radio y rango de vuelo menor de 30Km.

Para este sistema se usó el SK-450 conocido así por su diseño. La estructura del cuadrorotor consiste en una plataforma central en donde se coloca la electrónica para el control y la batería. La estructura central está unida a cuatro largueros equidistantes formando una cruz. Al extremo de cada larguero en la parte superior se encuentran los motores y en la parte inferior el tren de aterrizaje. Los largueros son de nylon y la plataforma central que los une está hecha de fibra de vidrio. El tamaño es de 450 mm de ancho, 80 mm de altura; lleva cuatro motores MT2213-935KV sin escobillas (BL) de 55 g de peso cada uno; los motores con los que cuenta son modelos recientes específicamente para usarlos en multirotores y se les puede adaptar hélices de 10 x 4.5 y 8 x 4.5 in; cuatro hélices de plástico de 10 x 4.5 in que equivale a 127 mm que es menor a 150 mm de radio, dos para giro en sentido horario y dos para sentido anti horario; cuatro controladores electrónicos de velocidad (ESC) de 20 A de la marca Turnigy; una batería de polímero de Litio de 2200 mAh, con un arreglo interno tres serie uno paralelo (3s1p) de 11.1v.

Se escoge el modelo SK450 porque las piezas no están soldadas, es armable puede cambiarse la plataforma central y formar una configuración asimétrica y se pueden remplazar el resto de los componentes ya que todos los elementos se instalan de forma externa y no hay ningún inconveniente físico que impida montar un piloto con dimensiones similares a la estructura central (100 mm x 100 mm). Es un modelo resistente a impactos, ligeramente flexible y pesa 300 g. El peso total del modelo incluyendo la electrónica que lleva (TivaC, sensores de navegación y etapa de acondicionamiento), motores, hélices, ESC y batería, es de 1.1 Kg. Por lo que el cuadrorotor SK450 entra en la categoría MAV.

Para el controlador de vuelo se utiliza la tarjeta de desarrollo TivaC Launch Pad de Texas Instruments que tiene los módulos necesarios para la implementación de un piloto automático. Utiliza un microcontrolador TM4C123GH6PMI de última generación de 32 bits ARM CortexM4 con manejo de punto flotante con una velocidad de 80 MHz. Cuenta con 8 módulos PWM (Pulse Width Modulation), que son suficientes ya que el cuadrorotor solo necesita 4, uno por cada motor. Cuenta con 10 módulos I^2C para poder comunicarse con sensores para navegación que utilicen este protocolo. Para una comunicación con la computadora se dispone de 8 módulos UART para protocolo serial y un módulo USB. Además de 2 módulos CAN, 4 SPI, 3 comparadores analógicos, 2 perro guardián, 8 temporizadores de 32bits cada uno, 12 convertidores analógico digital (ADC). Una memoria de programa de 256 KB, memoria de datos de 32 KB SRAM y memoria de datos de 2 KB EEPROM.

Como sensores de navegación para el piloto de vuelo se opta por utilizar la expansión que ofrece para la tarjeta de desarrollo Tiva C llamada Sensor Hub Booster Pack, ya que cuenta con tres giróscopos uno para cada eje (x, y, z), tres acelerómetros uno por cada eje y además cuenta con un barómetro que se puede utilizar para calcular la altitud del cuadrorotor. La ventaja de utilizar esta expansión es que no se necesita hacer ninguna modificación a la tarjeta TivaC, no hay necesidad de hacer un diseño electrónico adicional ya que se conecta y se alimenta directamente de la Tiva. Otra ventaja es que Texas Instruments proporciona librerías con funciones en lenguaje C para manejar los integrados del Sensor Hub.

En la figura 5-1, se puede observar de manera independiente los componentes que constituyen el cuadrorotor.

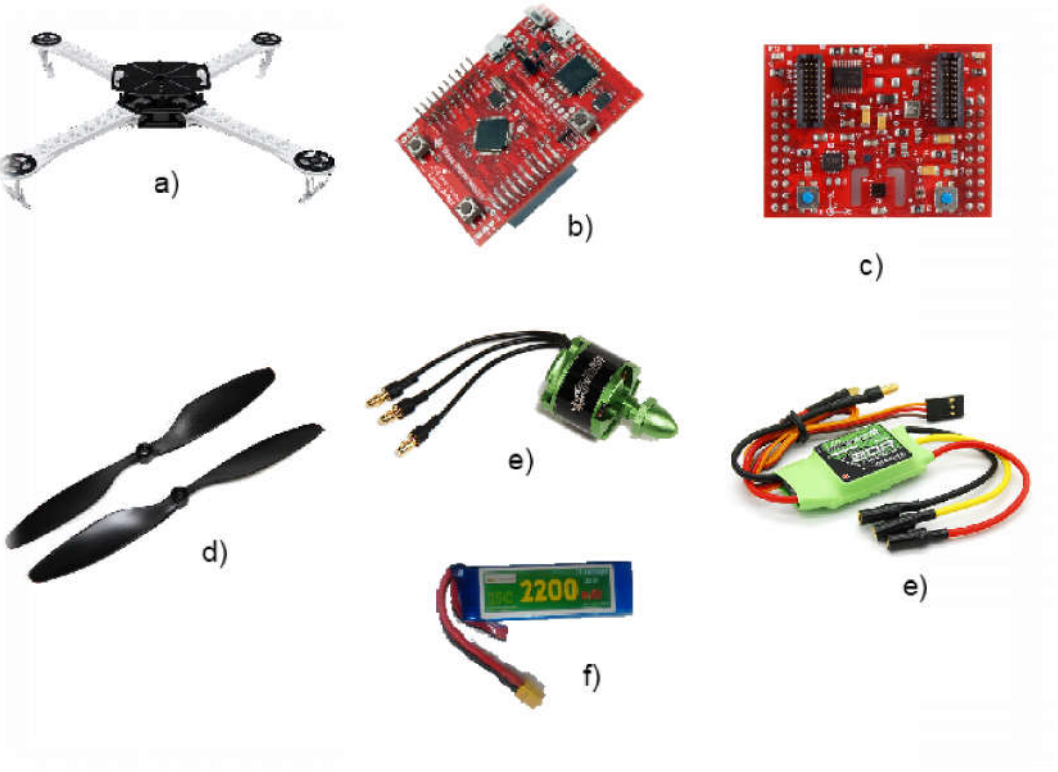

Figura 5.1 Componentes del cuadrorotor SK450

a) Estructura SK-450, b) TivaC TM4C123GXL c) Sensores de orientación y velocidad angular, d) Hélices, e) Motor sin escobillas, f) Controladores Electrónicos de Velocidad, g) Batería.

6 IMPLEMENTACIÓN DE LA PROGRAMACIÓN Y CIRCUITOS

En el esquema a bloques se representa la relación, conexión simplificada y operatividad de los elementos de control, interfaces, sensores y actuadores que conforman al cuadrorotor. La figura 6-1 muestra la implementación.

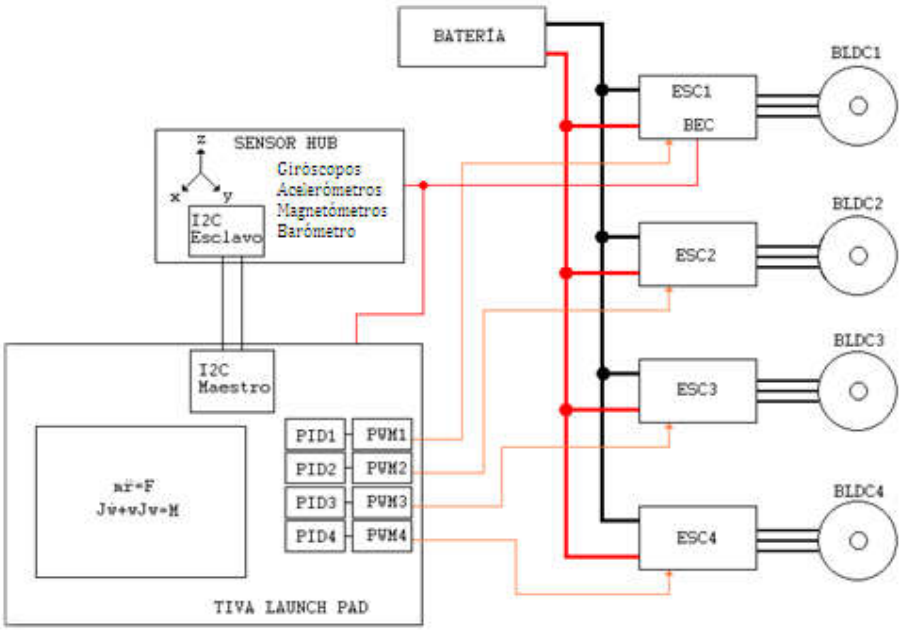

Figura 6.1 Diagrama a bloques de la implementación del cuadrorotor

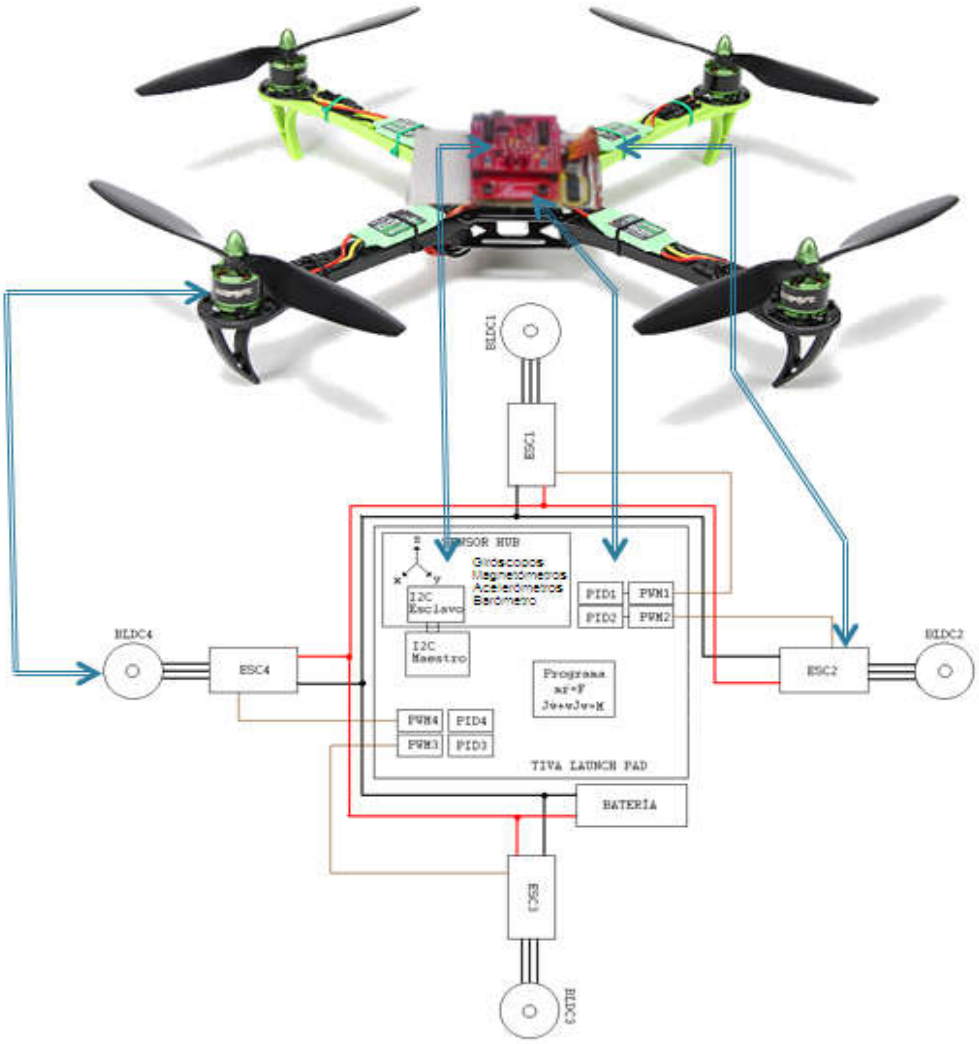

Figura 6.2 Implementación física y su relación con el diagrama a bloques

En el diagrama a bloques figura 6-1 se observa que el TivaC se comunica por medio de un módulo para protocolo I²C, con los sensores de navegación y barómetro que se encuentran embebidos en el Sensor Hub Booster Pack para obtener la velocidad angular [rad/s], aceleración [m/s²], orientación [T] (tesla) y altura [m]. En donde la Tiva C es el maestro y los sensores de navegación son los esclavos. Se encarga del control de los módulos PWM, de las operaciones de la dinámica del cuadrorotor, así como de los controladores PID, estos modifican el ancho de pulso de las señales PWM. Las salidas de los módulos PWM son las señales de control para programar los ESC y permiten regular la velocidad de los motores. Los ESC tienen una etapa de potencia para poder alimentar y controlar la velocidad de los motores BL.

6.1 Controlador Electrónico de velocidad ESC

Como la tarjeta de desarrollo TivaC se alimenta a un voltaje de 3 V, y la etapa de control de los ESC trabaja a 5V, se requiere diseñar un circuito electrónico para acondicionar los voltajes y poder conectar las salidas PWM de la TivaC a las entradas de control de los ESC. Este circuito solo necesita ser unidireccional y amplificar de 3 a 5 V.

Se propone el siguiente circuito con optoacoplador 4N25 y compuerta NOT (Figura 6-3) que permite ensamblar el TivaC con los ESC. La tensión en los extremos del LED en conducción es de 1.2 V y debe circular una corriente de 5 mA.

$$R1 = \frac{3V-1.2V}{5mA} = 360\Omega \tag{6.1}$$

Para un valor comercial de resistencia 330 Ω la corriente I que circula por el LED.

$$I = \frac{3V-1.2V}{330} = 5.45mA \tag{6.2}$$

Del lado derecho del optoacoplador se conecta una resistencia de 10 KΩ debido a la conexión a colector abierto del transistor para que circule una corriente de $5V/10\ K\Omega = 0.5\ \mu A$.

Nótese que el diagrama representa la conexión para un solo motor, el resto de motores llevan el mismo arreglo eléctrico.

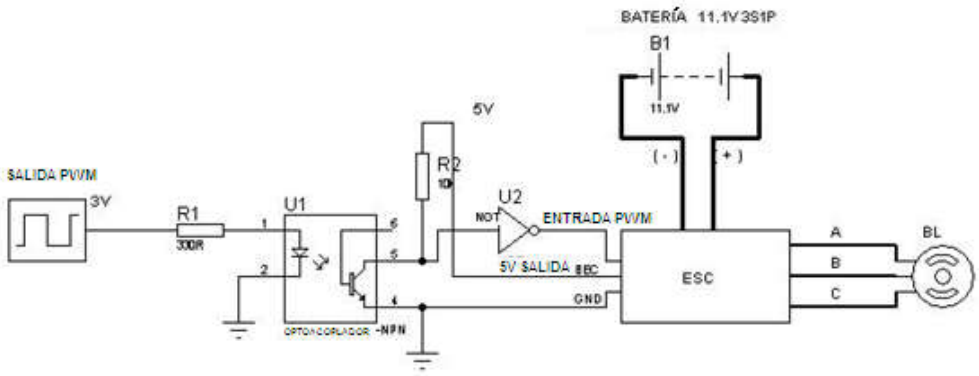

Figura 6.3 Conexión eléctrica piloto, ESC, batería y motor BL

Los motores BL no funcionan al conectarlos directamente a una fuente de poder. Los ESC se encargan de la secuencia de conmutación que por medio de un puente h trifásico, permite el funcionamiento del motor (Microchip AN885).

Para los ESC Multistar de 20 A es necesario controlarlos por medio de una señal PWM. Tiene dos modos de funcionamiento: modo normal y modo de programación.

En modo normal, al regular la señal de PWM, el ESC varía la velocidad del motor.

En modo programación se accesa a un menú donde se cambian algunas características para su funcionamiento como: calibración, tipo de batería y tiempo (para el tipo de motor según el número de polos).

6.1.1 CALIBRACIÓN DEL ESC

Esta dentro del modo programación. Cuando es la primera vez que se utiliza el ESC, es necesario calibrarlo. Este procedimiento sirve para reconocer la señal PWM de entrada, esto es porque los transmisores comerciales en ocasiones manejan diferentes periodos entre 10 a 20 ms, de esta forma puede ser compatible con varios tipos de transmisores.

Para la calibración del ESC se experimentó con la señal PWM de la figura 6-4, donde se muestra una señal con un periodo de 10 ms y un ancho de pulso que varía de 1 a 2 ms, los cuales están definidos como el valor mínimo y máximo respectivamente, que define el intervalo $min \leq PWM \leq max$ del ancho de pulso en la señal PWM (10 a 20% del ciclo de trabajo) que recibe el ESC con la que varia la velocidad ω del motor.

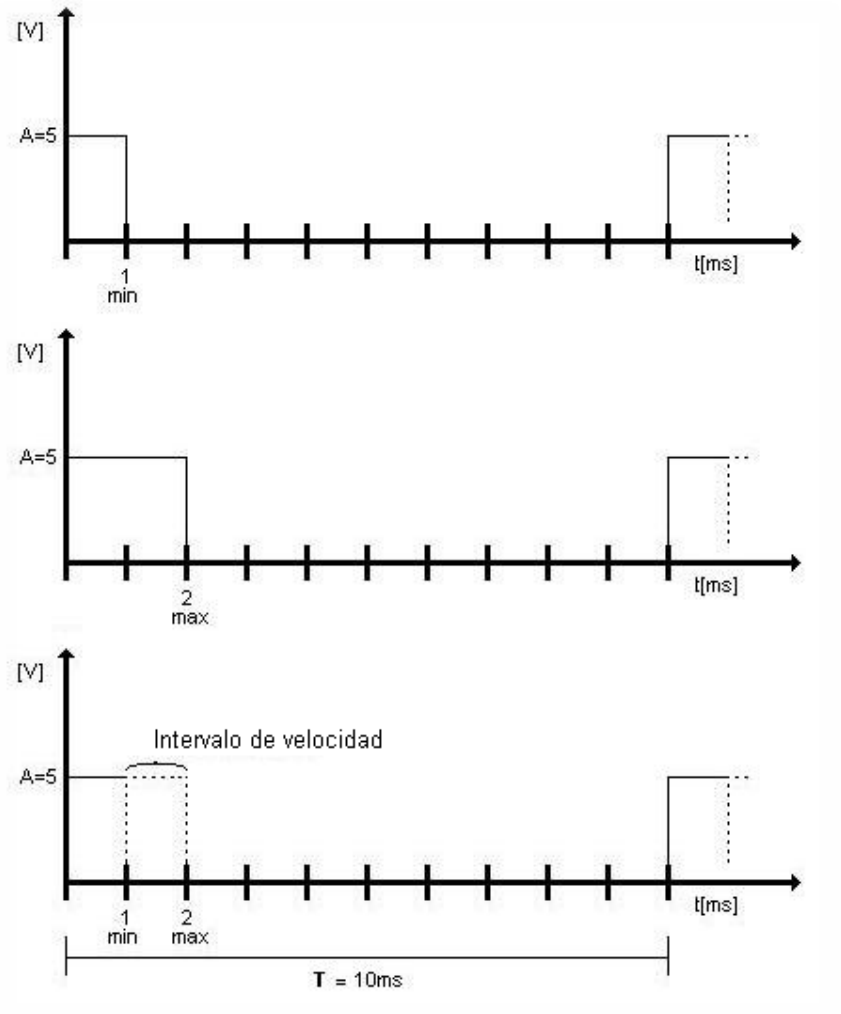

Figura 6.4 Señal PWM para el ESC

Para la calibración de ESC, se debe seguir la siguiente secuencia según el diagrama de flujo de la figura 6-5.

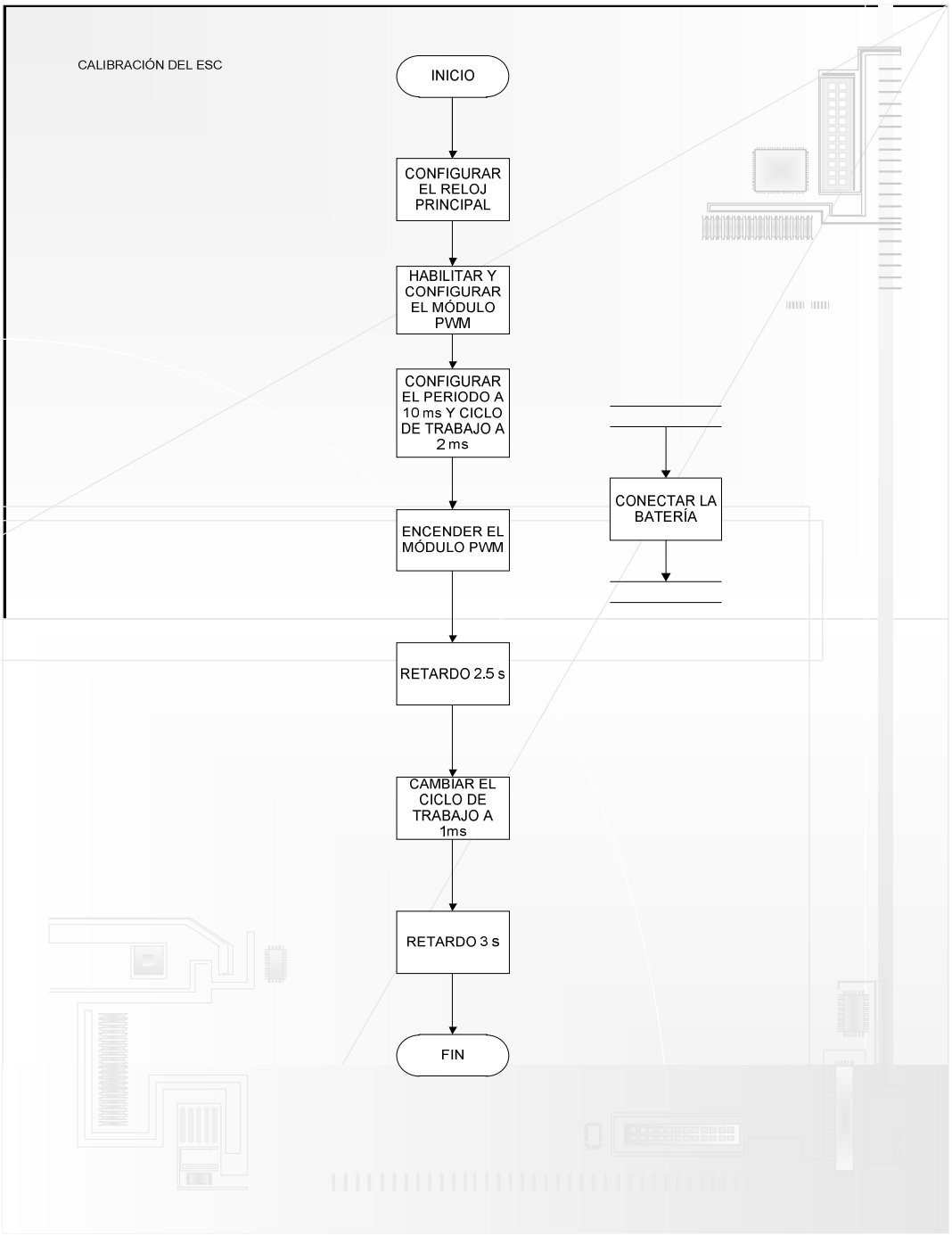

Figura 6.5 Calibración del ESC

6.1.2 *MODO PROGRAMACIÓN DEL ESC*

Una vez que el ESC está calibrado, es posible accesar a algunas características de configuración por medio de secuencias de acuerdo al siguiente diagrama de flujo (figura 6-6).

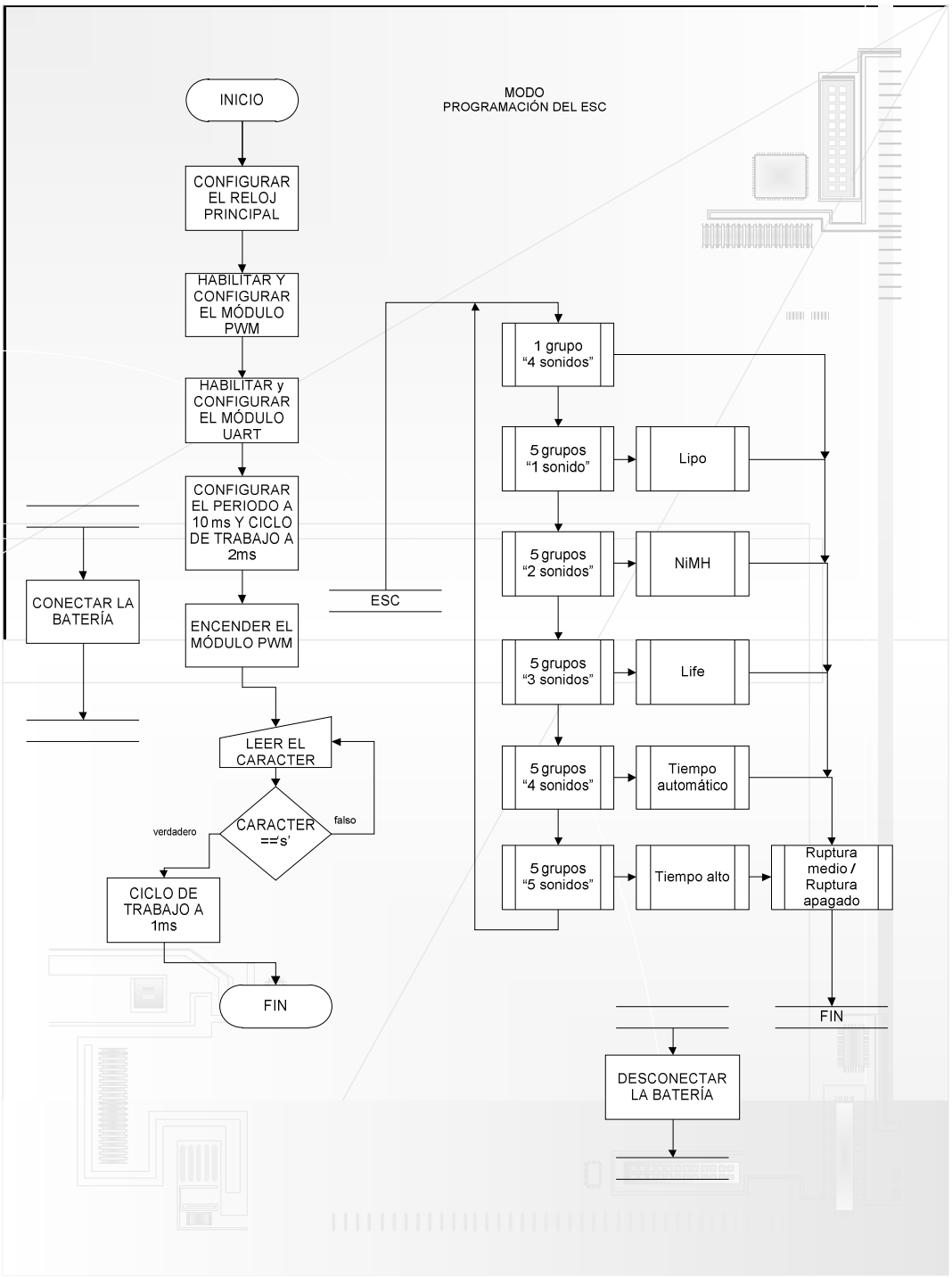

Figura 6.6 Modo programación del ESC

6.1.3 *MODO NORMAL DEL ESC*

Una vez correctamente configurado el siguiente paso para variar la velocidad del motor BL es de acuerdo al siguiente diagrama de flujo (Figura 6-7).

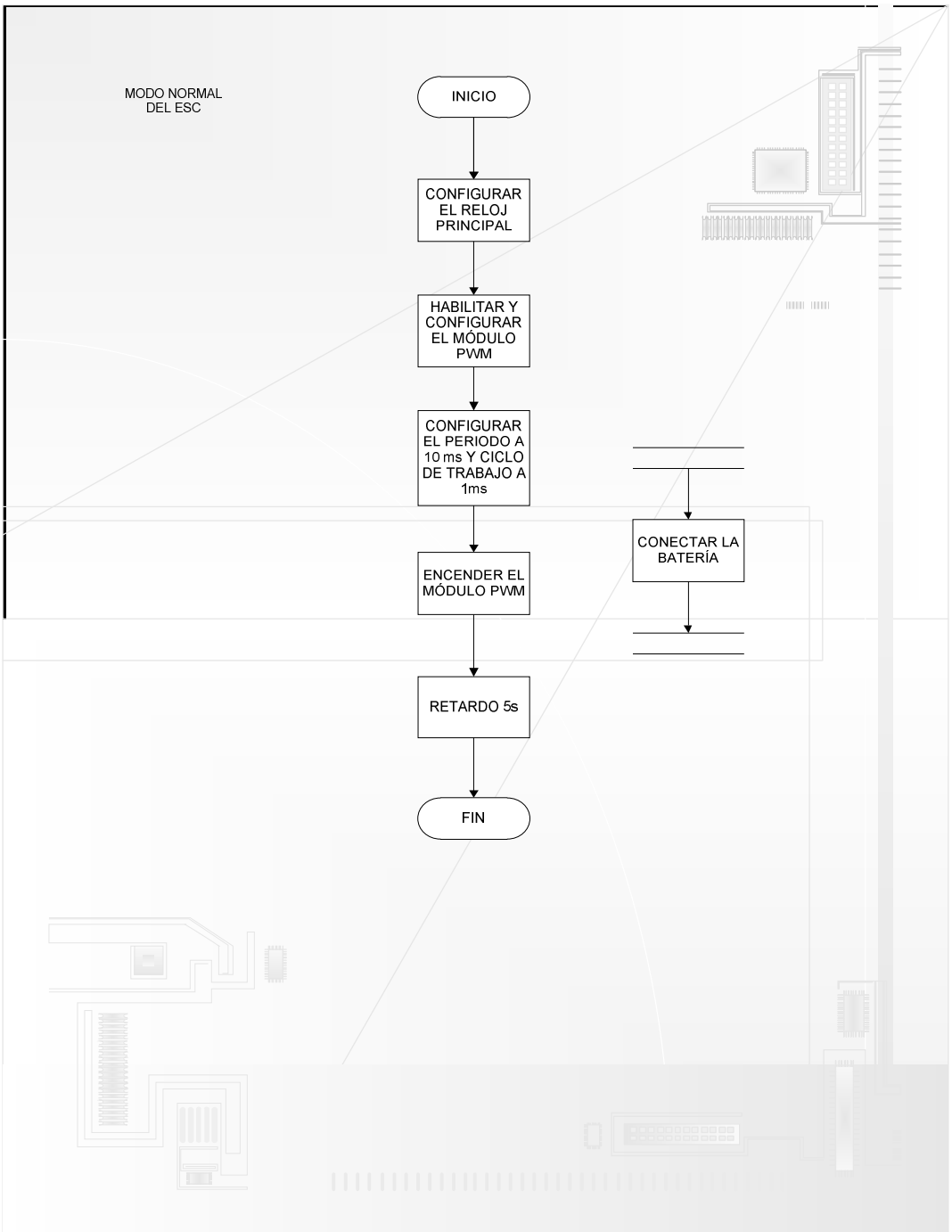

Figura 6.7 Modo normal del ESC

6.2 Sensor Hub Booster Pack

En esta implementación se toma como referencia la tabla 2-1 en donde las tasas de actualización en los sensores de posición varían entre 50 y 100 Hz. Esto quiere decir que si se implementa un controlador sencillo, la estabilidad del cuadrorotor a esa frecuencia no tendrá una rápida respuesta por lo que será difícil estabilizarlo en condiciones ambientales con mucho viento. Una opción es aumentar la tasa de muestreo de los sensores de posición por ejemplo a 1 KHz, pero se necesitan motores que puedan responder a esa velocidad y los motores sin escobillas MT2213-935KV que se usan en el SK-450, no son capaces de responder a esa frecuencia de 1 KHz. La ventaja es que al trabajar a una frecuencia de 50 Hz, la tarjeta TivaC dispone de más tiempo para procesar operaciones mientras llega la siguiente actualización de datos por parte de los sensores de posición. Por esta razón se decide utilizar una frecuencia baja de 50 Hz en la implementación.

El sensor Hub fue diseñado por el fabricante como expansión del Tiva launch pad. En la tabla 10-2 Interfaz de conexión del TivaC Launch Pad, se muestra la disposición de pines que utiliza para su conexión con el TivaC. El Sensor Hub cuenta con los siguientes sensores: TI TMP006 Infrared Temperature, Bosch BMP180 Digital Pressure, Invensense MPU-9150 9-axis Motion, Intersil ISL29023 Ambient & Infrared Light Sensor, Sensirion SHT21 Humidity & Ambient Temperature Sensor de los cuales para este propósito únicamente se necesita el MPU-9150 que contiene los 3 giróscopos, 3 acelerómetros, 3 magnetómetros y el BMP180 que contiene el sensor de presión (barómetro), ambos se comunican al TivaC por medio de protocolo I^2C.

En el desarrollo experimental se tomaron lecturas a través de los sensores MPU9150 y BMP180; estas se pueden visualizar por medio de puerto serial utilizando el módulo UART del Tiva y mostrándolos vía PC por medio de una interfaz o terminal al puerto serial.

El algoritmo para el manejo del Sensor Hub es el de la figura 6-8.

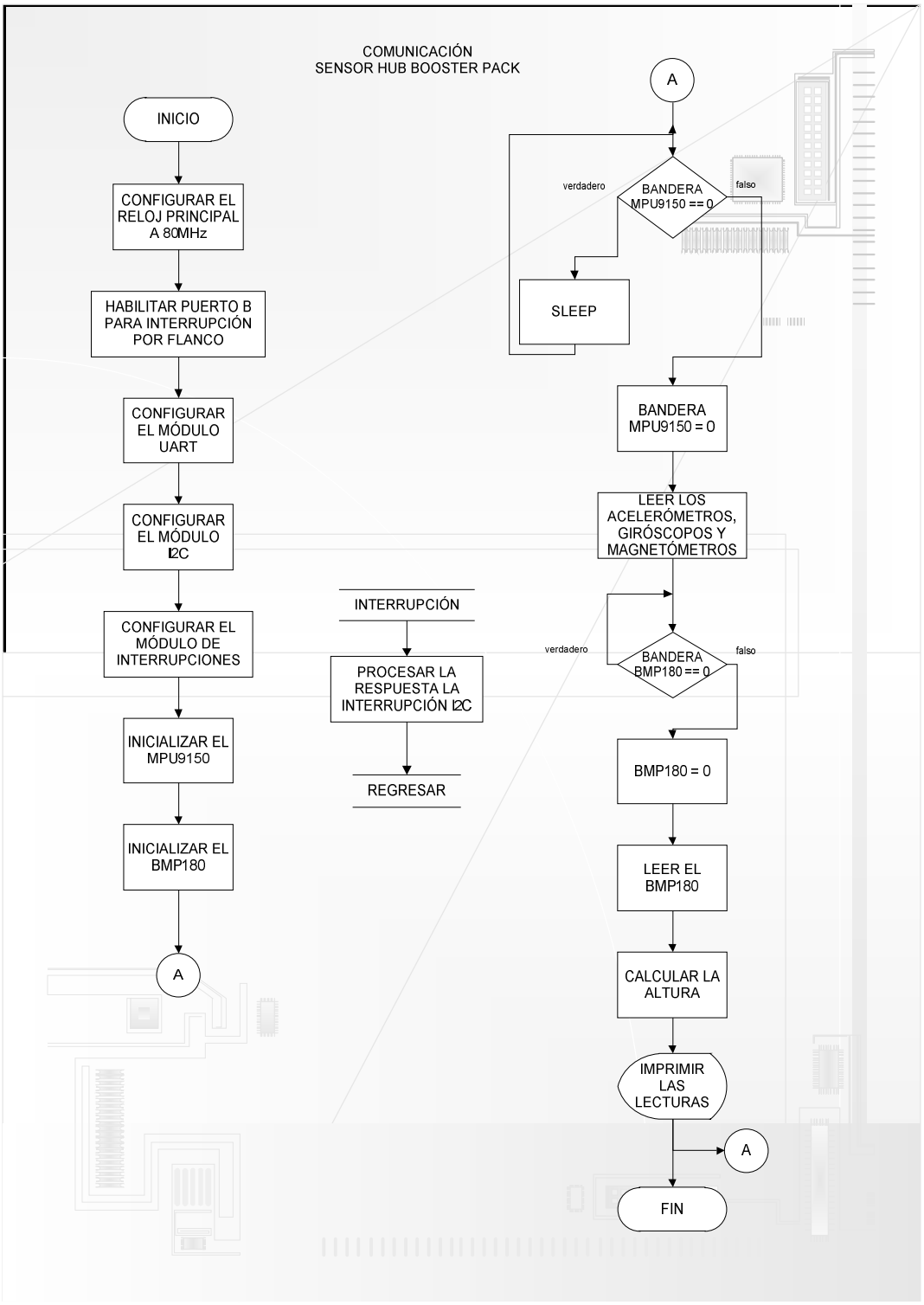

Figura 6.8 Comunicación con Sensor Hub Booster Pack

La adquisición de datos para los ángulos ϕ, θ y ϕ se muestran en las siguientes gráficas (figuras 6-9, 6-10, 6-11). Para el eje de las abscisas representa el tiempo donde cada 50 muestras representa un segundo. Tanto para ϕ como para θ, la variación está entre -0.15 y 0.15 grados. Mientras que para ϕ la variación entre 1 a 8 grados. Esto significa que para la rotación y el cabeceo hay lecturas más precisas que para la guiñada.

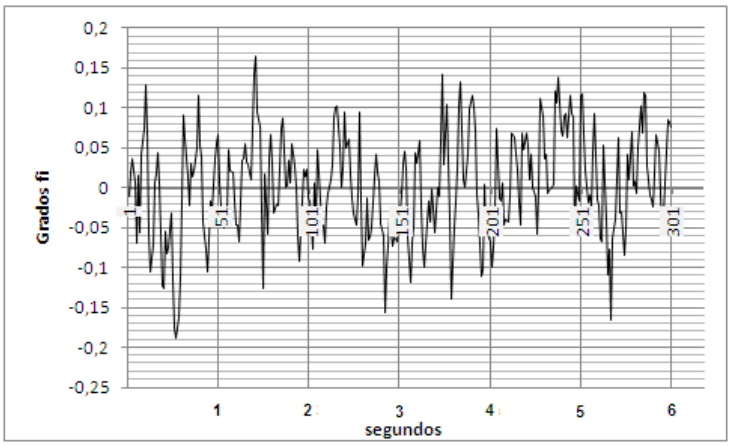

Figura 6.9 Adquisición para ϕ en 0 grados

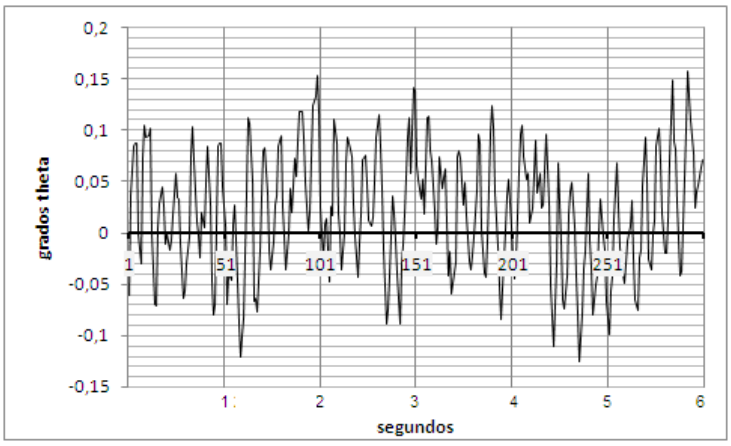

Figura 6.10 Adquisición para θ en 0 grados

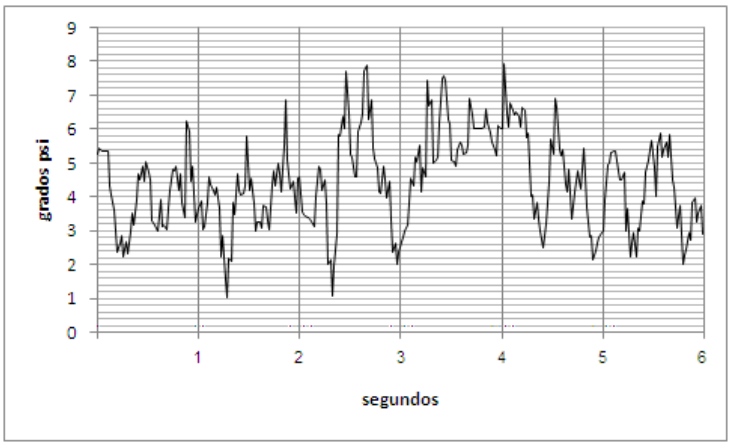

Figura 6.11 Adquisición para ψ en 4 grados

7 PRUEBAS EXPERIMENTALES

Para este experimento se colocó para cada motor BL una hélice de plástico EPP1045 y se le conectó un multímetro línea a línea para medir el voltaje con carga mientras se varía el ancho del pulso en la salida PWM en un motor para regular su velocidad.

Esto quiere decir que para el intervalo $min \leq PWM \leq máx$ le corresponde un valor de voltaje medido. Con este se obtiene la velocidad angular del motor de forma teórica de acuerdo al valor nominal proporcionado por el fabricante. $K_V = 935[RPM/V]$

El ancho de pulso que se varía $(min \leq PWM \leq máx)$ tiene un total de 1250 valores. Donde $0 = min$ y $1250 = máx$.

Al graficar las lecturas de voltaje de cada motor de acuerdo a la tabla 7-1, se muestra un comportamiento parecido al tiempo de carga de voltaje de un capacitor (figura 7-1).

Tabla 7-1 Mediciones experimentales

EPP1045	VOLTAJES			
PWM	**MOTOR$_1$**	**MOTOR$_2$**	**MOTOR$_3$**	**MOTOR$_4$**
0	0	0	0	0
50	0	0	0	0
100	0	0	0	0
150	2,77	1,63	1,7	1,7
200	4,4	3,01	3,08	3,09
250	5,33	3,83	3,82	3,84
300	6,01	4,49	4,48	4,5
350	6,29	4,98	5	5,04
400	6,56	5,43	5,4	5,47
450	6,83	5,74	5,75	5,81
500	7,09	6,11	6,12	6,21
550	7,23	6,38	6,38	6,49
600	7,34	6,65	6,65	6,77
650	7,47	6,95	6,95	7,06
700	7,6	7,24	7,27	7,35
750	7,72	7,49	7,51	7,62
800	7,85	7,75	7,79	7,9
850	7,97	8,01	8,07	8,19
900	8,04	8,22	8,25	8,41
950	8,04	8,22	8,25	8,39
1000	8,04	8,22	8,25	8,39
1050	8,04	8,22	8,25	8,39
1100	8,04	8,22	8,25	8,39
1150	8,04	8,22	8,25	8,39
1200	8,04	8,22	8,25	8,39
1250	8,04	8,22	8,25	8,39

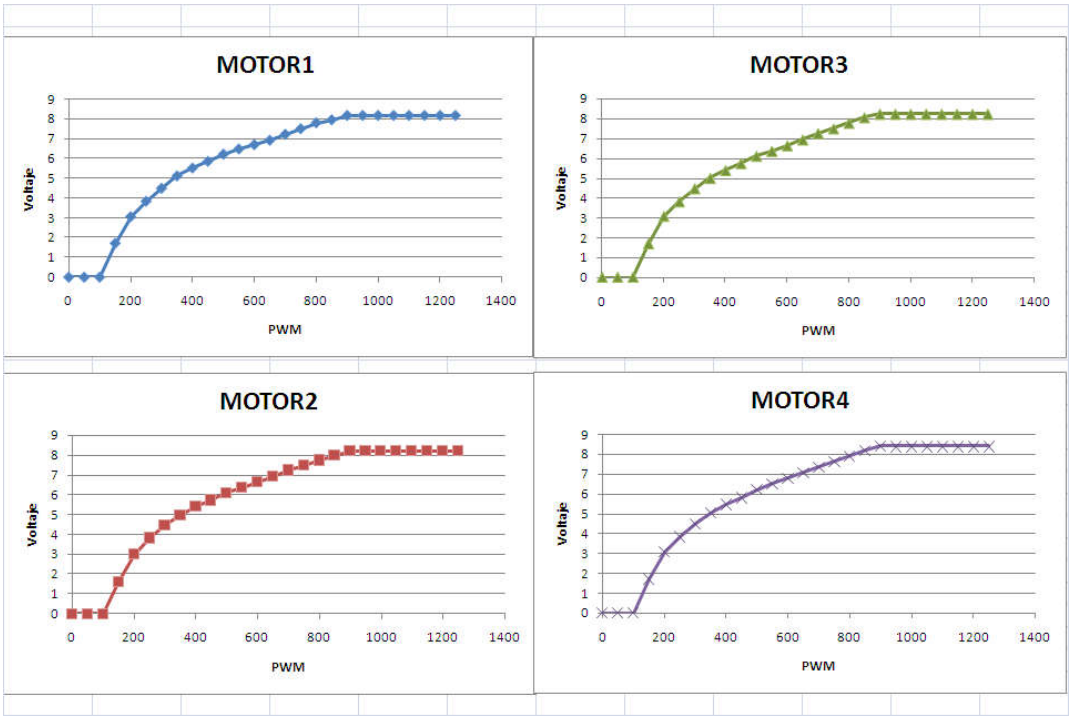

Figura 7.1 PWM - Voltaje

En la figura 7-2 se observa que dentro de un intervalo de valores entre 400 y 900 hay una línea recta, por lo que se propone utilizar el método de mínimos cuadrados para obtener el resto de valores de voltaje dentro de ese intervalo.

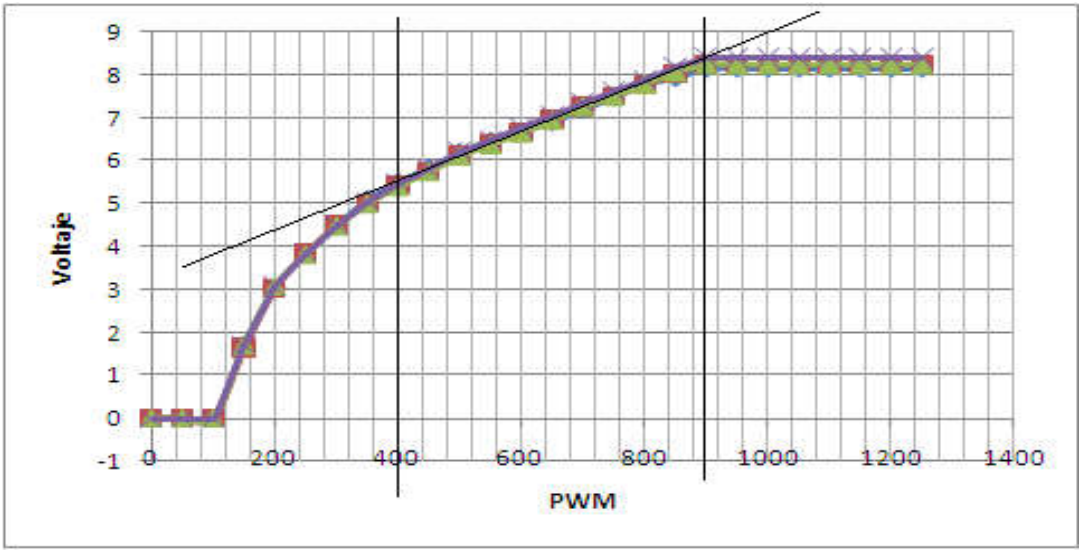

Figura 7.2 PWM-Voltaje, intervalo lineal

Según las especificaciones del motor, la constante de velocidad es de $K_V = 935[RPM/V]$ con lo que se puede encontrar la velocidad angular en RPM

$$RPM = K_V \times Volts \tag{7.1}$$

El método de mínimos cuadrados para encontrar la pendiente y ecuación de la recta se realizo de acuerdo a la fórmula (7.2) y (7.3) respectivamente.

$$m = \frac{\sum xy - \frac{\sum x \sum y}{n}}{\sum x^2 - \frac{(\sum x)^2}{n}} \tag{7.2}$$

$$y = mx + b \tag{7.3}$$

Donde:

y es la ecuación de la recta.

m es la pendiente.

Por lo que la ecuación se puede expresar de la siguiente forma:

$$Voltaje = mPWM + b \tag{7.4}$$

Con mediciones experimentales de la tabla 7-3 promediando los 4 motores se obtiene la pendiente m y b con lo que se puede encontrar el voltaje para cualquier valor de PWM de acuerdo a la formula (7.4).

m =0.005488485; b =3.349273

450	5.81909091
500	6.09351515
550	6.36793939
600	6.64236364
650	6.91678788
700	7.19121212
750	7.46563636
800	7.74006061
850	8.01448485
900	8.28890909

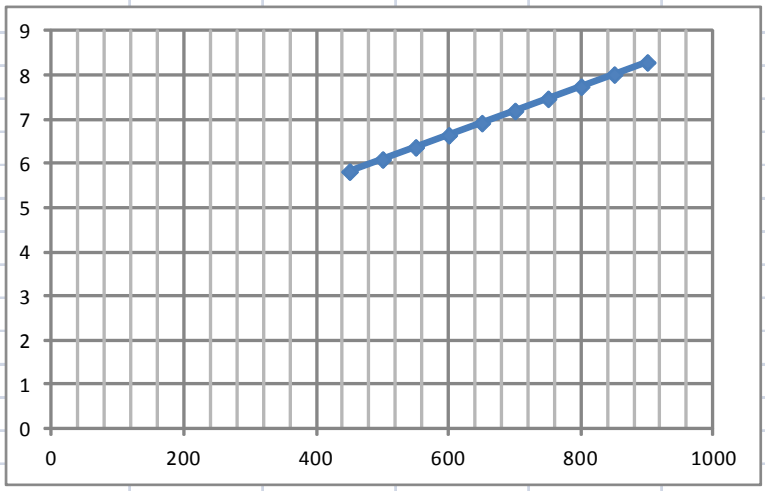

Figura 7.3 PWM-Voltajes estimados

ω se obtiene de la ecuación (7.5) convirtiendo primero a RPM multiplicando por la constante de velocidad $K_v = 935RPM/V$.

$$\omega = (mPWM + b)935 \tag{7.5}$$

La velocidad angular para estabilizar el cuadrorotor estará limitada a un valor máximo y mínimo aproximadamente de 5184.26 RPM y 7750.12 RPM, según la ecuación (7.5) con valores 400 y 900 para PWM según la figura 7-2.

Al realizar pruebas experimentales en donde el cuadrorotor comenzaba el vuelo con un valor de PWM=560,

$$\omega = ((0.005488485)560 + 3.349273)935 = 6005$$

considerando la ecuación (3.42) y despejando k_F.

$$k_F = \frac{mg}{4\omega^2} \tag{7.6}$$

Sustituyendo para masa $m = 1.1Kg$ del cuadrorotor, $g = 9.81\,{}^{m}/_{s^2}$ y velocidad $\omega = 6005\,rpm$ en (7.6).

$$k_F = \frac{1.1(9.81)}{4(6005)^2} = 74.8x10^{-9} \quad ,$$

se obtiene $k_F = 74.8x10^{-9}\left[\frac{N}{rpm^2}\right]$.

Para determinar la constante k_F se repitieron las pruebas en 5 ocasiones con la batería recientemente cargada y el valor de PWM se observaba por medio de una interfaz serial a través de la computadora, mientras el cuadrorotor empezaba a entrar en vuelo. Cabe mencionar que estas pruebas se realizan con un control manual, en donde desde la pc se envía el ancho de pulso adecuado para que el cuadrorotor entre en vuelo y este valor es visualizado en tiempo real por medio de la computadora.

El siguiente paso es proponer un algoritmo como programa principal que integre todos los módulos, componentes, ecuaciones dinámicas y de control para posteriormente programar en el entorno Code Composer Studio Version: 6.0.0.00190. Finalmente se prueba la constante calculada (7.6), se sintonizan las constantes PD y hacen pruebas de despliegue en el circuito para evaluar los resultados.

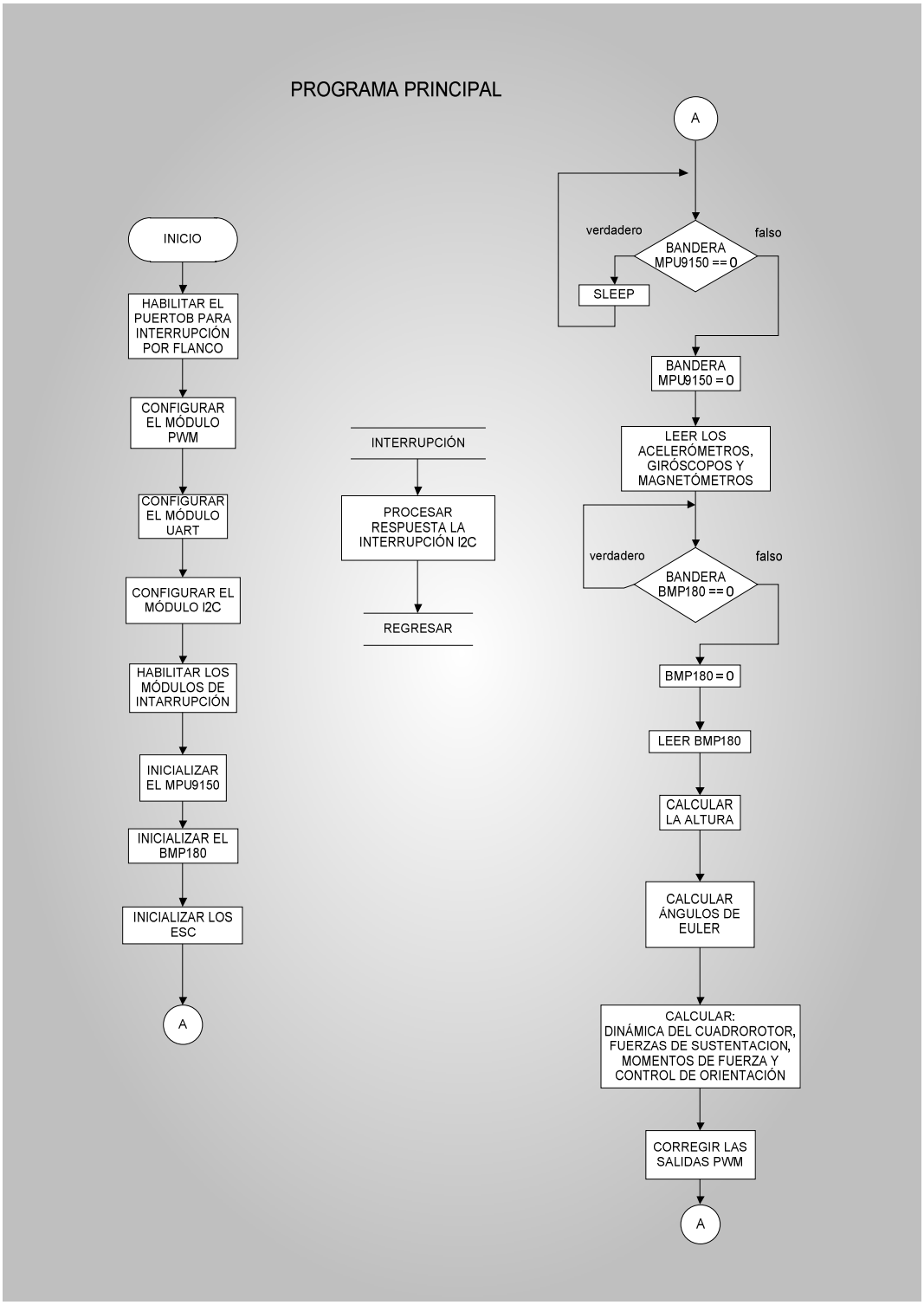

Figura 7.4 Algoritmo del programa principal

8 RESULTADOS DE LA IMPLEMENTACIÓN

Se integra el conjunto de elementos, batería, ESC, sensor hub con la tarjeta de desarrollo Tiva Launch Pad y se usan en conjunto para la operación de vuelo del cuadrorotor.

Se diseñan algoritmos para la programación y operación de los ESC, dinámica del cuadrotor y se programan en lenguaje C para la realización de pruebas experimentales.

Posteriormente, se hace la programación para la comunicación con los ESC comerciales por medio de secuencias PWM para su programación y se realiza una calibración de aceleración para todos los motores BL. Se tuvo éxito al variar la velocidad de los motores en el cuadrorotor. Con este paso logrado, se midieron voltajes en cada uno de los motores y se obtuvo la gráfica de la figura (7-2).

En las pruebas experimentales de medición de voltaje en relación con la variación del ancho de pulso (figura 7-2). Se observó que en todos los motores se presenta el mismo comportamiento y que hay muy poca diferencia entre cada uno (no mayores a una décima de volt). Se observó que se puede dividir en tres partes. En la primer parte, en el intervalo de 0 a 399 hay una aceleración de los motores BL en donde no existe la posibilidad de que el cuadrorotor comience a volar. En la segunda parte el intervalo 400 a 900 se observó que es prácticamente una línea recta, además dentro de ese intervalo el cuadrorotor, entra en vuelo y es útil para el control de orientación y estabilidad. La tercer parte no presenta cambios en el voltaje, se mantiene constante por tanto, no hay variación de velocidad y no es útil para el control del cuadrorotor. Estas pruebas se realizaron 4 veces para cada uno de los motores, se promedian y se presentan las gráficas de la figura 7-1.

Las condiciones ambientales eran las adecuadas ya que las pruebas se realizaron dentro del laboratorio de transductores. El ambiente era seguro ya que el cuadrorotor fue sujeto con hilos de nylon cuya longitud no permitía al cuadrorotor rebasar los 10 cm del suelo por lo que no había posibilidad de un accidente. Aún así en rara ocasión hubo intervenciones por parte del usuario al hacer los experimentos, por lo que las pruebas fueron una buena simulación de vuelo.

Se hace la programación para la comunicación I²C con los sensores de medición, utilizando funciones del Tiva Ware. Con las pruebas realizadas con los sensores de medición, giróscopos, acelerómetros y magnetómetros del MPU-9150 a través del software, se obtuvo una adquisición de datos a una taza de muestreo de 50 Hz para los giróscopos, acelerómetros y magnetómetros, que entregan una velocidad angular ω [rad/s], una aceleración lineal a [m/s^2] y la densidad de flujo magnético [T] respectivamente, además, de los ángulos de Euler para rotación, cabeceo y guiñada.. Las lecturas tomadas se pueden observar en las gráficas de las figuras 6-9, 6-10, 6-11.

Se diseña el algoritmo principal (Apéndice A) donde se relacionan los sensores de navegación, los ESC, los módulos PWM, los módulos I²C, el módulo UART, las ecuaciones dinámicas y de control para el desempeño del cuadrorotor.

Se propone una solución basada en las ecuaciones dinámicas y se integra con la técnica de control PID para corregir la velocidad angular de los motores (D. Mellinger et al, 2010) y se hace la programación de estas ecuaciones para su implementación en la Tiva C.

Se corre el programa principal del conjunto de módulos, ecuaciones dinámicas y de control en la tarjeta Tiva C, de esta forma y con ayuda de la interfaz serial por medio de la computadora, se depura el código fuente. Al encontrar errores de ejecución en el programa y haciendo pruebas físicas, se fueron corrigiendo errores de semántica en el programa. De forma iterativa se mejoró el código y se redujo para hacerlo más eficiente.

El primer paso para evaluar la ejecución del programa fue capturando los valores que el Tiva C calcula y entrega a los módulos PWM para corregir las velocidades angulares de los motores, mandando esos valores por puerto serial (figura 8-1) para graficar. Las condiciones son posición $\phi = 0$, $\theta = 0$ y $\psi = 0$ en el cuadrorotor y el programa $\phi = 0°$, $\theta = 0°$ y $\psi = 0°$ de igual forma. Las gráficas de las figuras, muestran valores muy aproximados a 560 de PWM que corresponde al valor de las pruebas experimentales donde el cuadrorotor comienza a volar. También se observa la sincronía entre las salidas PWM enviadas a los motores (figura 8-1 y 8-3).

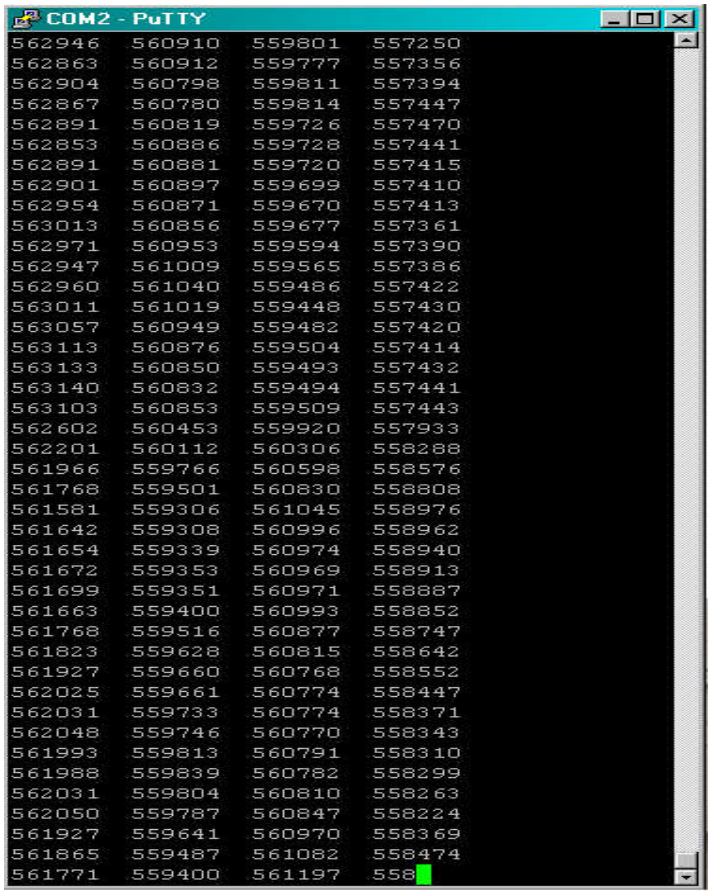

Figura 8.1 Visualización por puerto COM2 de las 4 salidas PWM.

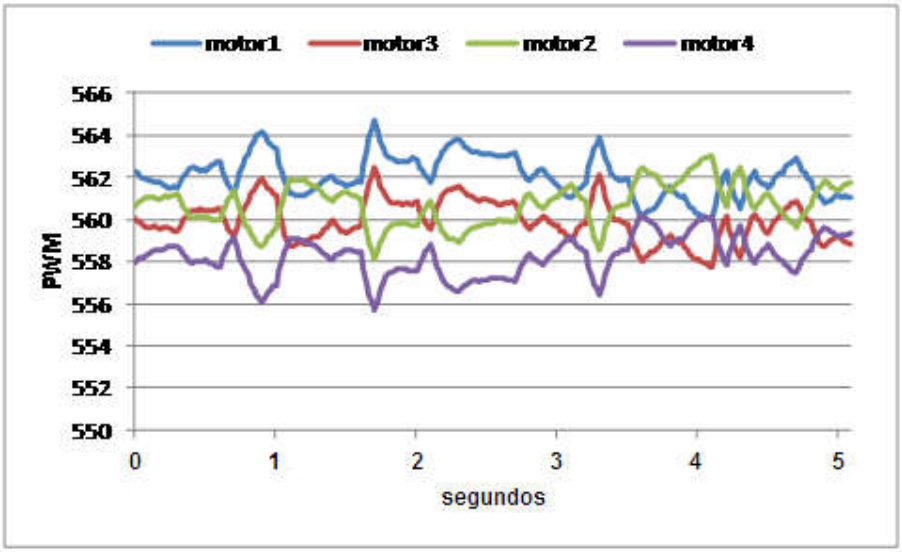

Figura 8.2 Salidas PWM para los 4 motores

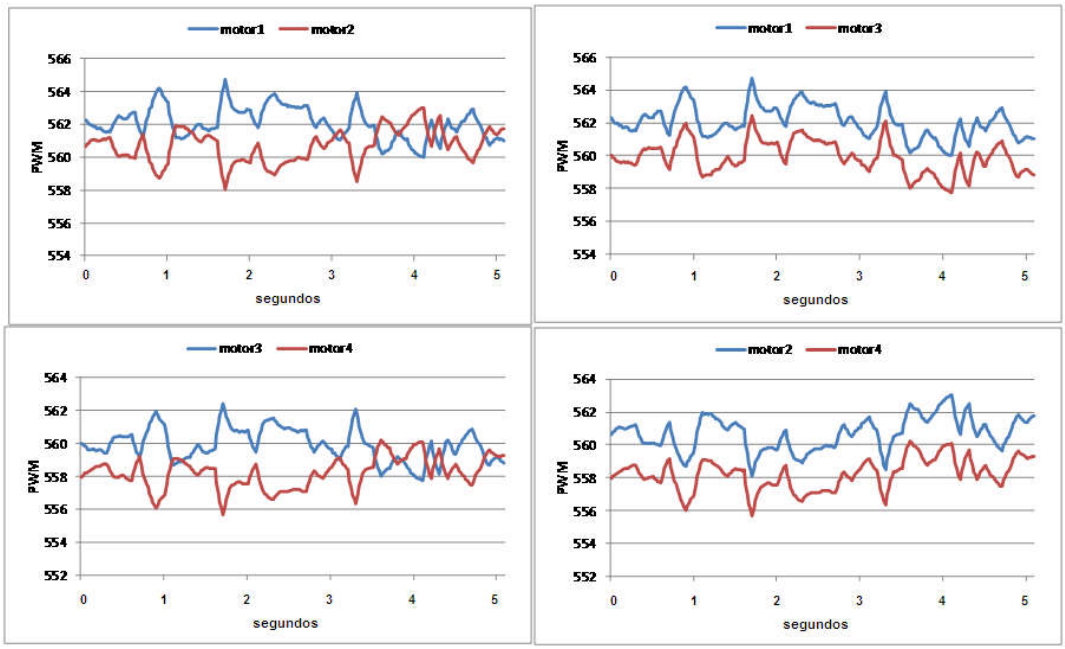

Figura 8.3 Comparación de las salidas PWM

Al rotar el cuadrorotor en $\psi = 45°$, $\phi = 0°$ y $\theta = 0°$, se observa como aumentan los valores de PWM en los motores 2 y 4, disminuyen en 1 y 3 para compensar la sustentación y que los momentos de fuerza en 2 y 4 hagan girar al cuadrorotor.

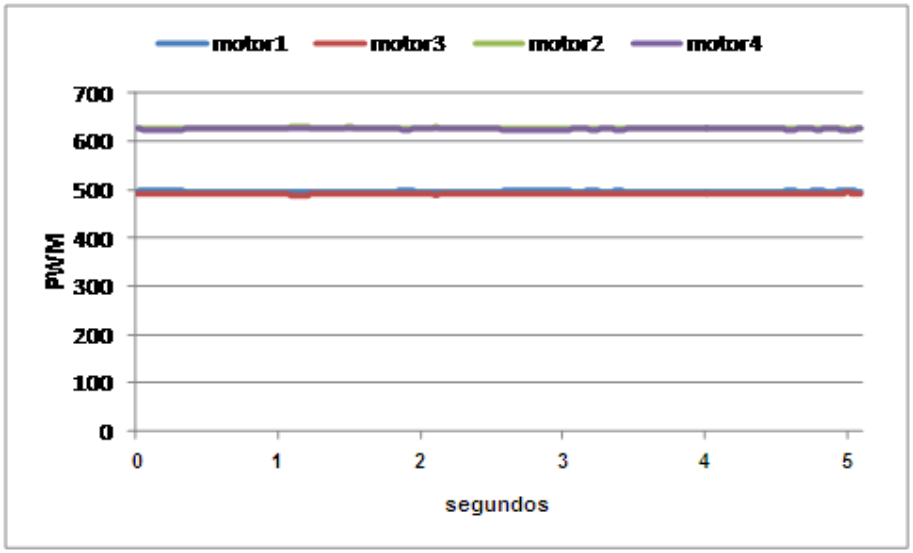

Figura 8.4 Rotación en el eje Z de 45°

De forma similar para $\phi = 30°$, $\theta = 0°$ y $\psi = 0°$ se observa que el motor 1 aumenta su valor de PWM y 3 disminuye para corregir el ángulo ϕ, mientras que 2 y 4 varían en valores de 550 y 560.

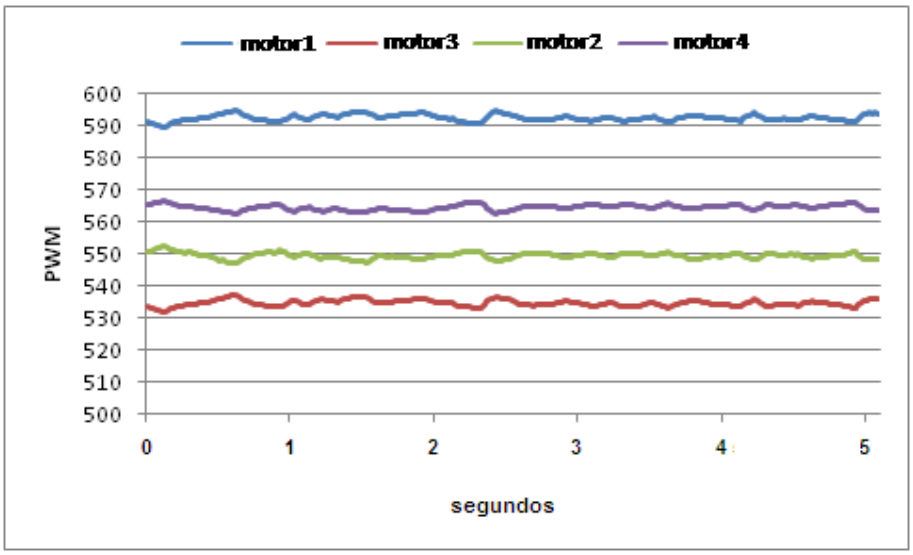

Figura 8.5 Rotación en el eje X de 30°

Para $\phi = 0°$, $\theta = 30°$ y $\psi = 0°$ se observa que el motor 2 aumenta su valor de PWM y 4 disminuye para corregir el ángulo θ, mientras que 1 y 3 se mantienen en valores cercanos a 560.

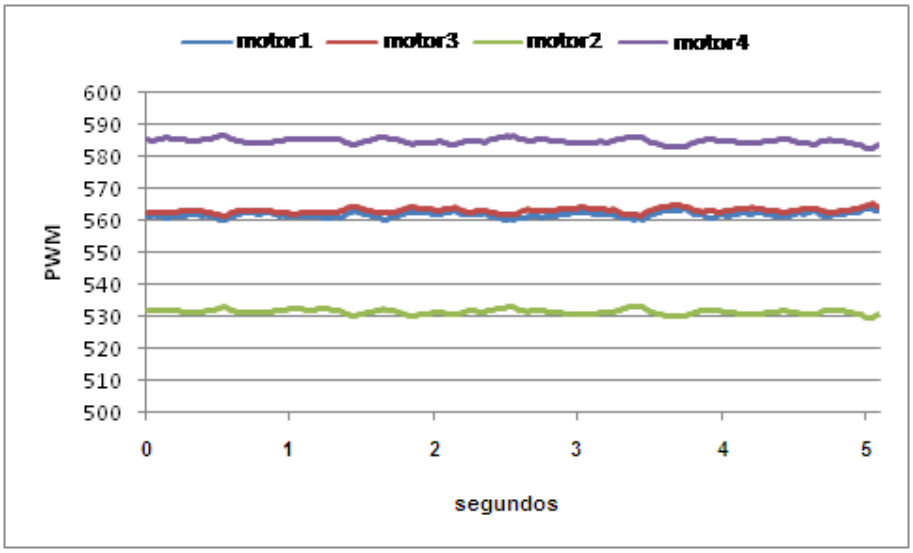

Figura 8.6 Rotación en el eje Y de 30°

En pruebas de vuelo, el cuadrorotor trata de estabilizarse pero tiende a caer y estrellarse si no hay ninguna intervención. Esto se debe a que no se tiene perfectamente caracterizados los motores BL, por lo que no se tiene la gráfica de la función de transferencia para sintonizar las constantes PID de forma matemática. En el caso de las hélices, no se tiene perfectamente caracterizado el modelo de hélices EPP1045 que se utilizó. La sintonización de manera práctica de la constante proporcional y derivativa mejoró el desempeño en la estabilización. El cuadrorotor oscila en vuelo estacionario, el uso en espacio libre sin ayuda de los hilos con que se sujetó, se puede evitar al ir sintonizando el control para que no resulte peligroso ya que puede estrellarse, romperse y/o dañar a alguna persona.

Durante las pruebas experimentales se midió el tiempo de vuelo estacionario del cuadrorotor y duraba entre 7 y 8 minutos de forma continua. De acuerdo a los datos de fabricante (Tabla 10-1), cada motor puede generar 850g de sustento con Propelas 1045 por cada motor, por lo que el peso límite que puede cargar es de 3400 g.

Por otra parte, el Tiva C cumplió con las necesidades ya que el tiempo que tarda para realizar todas las tareas y cálculos que fueron programados en lenguaje C, no rebasa los 20 ms del que dispone entre cada muestra que envían los sensores de navegación.

Para visualizar el rendimiento dinámico, se optó por hacer una evaluación intrusiva, en donde se envía un pulso de 1ms cuando el microcontrolador sale del modo de reposo (sleep) y otro pulso de 1ms cuando termina de ejecutar las lecturas, tareas, ecuaciones y vuelve a entrar en modo de reposo. Entre un pulso y otro el tiempo medido fue de 400 μs para la realización de operaciones, el resto de tiempo permanece en modo de reposo. En la figura 8-8 se muestra el rendimiento dinámico del Tiva C con el método intrusivo.

En cuanto a los recursos que utiliza es un 7% de memoria FLASH y un 9% de memoria RAM (figura 8-7), que es lo que muestra el entorno de programación Code Composer Studio después de compilar el código.

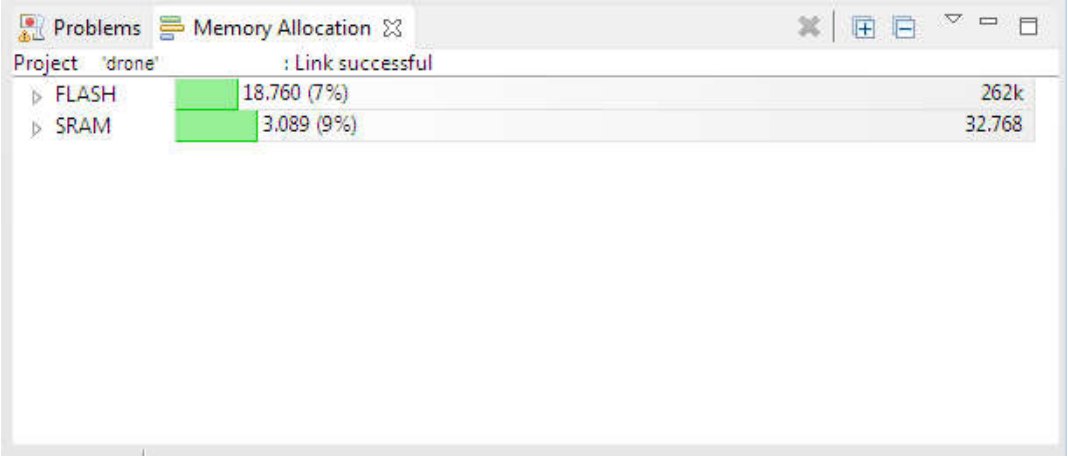

Figura 8.7 Uso de memoria FLASH y SRAM en el Tiva C

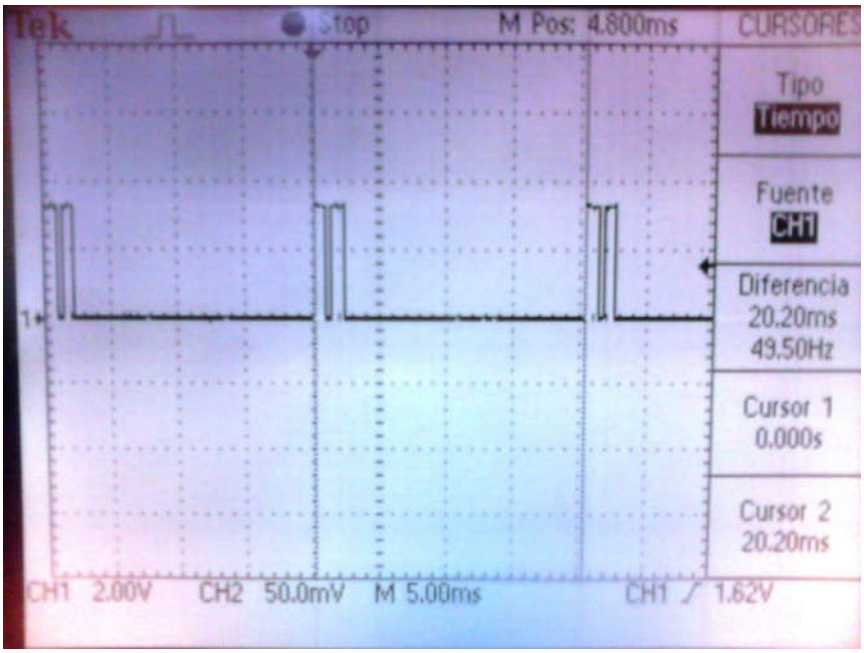

Figura 8.8 Rendimiento dinámico de la Tiva C

9 CONCLUSIONES

La principal contribución es la implementación de ecuaciones dinámicas y controlador PD en una tarjeta de desarrollo Tiva C como parte del desarrollo de un piloto automático que puede tener mejoramientos para futuras aplicaciones.

La utilización de motores con sensores para determinar la posición del rotor, como sensores Hall, puede simplificar las ecuaciones y facilitar el diseño del programa, pero tendrá la desventaja de ser un sistema más caro.

Como siguiente contribución se observa que cuando se usan motores que no traen sensores de posición para el rotor, no es posible la obtención de la velocidad angular de manera precisa. En este diseño, la única forma de corregir las RPM de los motores, es de forma indirecta por medio de los sensores de navegación. Por esta razón se propone el método de mínimos cuadrados y combinando aquel planteado por Mellinger, se corrigen las velocidades de los motores, pasando por el modelo dinámico y un control de orientación que usa un PD. Las señales de control para el PD son los ángulos y las componentes de velocidad angular que son obtenidas con los sensores de navegación. Una de las diferencias entre el trabajo propuesto y el de Mellinger es la frecuencia de muestreo con la que se trabaja. La frecuencia presentada en este trabajo es de 50 Hz, y es mucho menor a la de 1 KHz del trabajo de Mellinger. Es una desventaja ya que la precisión del PD es dependiente de la frecuencia de muestreo. La ventaja es que se pueden utilizar sensores más económicos y que no necesitan microcontroladores más veloces para procesar la información.

Otra contribución es con respecto a la implementación en una tarjeta Tiva Launch Pad, existe la ventaja de que se conecta a la computadora y se prueban los algoritmos implementados para el cuadrorotor en tiempo real. De esta forma se pueden corregir errores en el diseño del programa de acuerdo a la implementación física del cuadrorotor y a las ecuaciones.

Como trabajo futuro se propone: agregar más algoritmos y ecuaciones para mejorar la estabilidad del cuadrorotor ya que se usaron recursos por debajo del 10 % de SRAM y FLASH del Tiva C; aumentar la frecuencia de muestreo de los sensores de navegación a 100 Hz y caracterizar los motores y hélices para poder sintonizar de forma matemática las constantes del controlador PD.

Respecto a la circuitería, quedan pines para implementar otros dispositivos como un GPS para diseñar un control de posición y agregar una trayectoria 3D para la navegación del cuadrorotor. También un transmisor y receptor para una comunicación inalámbrica para enviar los datos a una estación terrena.

10 APÉNDICE

10.1 Apéndice A Programa principal

```
/*************************************************************
* PROGRAMA PRINCIPAL
* Creado: 2/01/2016
*
* Carlos Fernando Ortega Nava
*
*************************************************************/
//CABECERA
#include <math.h>
#include <stdint.h>//tipos de variable
#include <stdbool.h>
#include "inc/hw_memmap.h" //direcciones de memoria
#include "inc/hw_ints.h" //dirección de interrupción
#include "driverlib/debug.h"
#include "driverlib/gpio.h" //funciones y definiciones GPIO
#include "driverlib/interrupt.h" //funciones de interrupción
#include "driverlib/pin_map.h" //direcciones para pines
#include "driverlib/rom.h" //
#include "driverlib/sysctl.h" //funciones para el reloj
#include "driverlib/uart.h" //funciones para uart
#include "utils/uartstdio.h" //funciones para imprimir
#include "sensorlib/hw_mpu9150.h" //prototipos
#include "sensorlib/hw_ak8975.h"
#include "sensorlib/i2cm_drv.h"
#include "sensorlib/ak8975.h"
#include "sensorlib/mpu9150.h" //Macros
#include "sensorlib/comp_dcm.h" //Filtro complementario de matriz de cosenos
directores
#include "drivers/rgb.h"
//For UART
#include "driverlib/fpu.h"
//For pressure sensor
#include "driverlib/fpu.h"
#include "inc/hw_types.h"
#include "driverlib/systick.h"
#include "sensorlib/hw_bmp180.h"
#include "sensorlib/bmp180.h"
//For PWM module
#include "driverlib/pwm.h"
#include "inc/hw_gpio.h"

//DEFINICIONES
//*****************************************************************************
// MPU9150 I2C Dirección de esclavo
//*****************************************************************************
#define MPU9150_I2C_ADDRESS     0x68
//*****************************************************************************
// BMP180 I2C Dirección de esclavo
//*****************************************************************************
#define BMP180_I2C_ADDRESS      0x77
//*****************************************************************************
```

```
// Periodo de muestreo
// Distancia motor al centro de masa
//*****************************************************************************
#define defMuestreo 0.02
#define defDistancia 0.24
//*****************************************************************************
// Matriz de inercia
//*****************************************************************************
#define defI11 0.058
#define defI22 0.058
#define defI33 0.76
//*****************************************************************************
//  PD Constantes
//*****************************************************************************
#define      def_kp 3
#define      def_kd 1
//*****************************************************************************
// Constantes del modelo del motor
//*****************************************************************************
#define def_kF      0.00000660693425f//sustentación
#define def_kM      0.0000001367835979f//momento
#define def_km      0.05//motor
//*****************************************************************************
// PI
//*****************************************************************************
#define M_PI 3.14159265358979323846
//*****************************************************************************
// Estructura para el driver I2C maestro
//*****************************************************************************
tI2CMInstance g_sI2CInst;
//*****************************************************************************
// Estructura para el driver del sensor ISL29023
//*****************************************************************************
tMPU9150 g_sMPU9150Inst;
//*****************************************************************************
// Estructura para el driver del sensor BMP180
//*****************************************************************************
tBMP180 g_sBMP180Inst;
//*****************************************************************************
// Bandera para el BMP180
//*****************************************************************************
volatile uint_fast8_t g_vui8DataFlag;
//*****************************************************************************
// Estructura para el manejo de DCM
//*****************************************************************************
tCompDCM g_sCompDCMInst;
//*****************************************************************************
// Bandera para indicar que la transmisión del MPU9150 I2C está completa
//*****************************************************************************
volatile uint_fast8_t g_vui8I2CDoneFlag;
//*****************************************************************************
// Bandera para indicar que hay un error en la transmisión del MPU9150
//*****************************************************************************
volatile uint_fast8_t g_vui8ErrorFlag;
//*****************************************************************************
// Bandera para indicar que el dato del MPU9150 está listo para ser
// recuperado
//*****************************************************************************
volatile uint_fast8_t g_vui8DataFlag;
```

```
//*****************************************************************************
//
// Subrutina de interrupción del MPU9150.
//
//*****************************************************************************
void
MPU9150AppCallback(void *pvCallbackData, uint_fast8_t ui8Status)
{
// Si la transmisión es exitosa fija la bandera para indicar que la
// transmisión esta completa y el dato está listo
    if(ui8Status == I2CM_STATUS_SUCCESS)
    {
        g_vui8I2CDoneFlag = 1;
    }
// Almacena la más reciente condición de estado, en caso de que fuera un
// error
    g_vui8ErrorFlag = ui8Status;
}

//*****************************************************************************
//
// Rutina de interrupción del BMP180
//
//*****************************************************************************
void BMP180AppCallback(void* pvCallbackData, uint_fast8_t ui8Status)
{
    if(ui8Status == I2CM_STATUS_SUCCESS)
    {
        g_vui8DataFlag = 1;
    }
}

//*****************************************************************************
//
// Función que espera que las transmisiones estén completas para el MPU9150
//
//*****************************************************************************
void
MPU9150AppI2CWait(char *pcFilename, uint_fast32_t ui32Line)
{
// Procesador a dormir mientras índice que la transmisión esta completa
    while((g_vui8I2CDoneFlag == 0) && (g_vui8ErrorFlag == 0))
    {
    }
    // Llama al manejador de errores
    if(g_vui8ErrorFlag)
    {
        MPU9150AppErrorHandler(pcFilename, ui32Line);
    }
    // Borra la bandera para un siguiente uso
    g_vui8I2CDoneFlag = 0;
}

//*****************************************************************************
//
// Configuración del UART
//
//*****************************************************************************
```

```
void
ConfigureUART(void)
{
    // Habilitar periferico UART
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
    // Habilitar UART
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);
    // Configurar pines para UART
    ROM_GPIOPinConfigure(GPIO_PA0_U0RX);
    ROM_GPIOPinConfigure(GPIO_PA1_U0TX);
    ROM_GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);
    // Usar el oscilador interno de 16MHz
    UARTClockSourceSet(UART0_BASE, UART_CLOCK_PIOSC);
    // Baudaje
    UARTStdioConfig(0, 115200, 16000000);
}

//*****************************************************************************
//
//Configurar el I2C BMP180
//
//*****************************************************************************
void
Init_BMP180(void)
{
    // Inicializar BMP180.
    BMP180Init(&g_sBMP180Inst, &g_sI2CInst, BMP180_I2C_ADDRESS,
               BMP180AppCallback, &g_sBMP180Inst);
    // Esperar a que la solicitud de inicialización este completa
    while(g_vui8DataFlag == 0)
    {
       // Esperar a que la transmisión I2C este completa
    }
    // Reiniciar bandera
    g_vui8DataFlag = 0;
}

//*****************************************************************************
//
// Configurar I2C MPU9150
//
//*****************************************************************************
void
Init_MPU9150(void)
{
    // Inicialiar el Driver.
    MPU9150Init(&g_sMPU9150Inst, &g_sI2CInst, MPU9150_I2C_ADDRESS,
                MPU9150AppCallback, &g_sMPU9150Inst);
    // Esperar a que la transmisión este completa
    MPU9150AppI2CWait(__FILE__, __LINE__);
    // Configurar características de filtrado del sensor
    g_sMPU9150Inst.pui8Data[0] = MPU9150_CONFIG_DLPF_CFG_94_98;
    g_sMPU9150Inst.pui8Data[1] = MPU9150_GYRO_CONFIG_FS_SEL_250;
    g_sMPU9150Inst.pui8Data[2] = (MPU9150_ACCEL_CONFIG_ACCEL_HPF_5HZ |
                                  MPU9150_ACCEL_CONFIG_AFS_SEL_2G);
    MPU9150Write(&g_sMPU9150Inst, MPU9150_O_CONFIG, g_sMPU9150Inst.pui8Data,
                 3,MPU9150AppCallback, &g_sMPU9150Inst);

    // Esperar a que la transmisión este completa
```

```
    MPU9150AppI2CWait(__FILE__, __LINE__);
    // Configurar pin de interrupción del MPU9150
    g_sMPU9150Inst.pui8Data[0] = MPU9150_INT_PIN_CFG_INT_LEVEL |
                                 MPU9150_INT_PIN_CFG_INT_RD_CLEAR |
                                 MPU9150_INT_PIN_CFG_LATCH_INT_EN;
    g_sMPU9150Inst.pui8Data[1] = MPU9150_INT_ENABLE_DATA_RDY_EN;
    MPU9150Write(&g_sMPU9150Inst, MPU9150_O_INT_PIN_CFG,
                 g_sMPU9150Inst.pui8Data, 2, MPU9150AppCallback,
                 &g_sMPU9150Inst);
    // Esperar a que la transmisión este completa
    MPU9150AppI2CWait(__FILE__, __LINE__);
}

//*****************************************************************************
//
// Configurar las salidas PWM
//
//*****************************************************************************
void
Init_PWM(void)
{
      // Configurar el reloj para el módulo PWM
      SysCtlPWMClockSet(SYSCTL_PWMDIV_64);//40MHz/64=625KHz
      // Habilitar los periféricos usados por el módulo PWM
      SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
      SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);
      SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM0);
      SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM1);
      // Configurar los pines como salidas PWM
    GPIOPinConfigure(GPIO_PF1_M1PWM5);  //ROTOR1
    GPIOPinConfigure(GPIO_PF2_M1PWM6); //ROTOR2
    GPIOPinConfigure(GPIO_PF3_M1PWM7); //ROTOR3
    GPIOPinConfigure(GPIO_PB5_M0PWM3); //ROTOR4
    GPIOPinTypePWM(GPIO_PORTF_BASE, GPIO_PIN_1 | GPIO_PIN_2 |
                   GPIO_PIN_3);//ROTOR1|ROTOR2|ROTOR3
    GPIOPinTypePWM(GPIO_PORTB_BASE, GPIO_PIN_5 ); //ROTOR4
      // Configurar el generador de PWM para el modo count down mode
      PWMGenConfigure(PWM1_BASE, PWM_GEN_2, PWM_GEN_MODE_DOWN |
      PWM_GEN_MODE_NO_SYNC); //ROTOR1
      PWMGenConfigure(PWM1_BASE, PWM_GEN_3, PWM_GEN_MODE_DOWN |
      PWM_GEN_MODE_NO_SYNC); //ROTOR2,ROTOR3
      PWMGenConfigure(PWM0_BASE, PWM_GEN_1, PWM_GEN_MODE_DOWN |
      PWM_GEN_MODE_NO_SYNC); //ROTOR4
      // Fijar el periodo
      PWMGenPeriodSet(PWM1_BASE, PWM_GEN_2, 12500); //ROTOR1
      PWMGenPeriodSet(PWM1_BASE, PWM_GEN_3, 12500); //ROTOR2,ROTOR3
      PWMGenPeriodSet(PWM0_BASE, PWM_GEN_1, 12500); //ROTOR4
      // Fijar el ciclo de trabajo
      PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5, 0); //ROTOR1
      PWMPulseWidthSet(PWM1_BASE, PWM_OUT_6, 0); //ROTOR2
      PWMPulseWidthSet(PWM1_BASE, PWM_OUT_7, 0); //ROTOR3
      PWMPulseWidthSet(PWM0_BASE, PWM_OUT_3, 0); //ROTOR4
      // Iniciar los timer para el generador de PWM
      PWMGenEnable(PWM1_BASE, PWM_GEN_2); //ROTOR1
      PWMGenEnable(PWM1_BASE, PWM_GEN_3); //ROTOR2,ROTOR3
      PWMGenEnable(PWM0_BASE, PWM_GEN_1); //ROTOR4
}

//*****************************************************************************
```

```
//
// Habilitar las salidas PWM
//
//*****************************************************************************
void
PWM_ON(void)
{
      PWMOutputState(PWM1_BASE, PWM_OUT_5_BIT | PWM_OUT_6_BIT |
      PWM_OUT_7_BIT, true); //ROTOR1,ROTOR2,ROTOR3
      PWMOutputState(PWM0_BASE, PWM_OUT_3_BIT, true); //ROTOR4
}

//*****************************************************************************
//
// Retardo en milisegundos
//
//*****************************************************************************
void delayms(int ms) {
    SysCtlDelay( (SysCtlClockGet()/(3*1000))*ms ) ;
}

//*****************************************************************************
//
// Funciones prototipo para el ESC
//
//*****************************************************************************
// Fijar el ciclo de trabajo a 10%
void StartESC (void){
      PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5, 1250); //ROTOR1
      PWMPulseWidthSet(PWM1_BASE, PWM_OUT_6, 1250); //ROTOR2
      PWMPulseWidthSet(PWM1_BASE, PWM_OUT_7, 1250); //ROTOR3
      PWMPulseWidthSet(PWM0_BASE, PWM_OUT_3, 1250); //ROTOR4
      delayms(4000); //5sec
}

void AdjustThrottle (void){
      // Fijar el ciclo de trabajo a 20%
      PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5, 2500); //ROTOR1
      PWMPulseWidthSet(PWM1_BASE, PWM_OUT_6, 2500); //ROTOR2
      PWMPulseWidthSet(PWM1_BASE, PWM_OUT_7, 2500); //ROTOR3
      PWMPulseWidthSet(PWM0_BASE, PWM_OUT_3, 2500); //ROTOR4
      delayms(3000); //3s
      // Fijar el ciclo de trabajo a 10%
      PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5, 1250); //ROTOR1
      PWMPulseWidthSet(PWM1_BASE, PWM_OUT_6, 1250); //ROTOR2
      PWMPulseWidthSet(PWM1_BASE, PWM_OUT_7, 1250); //ROTOR3
      PWMPulseWidthSet(PWM0_BASE, PWM_OUT_3, 1250); //ROTOR4
      delayms(3000); //3s

}

//*****************************************************************************
//
// Programa principal
//
//*****************************************************************************
int
main(void)
{
```

```
//Contenedores
int_fast32_t i32IPart[16], i32FPart[16];
uint_fast32_t ui32Idx, ui32CompDCMStarted;
float pfData[16];
float *pfAccel, *pfGyro, *pfMag, *pfEulers;
//Para el sensor de presión
float fPressure, fAltitude;
int32_t i32IntegerPart;
int32_t i32FractionPart;
//Para instrumentar el cuadrorotor
int32_t i32IntPart[4];
int32_t i32FraPart[4];
int i=0;
int iconta;
// Inicializar punteros
pfAccel = pfData;
pfGyro = pfData + 3;
pfMag = pfData + 6;
pfEulers = pfData + 9;
//Para limitar lecturas en el sensor de presión
char conta=0;
//Modelo del motor
float fF1Sustentacion,fF2Sustentacion,fF3Sustentacion,fF4Sustentacion;
float fM1Momento,fM2Momento,fM3Momento,fM4Momento;
//Orientación
float    fDeltaOmegaFi,
         fDeltaOmegaTheta,
         fDeltaOmegaPsi;
//Entradas de control manual
float    fFi_des = 0;
float    fTheta_des = 0;
float    fPsi_des;
float    fp_des = 0;
float    fq_des = 0;
float    fr_des = 0;
float    fErr_p,
         fErr_q,
         fErr_r;
  float  fgErr_p,
         fgErr_q,
         fgErr_r;
  float fOmega1des,
         fOmega2des,
         fOmega3des,
         fOmega4des;
  float fFi, fTheta, fPsi;

  float  fPWM[4];

  //Componentes angulares
  float  fAngular_p,
         fAngular_q,
         fAngular_r;
  float  fAngular_pder,
         fAngular_qder,
         fAngular_rder;
  float  fretAngular_p,
         fretAngular_q,
         fretAngular_r;
```

```
  //
  float fcTheta, fsTheta;
  float fcFi, fsFi;

  float fVa1,fVa2,fVa3,  fVb1,fVb2,fVb3;

//
// Configurar reloj principal a  80 Mhz
//
ROM_SysCtlClockSet(SYSCTL_SYSDIV_2_5 | SYSCTL_USE_PLL | SYSCTL_XTAL_16MHZ
                   |SYSCTL_OSC_MAIN);

// Habilitar el puerto B para interrupción
//
ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);

// Inicialiar PWM
//
Init_PWM();    //Configurar los módulos PWM
PWM_ON();
ConfigGPIO_Instrum();

// Inicializar UART.
ConfigureUART();

// Habilitar periféricos I2C
ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_I2C3);
ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);

// Configurar los pines para I2C en el puerto D0 y D1
ROM_GPIOPinConfigure(GPIO_PD0_I2C3SCL);
ROM_GPIOPinConfigure(GPIO_PD1_I2C3SDA);

// Asociar la función I2C para esos pines
ROM_GPIOPinTypeI2CSCL(GPIO_PORTD_BASE, GPIO_PIN_0);
ROM_GPIOPinTypeI2C(GPIO_PORTD_BASE, GPIO_PIN_1);

// Configurar u habilitar la interrupción para la INT del MPU9150
ROM_GPIOPinTypeGPIOInput(GPIO_PORTB_BASE, GPIO_PIN_2);
GPIOIntEnable(GPIO_PORTB_BASE, GPIO_PIN_2);
ROM_GPIOIntTypeSet(GPIO_PORTB_BASE, GPIO_PIN_2, GPIO_FALLING_EDGE);
ROM_IntEnable(INT_GPIOB);

// Mantener activos estos módulos si se entra en modo sueño
ROM_SysCtlPeripheralClockGating(true);
ROM_SysCtlPeripheralSleepEnable(SYSCTL_PERIPH_GPIOB);
ROM_SysCtlPeripheralSleepEnable(SYSCTL_PERIPH_UART0);
ROM_SysCtlPeripheralSleepEnable(SYSCTL_PERIPH_I2C3);
ROM_SysCtlPeripheralSleepEnable(SYSCTL_PERIPH_PWM0);
ROM_SysCtlPeripheralSleepEnable(SYSCTL_PERIPH_PWM1);

// Habilitar todas las interrupciones
ROM_IntMasterEnable();

// Habilitar el módulo I2C
// Esta función se escribe después de inicializar el módulo
I2CMInit(&g_sI2CInst, I2C3_BASE, INT_I2C3, 0xff, 0xff,
         ROM_SysCtlClockGet());
```

```
//
Init_MPU9150();
Init_BMP180();

// Iniciar DCM a una taza de 50 Hz
// accel weight = .2, gyro weight = .8, mag weight = .2
CompDCMInit(&g_sCompDCMInst, 1.0f / 50.0f, 0.2f, 0.6f, 0.2f);

ui32CompDCMStarted = 0;

//Inicializar el ESC
StartESC ();

  //Acelerar a un valor inicial
  for (i=0;i<300;i++)
  {
        PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5, 1250+i); //ROTOR1
        PWMPulseWidthSet(PWM1_BASE, PWM_OUT_6, 1250+i); //ROTOR2
        PWMPulseWidthSet(PWM1_BASE, PWM_OUT_7, 1250+i); //ROTOR3
        PWMPulseWidthSet(PWM0_BASE, PWM_OUT_3, 1250+i); //ROTOR4
        delayms(10);
  }

  iconta=0;

while(1)
{

    // Modo sleep mientras espera a que el dato esté listo
    //
    while(!g_vui8I2CDoneFlag)
    {
         ROM_SysCtlSleep();
    }


    // Borrar bandera
    g_vui8I2CDoneFlag = 0;

    // Leer aceleraciones en m/s^2.
    MPU9150DataAccelGetFloat(&g_sMPU9150Inst, pfAccel, pfAccel + 1,
                             pfAccel + 2);

    // Leer velocidades angulares en rad/sec
    MPU9150DataGyroGetFloat(&g_sMPU9150Inst, pfGyro, pfGyro + 1,
                            pfGyro + 2);
   // Leer campo magnético en tesla
    MPU9150DataMagnetoGetFloat(&g_sMPU9150Inst, pfMag, pfMag + 1,
                               pfMag + 2);

    // Revisar si es la primera lectura
    if(ui32CompDCMStarted == 0)
    {
        ui32CompDCMStarted = 1;
        CompDCMMagnetoUpdate(&g_sCompDCMInst, pfMag[0], pfMag[1],
                             pfMag[2]);
        CompDCMAccelUpdate(&g_sCompDCMInst, pfAccel[0], pfAccel[1],
                           pfAccel[2]);
        CompDCMGyroUpdate(&g_sCompDCMInst, pfGyro[0], pfGyro[1],
```

```
                        pfGyro[2]);
        CompDCMStart(&g_sCompDCMInst);
    }
    else
    {
        // Realiza una actualización
        CompDCMMagnetoUpdate(&g_sCompDCMInst, pfMag[0], pfMag[1],
                             pfMag[2]);
        CompDCMAccelUpdate(&g_sCompDCMInst, pfAccel[0], pfAccel[1],
                           pfAccel[2]);
        CompDCMGyroUpdate(&g_sCompDCMInst, -pfGyro[0], -pfGyro[1],
                          -pfGyro[2]);
        CompDCMUpdate(&g_sCompDCMInst);
    }

        // Leer (Roll Pitch Yaw)
        CompDCMComputeEulers(&g_sCompDCMInst, pfEulers, pfEulers + 1,
                             pfEulers + 2);

         // Convertir a micro tesla
        pfMag[0] *= 1e6;
        pfMag[1] *= 1e6;
        pfMag[2] *= 1e6;

// Convertir flotante a enteros para visualizar en la computadora
        for(ui32Idx = 3; ui32Idx < 12; ui32Idx++)
        {

            // Flotante a entero sin decimal
            i32IPart[ui32Idx] = (int32_t) pfData[ui32Idx];
            // Multiplicar por 1000 para conservar tres decimales
            i32FPart[ui32Idx] = (int32_t) (pfData[ui32Idx] * 1000.0f);
                      // Restar enteros
            i32FPart[ui32Idx] = i32FPart[ui32Idx] -
                                (i32IPart[ui32Idx] * 1000);
            // hacer la parte decimal positiva si es negativa
            if(i32FPart[ui32Idx] < 0)
            {
                i32FPart[ui32Idx] *= -1;
            }
        }


        //Lectura del BMP180
        conta++;
        if (conta==2){
         conta=0;
                //
                BMP180DataRead(&g_sBMP180Inst, BMP180AppCallback,
                               &g_sBMP180Inst);
                while(g_vui8DataFlag == 0)
                {
                    // Esperar a que el nuevo dato esté disponible
                }

                // Reiniciar bandera.
                g_vui8DataFlag = 0;

            // Obtener una copia local de los últimos datos de la presión
```

```
                // del aire en formato float.
                    BMP180DataPressureGetFloat(&g_sBMP180Inst, &fPressure);


                    // Cálculo de altura
                    fAltitude = 44330.0f * (1.0f - powf(fPressure /
                                                101325.0f, 1.0f / 5.255f));
                    // Convertir flotante a enteros para imprimir
                    i32IntegerPart = (int32_t) fAltitude;
                    i32FractionPart =(int32_t) (fAltitude * 1000.0f);
                    i32FractionPart = i32FractionPart - (i32IntegerPart *
                                                         1000);
                    if(i32FractionPart < 0)
                    {
                        i32FractionPart *= -1;
                    }
                    UARTprintf("A %3d.%03d", i32IntegerPart,
                                i32FractionPart);
                    // retorno de carro
                    UARTprintf("\n");

                    }

/*************************************************************************
*
*IMPLEMENTACIÓN DE ECUACIONES DINÁMICAS PARA EL CONTROL DEL CUADROROTOR
*
*************************************************************************/
            //Error en Componentes angulares de velocidad
            fErr_p=(fp_des-fretAngular_p);
            fErr_q=(fq_des-fretAngular_q);
            fErr_r=(fr_des-fretAngular_r);

            fFi = pfEulers[0];
            fTheta = pfEulers[1];
            fPsi = pfEulers[2];

            //PD control
            fDeltaOmegaFi  = def_kp*(fFi_des-fFi)+def_kd*(fErr_p-
                                fgErr_p)/defMuestreo;
            fDeltaOmegaTheta = def_kp*(fTheta_des-fTheta)+def_kd*(fErr_q-
                                fgErr_q)/defMuestreo;
            fDeltaOmegaPsi       = def_kp*(fPsi_des-fPsi)+def_kd*(fErr_r-
                                fgErr_r)/defMuestreo;

            //Guardar componentes angulares anteriores
            fgErr_p = fErr_p;
            fgErr_q = fErr_q;
            fgErr_r = fErr_r;

            //Control de orientación
            fOmega2des = (629+32*0)-(fDeltaOmegaTheta)+   (fDeltaOmegaPsi);
            fOmega1des = (629+32*0)+(fDeltaOmegaFi)-      (fDeltaOmegaPsi);
            fOmega4des = (629+32*0)+(fDeltaOmegaTheta)+   (fDeltaOmegaPsi);
            fOmega3des = (629+32*0)-(fDeltaOmegaFi)-      (fDeltaOmegaPsi);

            //velocidad a PWM
            fPWM[0] = (60/(2*M_PI*935))*fOmega1des-3.349273f;
            fPWM[1] = (60/(2*M_PI*935))*fOmega2des-3.349273f;
```

```
fPWM[2] = (60/(2*M_PI*935))*fOmega3des-3.349273f;
fPWM[3] = (60/(2*M_PI*935))*fOmega4des-3.349273f;

fPWM[0] /= 0.005488485f;
fPWM[1] /= 0.005488485f;
fPWM[2] /= 0.005488485f;
fPWM[3] /= 0.005488485f;

//Filtros para no rebasar un valor máximo o mínimo
if (fPWM[0]>900) {fPWM[0] = 900;}
if (fPWM[0]<0) {fPWM[0] = 0;}
if (fPWM[1]>900) {fPWM[1] = 900;}
if (fPWM[1]<0) {fPWM[1] = 0;}
if (fPWM[2]>900) {fPWM[2] = 900;}
if (fPWM[2]<0) {fPWM[2] = 0;}
if (fPWM[3]>900) {fPWM[3] = 900;}
if (fPWM[3]<0) {fPWM[3] = 0;}

for(i = 0; i < 4; i++)
{
i32IntPart[i] = (int32_t) fPWM[i];
i32FraPart[i] = (int32_t) (fPWM[i] * 1000.0f);
i32FraPart[i] = i32FraPart[i] - (i32IntPart[i] * 1000);
}

//Salidas PWM
 PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5, 1250+fPWM[0]); //ROTOR1
 PWMPulseWidthSet(PWM1_BASE, PWM_OUT_6, 1250+fPWM[1]); //ROTOR2
 PWMPulseWidthSet(PWM1_BASE, PWM_OUT_7, 1250+fPWM[2]); //ROTOR3
 PWMPulseWidthSet(PWM0_BASE, PWM_OUT_3, 1250+fPWM[3]); //ROTOR4

//Modelo del motor
       fF1Sustentacion = def_kF*(fOmega1des*fOmega1des);
       fF2Sustentacion = def_kF*(fOmega2des*fOmega2des);
       fF3Sustentacion = def_kF*(fOmega3des*fOmega3des);
       fF4Sustentacion = def_kF*(fOmega4des*fOmega4des);

       fM1Momento = def_kM*(fOmega1des*fOmega1des);
       fM2Momento = def_kM*(fOmega2des*fOmega2des);
       fM3Momento = def_kM*(fOmega3des*fOmega3des);
       fM4Momento = def_kM*(fOmega4des*fOmega4des);

       fcFi = cosf(pfEulers[0]); fsFi = sinf(pfEulers[0]);
       fcTheta = cosf(pfEulers[1]); fsTheta = sinf(pfEulers[1]);

       //Componentes angulares
       fAngular_p =pfGyro[0]*fcTheta-pfGyro[2]*fcFi*fsTheta;
       fAngular_q =pfGyro[1]+pfGyro[2]*fsFi;
       fAngular_r =pfGyro[0]*fsTheta+pfGyro[2]*fcFi*fcTheta;

       //Dinámica de cuerpo rígido
       fVa1 = fAngular_p;
       fVa2 = fAngular_q;
       fVa3 = fAngular_r;

       fVb1 = defI11*fAngular_p;
       fVb2 = defI22*fAngular_q;
       fVb3 = defI33*fAngular_r;
```

```
                    fAngular_pder = defDistancia*(fF3Sustentacion-
                                    fF1Sustentacion)-(fVa2*fVb3-fVa3*fVb2);
                    fAngular_qder = defDistancia*(fF2Sustentacion-
                                    fF4Sustentacion)-(fVa3*fVa1-fVa1*fVb3);
                    fAngular_rder = (fM1Momento-fM2Momento+fM3Momento-
                                         fM4Momento)-(fVa1*fVb2-fVa2*fVb1);

                    fAngular_pder /= defI11;
                    fAngular_qder /= defI22;
                    fAngular_rder /= defI33;


                    fretAngular_p = defMuestreo*fAngular_pder;
                    fretAngular_q = defMuestreo*fAngular_qder;
                    fretAngular_r = defMuestreo*fAngular_rder;

////////////////////////////////////////////////////////////////
                    //Instrumentación. Imprimir salidas PWM
                    UARTprintf("%03d.%03d\n",i32IntPart[0], i32FraPart[0]);
                    UARTprintf("%03d.%03d ",i32IntPart[2], i32FraPart[2]);
                    UARTprintf("%03d.%03d ",i32IntPart[1], i32FraPart[1]);
                    UARTprintf("%03d.%03d \n",i32IntPart[3], i32FraPart[3]);
    }
}
```

10.2 Apéndice B Diagramas eléctricos

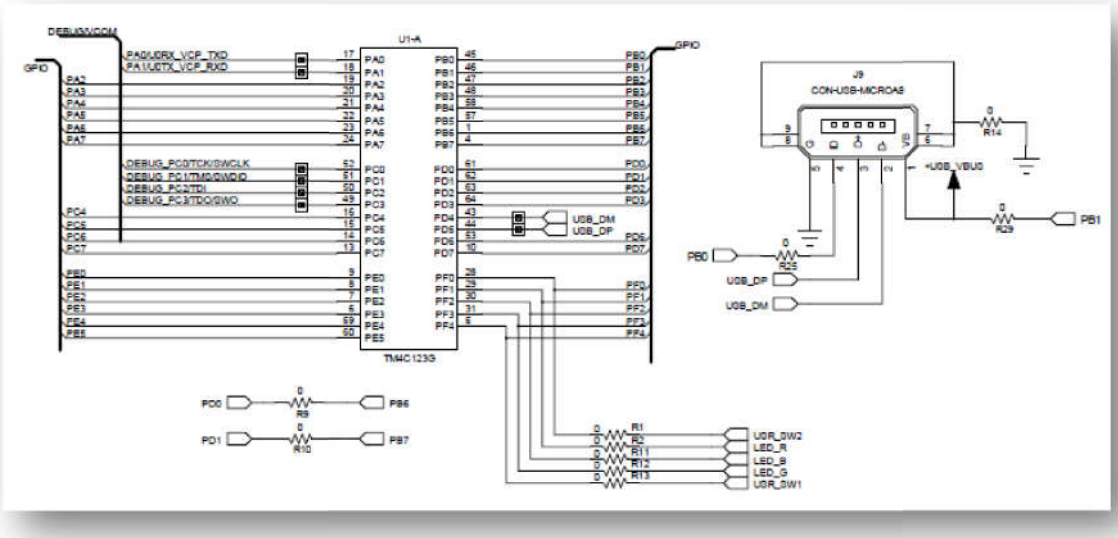

Figura 10.1 Microcontrolador TM4C123G

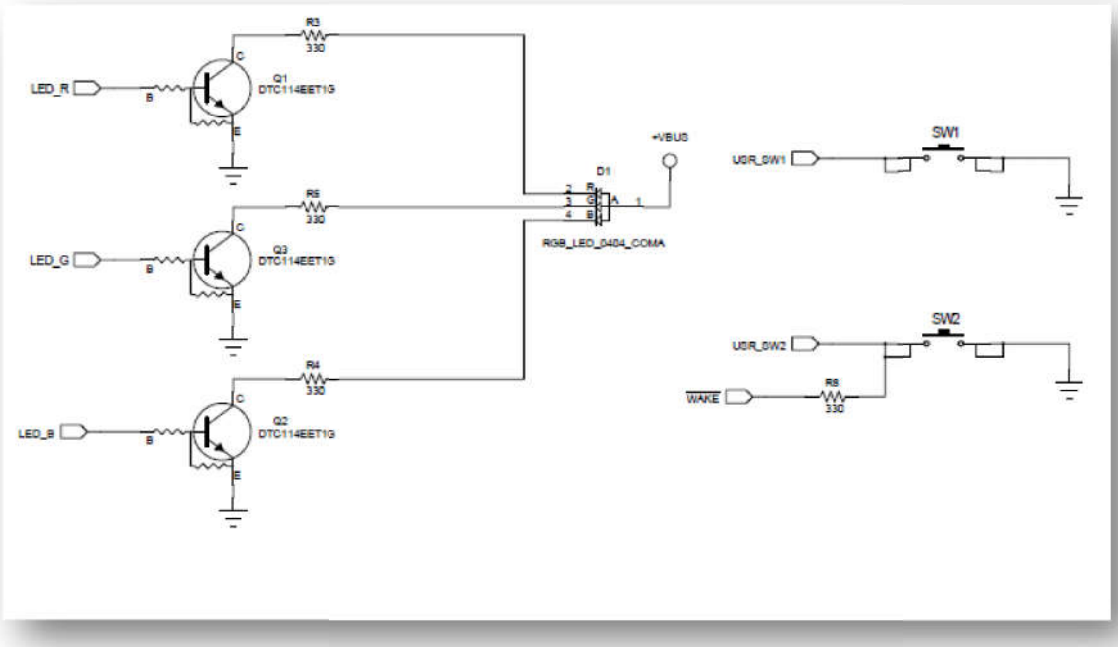

Figura 10.2 Botones y RGB LED

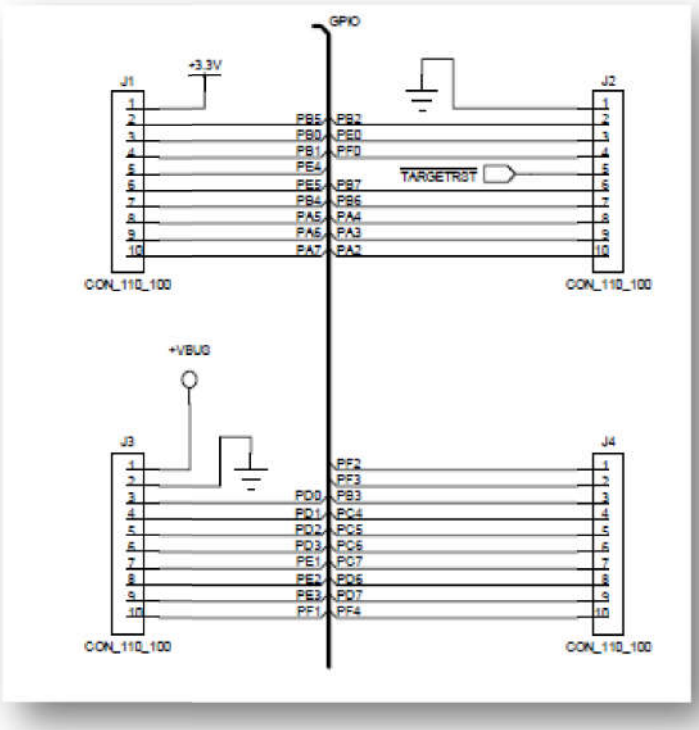

Figura 10.3 Headers JI, J2, J3 y J4 de la Tiva

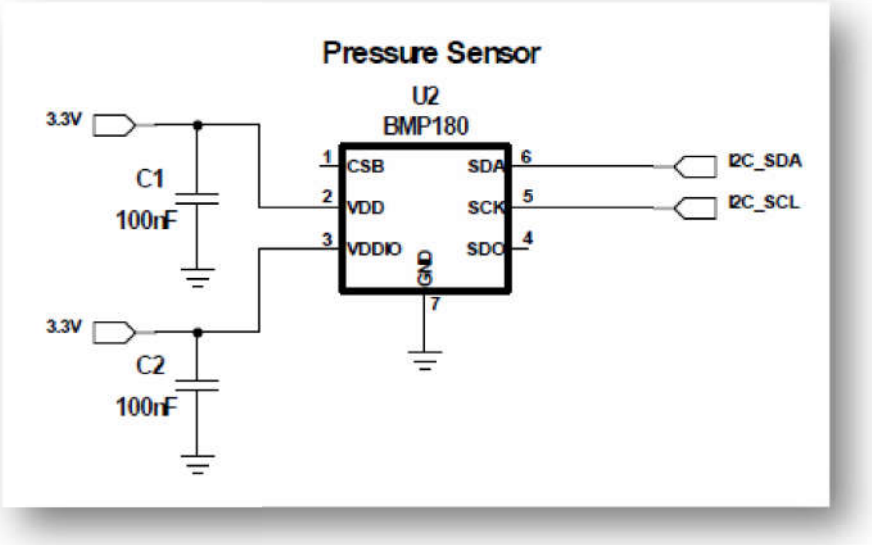

Figura 10.4 Sensor de presión BMP180

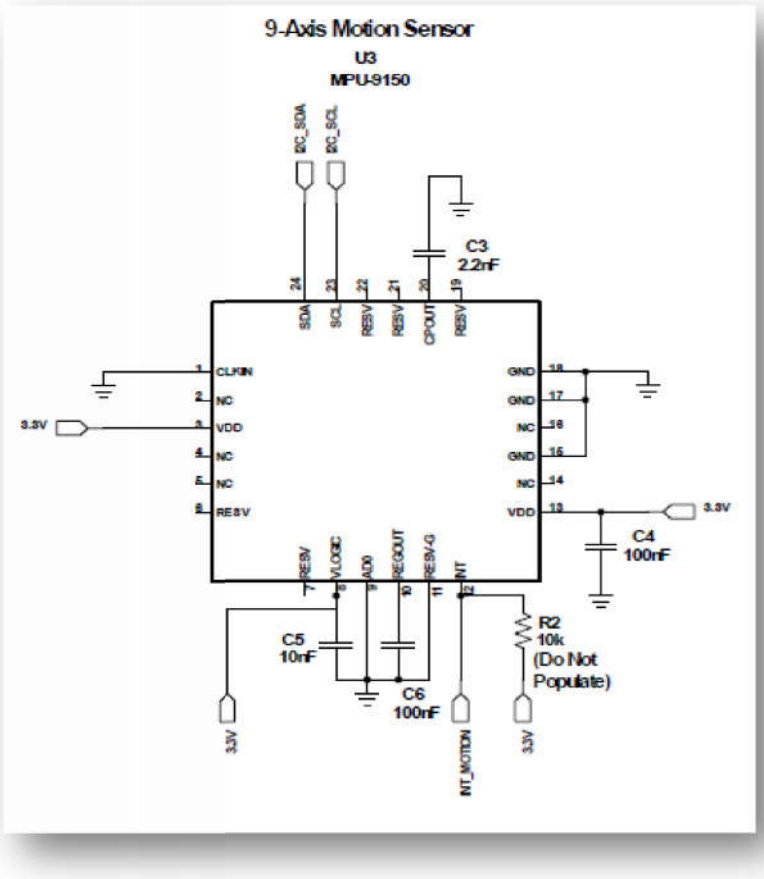

Figura 10.5 Unidad de medición inercial MPU-9150

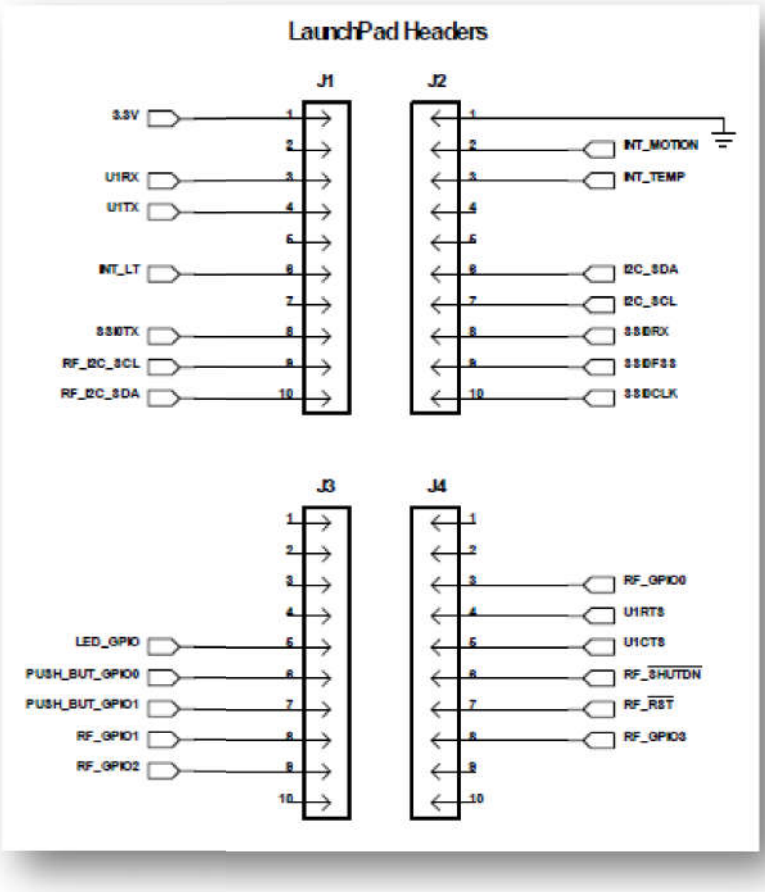

Figura 10.6 Headers J1, J2, J3, J4 del Sensor Hub

10.3 Apéndice C Datos técnicos

Tabla 10-1 Características del cuadrorotor

CARACTERÍSTICAS DEL CUADROROTOR	
Plataforma SK450	
Ancho	450mm
Alto	80mm
Peso	300g
BL Multistar Motor	
Constante de Velocidad K_V	2213 – 935 RPM/V
Máxima Corriente	15 A
Corriente sin carga	0.4 A
Resistencia	180 ohm
Potencia	200 W
Peso	55g
Sustento máximo	850g
Turnigy ESC	

Constante de Corriente	20 A
Voltaje de entrada	2-6 V
Salida del BEC	5V/4A
PWM	8KHz
Peso	30g
Hélices EPP1045	
Diámetro	8 pulgadas
Inclinación	4.5 pulgadas
Peso c/u	8 gramos
Baterías	
Capacidad	2200 mAh
Voltaje	11.1 V
Corte de carga	12.6 V
Corte de descarga	9.0 V
Constante de descarga	1.5 C (3.3A)
Taza de carga máxima	1C (2.2 A)
Taza estándar	0.5C (1.1 A)
Largo	100mm
Ancho	33mm
Altura	22mm
Tiva C Lanch Pad TM4C123G	
ARM Cortex	M4 Con punto flotante
Frecuencia	80MHz
Arquitectura	32 bits
Memoria de programa	256KB Flash
Memoria de datos	32KB SRAM
Memoria de datos	2KB EEPROM
PWM	8 módulos
ADC	12 canales – 12 bits
UART	8 módulos
USB	Full/Hight Speed
I2C	10 módulos
SPI	4 módulos
CAN	2 módulos
Comparadores analógicos	3 módulos
Perro Guardián	2 módulos
Temporizadores	8 módulos - 32bits
Sensor Hub Booster Pack	
Sensor de movimiento	MPU-9150
Acelerómetro	3-ejes
Giróscopo	3-ejes
Magnetómetro	3-ejes
Sensor de temperatura	TMP006
Sensor de presión barométrica	Bosh BMP180
Sensor de ambiente y luz infrarroja	ISL29023
Sensor de humedad	SHT21

La interfaz de conexión Sensor Hub está diseñada para ser ensamblada con la tarjeta de desarrollo Tiva C, ésta última cuenta con cuatro conectores de 10 pines cada uno denominados J1, J2, J3, J4 y corresponde a los cuatro conectores del Sensor Booster Pack conectados uno a uno, J1 con J1, J2 con J2, J3 con J3 y J4 con J4. Se presentan las interfaces correspondientes en las tablas 10-2 y 10-3.

Tabla 10-2 Interfaz de conexión del Sensor Hub Booster Pack

Sensor Hub Booster Pack Interfaz de conexión				
	J1	J3	J4	J2
1	3.3V IN			Ground
2				Interrupt motion
3	UART RX		RF GPIO	Interrupt tempt
4	UART TX		UART RTS	
5		User LED	UART CTS	
6	Interrupt_LT	User Button	RF Shutdown	Sensor I2C
7		User Button	RF Reset	Sensor I2C
8	SSI TX	RF GPIO	RF GPIO	SSI RX
9	RF I2C	RF GIPIO		SSI SS
10	RF I2C			SSI CLK

Tabla 10-3 Interfaz de conexión del Tiva Launch Pad

Tiva Launch Pad Interfaz de conexión				
	J1	J3	J4	J2
1	3.3V	VBUS	PF2	GND
2	PB5	GND	PF3	PB2
3	PB0	PD0	PB3	PE0
4	PB1	PD1	PC4	PF0
5	PE4	PD2	PC5	RST
6	PE5	PD3	PC6	PB7
7	PB4	PE1	PC7	PB6
8	PA5	PE2	PD6	PA4
9	PA6	PE3	PD7	PA3
10	PA7	PF1	PF4	PA2

En la tabla 10-4 se muestran los pines de la tarjeta TIVA Launch Pad y los recursos asociados a esos pines como botones y LEDs.

Tabla 10-4 Botones y LED para el usuario

Botones y RGB LED		
PF4	GPIO	SW1
PF0	GPIO	SW2
PF1	GPIO	RGB LED (R)
PF2	GPIO	RGB LED (B)
PF3	GPIO	RGB LED (G)

11 REFERENCIAS

A. Modirrousta and M. Khodabandeh (2015). Adaptive non-singular terminal sliding mode controller: new design for full controlof the quadrotor with external disturbances
Transactions of the Institute of Measurement and Control, sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/0142331215611210

A. L. Salih, M. Moghavvemi, H. A. F. Mohamed and K. Sallom Gaeid, (2010). Modelling and pid controller design for a quadrotor unmanned air vehicle.
Procedings of 2010 Cluj-Napoca Conference. Volume 1. pp 1-5
DOI: 10.1109/AQTR.2010.5520914

B. Xian, B. Zhao, Y. Zhang and X. Zhang (2015). A low-cost hardware-in-the-loop-simulation testbed of quadrotor UAV and implementation of nonlinear control schemes. Robotica,
Available on CJO 2015, pp 1 - 25
doi:10.1017/S0263574715000727

C. Diao, B. Xian, Q. Yin, W. Zeng, H. Li, and Y. Yang (2011). A nonlinear adaptive control approach for quadrotor uavs.
Proceedings of 2011 8th Asian Control Conference (ASCC),2011.
Vol. MoA6.4, pp. 223-228

C. Enemark (2014). Drones, Risk and Perpetual Force
Ethics & International Affairs, 28, no. 3 (2014), pp. 365–381.Carnegie Council for Ethics in International Affairs
doi:10.1017/S0892679414000446

C. Rosales, D. Gandolfo, G. Scaglia, M. Jordan and R. Carelli (2015). Trajectory tracking of a mini four-rotor helicopter in dynamic environments - a linear algebra approach.
Robotica, 33, pp 1628-1652 doi:10.1017/
S0263574714000952

C. Ortega, J. Álvarez, (2015). Implementación de un sistema embebido para el control de trayectorias de un mini cuadro-rotor.
SOMIXXX Congreso de Instrumentación, pp. 1-8

D. Lee, H. Jin Kim, S. Sastry (2009). FeedBack Linearization vs. Adaptive Sliding Mode Control for a Quadrotor Helicopter,
Publisher Institute of Control, Robotics and Systemsand The Korean Intitute of Electrical Engineers, Volume 7, ISSN 1598-6446, pp 419 - 418

D. Gurdan, J. Stumpf, M. Achtelik, K. Doth, G. Hirzinger, D. Rus (2007).
Energy-efficient Autonomous Four-rotor Flying Robot Controlled at 1 kHz
Proceedings of the IEEE International Conference on Robotics and Automation pp 361-366
1-4244-0602-1/07/$20.00 ©2007 IEEE

D. Cremers (2014), Computer Vision: ACCV 2014
Volume 9003-9007, Publisher Springer

D. Mellinger, N. Michael, M. Kumar. (2012). Trajectory Generation and Control for Precise Aggressive Maneuvers with Quadrotors. The international Journal of Robotics Research April 2012, Vol 31 No. 5, pp. 664-674

E. García Breijo (2010), COMPILADOR C CCS Y SIMULADOR PARA PROTEUS PARA MICROCONTROLADORES PIC
Primera Edición, pp. 153-163

E. Altug, J. P. Ostrowsk, C. J. Taylor (2005). Control of a Quadrotor Helicopter Using Dual Camera Visual Feedback
The International Journal of Robotics Research Vol. 24, No. 5, pp. 329-341
DOI: 10.1177/0278364905053804

G. M. Hoffmann, S. L. Waslander (2008). Quadrotor Helicopter Trajectory Tracking Control
American Institute of Aeronautics and Astronautics, pp 1-14
ISBN:9781605608082

G. Scaglia, L. Q. Montoya, V. Mut and Fernando di Sciascio (2009). Numerical methods based controller design for mobile robots. Robotica, 27, pp 269-279
doi:10.1017/S0263574708004669

G. V. Raffo, M. G. Ortega and F. R. Rubio (2008) Backstepping/Nonlinear H_∞ Control for path Tracking of Quadrorotor Unmanned Aerial Vehicle.
2008 American Control Conference Westin Seattle Hotel, Seattle,
Washington, USA. ISBN 978-1-4244-2078-0 Vol ThC12.6 pp. 3356-3361

G. J. E. Scaglia, V. A. Mut, M. Jordan, C. Calvo, L. Quintero (2008). Robust-control-based controller design for a mobile robot
© Springer Science+Business Media B.V. 2008, J Eng Math (2009) 63:17–32
DOI 10.1007/s10665-008-9252-0

H. Romero, S. Salazar and R. Lozano (2012). Visual servoing applied to real-time stabilization of a multi-rotor UAV. Robotica, 30, pp 1203-1212
doi:10.1017/S026357471200001X

H. Xie, A. F. Lynch and M. Jagersand (2015). Dynamic IBVS of a rotary wing UAV using line features.
Robotica, Available on CJO, pp 1-18
doi:10.1017/S0263574714002707

I. Henderson (2010). Civilian Intelligence Agencies and the Use of Armed Drones.
Yearbook of International Humanitarian Law, 13, pp 133-173
doi:10.1007/978-90-6704-811-8_4

J. Anderson (2000), INTRODUCTION TO FLIGHT
7th Edition, I
SBN-13: 978-0073380247

J. A. Guerrero, J. A. Escareno, Y. Bestaoui, (2013). Quad-Rotor Mav Trayectory Planning in Wind Fields.
Publisher IEEE, 2013IEEE International Conference on Robotics and Automation (ICRA), ISBN 978-1-4673-5641-1, pp. 778 - 783

J. Engel and T. Schöps and D. Cremers (2014). LSD-SLAM: Large-Scale Direct Monocular SLAM
13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part II,
ISBN 978-3-319-10604-5, DOI 10.1007/978-3-319-10605-2_54

J. Engel, J. Stückler, D. Cremers (2014). Large-Scale Direct SLAM with Stereo Cameras
13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part II,pp 834-849
ISBN 978-3-319-10604-5, DOI 978-3-319-10605-2_54

J. E. Gómez-Balderas, S. Salazar, J. A. Guerrero and R. Lozano (2014). Vision-based autonomous hovering for a miniature
quad-rotor. Robotica, 32, pp 43-61
doi:10.1017/S0263574713000611

J. E. Normey-Ricoa, I. Alcala, J. Gomez-Ortega, E. F.Camachob (2001). Mobile robot path tracking using a robust PID controller
Control Engineering Practice 9 (2001) pp 1209–1214 2001 Published by Elsevier Science Ltd.
PII: S0967-0661(01)00066-1

J. Witenburg (2007), DINAMICS OF MULTYBODY SYSTEMS
Second Editión, Editorial Springer, Library of Congress Control Number: 2007932332
ISBN 978-3-540-73913-5

K. J. Yonn and N. S. Goo, (2011). Development of Small Autonomous Flying Robot With Four-Rotor System, Print The 15th International Conference on Advanced Robotics,
ISBN 978-1-4577-1158-9, pp. 150 - 154

M. A. M. Basri, A. R. Husain and K. A. Danapalasingam (2014). Intelligent adaptive backstepping control for MIMO uncertain non-linear quadrotor helicopter systems
Transactions of the Institute of Measurement and Control, sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/0142331214538900

M. S. Grewal, L. R. Weill, A. P. Andrews (2001). Global Positioning Systems, Inertial Navigation, and Integration. Copyright 2001 John Wiley & Sons, Inc. Print ISBN 0-0471-35032-X.

M. Santos, V. López (2010). Intelligent Fuzzy Controller of a Cuadrorotor
Procedings 2010 International Conference on Intelligent Systems and Knowledge Engineering (ISKE), Hangzhou.
ISBN 978-1-4244-6791-4 pp. 141-146

N. Sinan Özbek, M. Önkol3 and M. Önder Efe (2015). Feedback control strategies for quadrotor-type aerial robots: a survey
Transactions of the Institute of Measurement and Control, sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/0142331215608427

O. J. Oguntoyinbo, (2009). PID Control of Brushhless DC Motor and Robot Trayectory Planing and Simulation whith MATLAB®/SIMULINK®, University of applied Sciences

P. Castillo, R. Lozano (2004). Stabilization of a mini-rotorcraft having four rotors.
Proceedings IEEE/RSJ International Conference en Intelligent Robots and Systems.
ISBN 0-7803-8463-6 Volume 3. pp 2693-2698

R. Austin (2010). UNMANNED AIRCRAFT SYSTEMS UAVS DESIGN, DEVELOPMENT AND DEPLOYMENT
A John Wiley and Sons, Ltd., Publication
ISBN 9780470058190 (H/B)

R. Xu and Ü. Özgüner (2006). Sliding mode control of a quadrotor helicopter.
Proceedings of the 45th IEEE Conference on Decision & Control San Diego, CA, USA, 2006,
ISBN 1-4244-0171-2, Vol,FrA10.4 pp. 4957-4962

S. Bouabdallah, P. Murrieri, R. Siegwart (2004). Design and Control of an Indoor Micro Quadrotor
Proceedings of the 2004 IEEE, International Conferenceon Robotics8 Automation pp 4393-4398
0-7803-8232-3/04/$17.0002004 IEEE

S. Bouabdallah and R. Siegwart, (2005). Backstepping and Sliding-Mode Techniques Applied to an Indoor Micro Quadrotor, Proceedings of the 2005 IEEE, International Conference on Robotics and Automation ISBN 0-7803-8914-X, pp. 2247 - 2252

S. Bouadallah, A. Noth, R. Siegwart (2004). PID VS LQ Control Techniques Applied to an Indoor Micro Quadrotor.
Proceedings of 2004 IEEE/RSJ International Conference on Intelligent
Robots and Systems, Volume 3 ISBN 0-7803-8463-6, pp. 2451 – 2456

T. Madani and A. Benallegue (2006). Backstepping control for a quadrotor helicopter.
Proceedings of the 2006 IEEE/RSJ; International Conference on Intelligent
Robots and Systems, Beijing, China, 2006; ISBN 1-4244-0258-1, pp. 3255-3260

V. K. R. Singh Patel, A. K. Pandey,(2013). Modeling and Performance Analysis of PID Controlled BLDC Motor and Differents Schemes of PWM Controlled BLDC Motor, International Journal of Scientific and Reserch Publications, Volume 3, ISSN 2250 – 3153

V. Usenko, J. Engel, J. Stückler, and D. Cremers (2015). Reconstructing Street-Scenes in Real-Time From a Driving Car
Proc. of the Int. Conference on 3D Vision (3DV), pp 1-8

Z. Zuo (2012). Adaptive trajectory tracking control design with command filtered compensation for a quadrotor
Journal of Vibration and Control 19(1), pp 94–108
DOI: 10.1177/1077546311431270

Bosch BMP180 Digital Pressure Sensor Data Sheet Document Number BST-BMP180-DS000-12, Revison2.8
Technical Reference Code 0 273 300 244
Fabricante: Bosch

MPU-9150 Product Specification Revision 4.3,
Document Number PS-MPU-9150A-00
Fabricante: IvenSense

Tiva TM4C123GH6PM Microcontroller Data Sheet,
DS-TM4C123GH6PM-15842.2741
Fabricante: Texas Instruments

TURNIGY Manual for Brushless Motor Speed Controller
Fabricante Turnigy http://www.turnigy.com,