UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DISEÑO E IMPLEMENTACIÓN EN FPGA DE UN SISTEMA DE CONTROL DE ORIENTACIÓN PARA UN SIMULADOR DE VUELO SATELITAL

T E S I S

PARA OBTENER EL GRADO ACADÉMICO DE

INGENIERO EN COMPUTACIÓN

P R E S E N T A

JOSÉ FRANCISCO OSORIO JIMÉNEZ

DIRECTOR DE TESIS: DR. ESAÚ VICENTE VIVAS

MÉXICO, D.F. 2013

JURADO ASIGNADO.

PRESIDENTE: **JOSÉ ALBERTO TEMPLOS CARBAJAL**

VOCAL: **ESAÚ VICENTE VIVAS**

SECRETARIO: **NORMA ELVA RODRIGUEZ CHAVEZ**

SUPLENTE PRIMERO: **PATRICIA HONG QUEZADA**

SUPLENTE SEGUNDO: **JORGE VALERIANO ASSEM**

Índice

Introducción

Sin lugar a duda el desarrollo de tecnología espacial es una asignatura donde confluyen y se fusionan varias áreas de la ciencia y la ingeniería. A partir del desarrollo y puesta en órbita del primer satélite artificial *Sputnik* en 1957, el desarrollo de la tecnología espacial ha venido avanzando cada día más hasta la época actual, asimilando y aprovechando a lo largo de todo este tiempo los grandes avances, que en paralelo se han conseguido en la microelectrónica, la computación, los procesos de diseño y manufactura, ciencias de materiales, etcétera. Esta simbiosis integra por un lado el desarrollo de tecnología espacial con el desarrollo de nuevas tecnologías y procesos, ha permitido al hombre alcanzar objetivos asombrosos y crear vehículos espaciales que cumplen con misiones que cada día aumentan sus exigencias técnicas.

En el caso particular del desarrollo de subsistemas satelitales, este constante desarrollo se ha visto aprovechado ampliamente. Actualmente las empresas de desarrollo de tecnología espacial e incluso agencias espaciales tan importantes como la misma NASA de EE.UU., han adoptado la filosofía de trabajo *cheaper, better, faster* en todos sus productos y desarrollos. Esto conlleva la instrumentación de procesos y técnicas de diseño y desarrollo que permitan por un lado la reducción de costos, la reducción de tiempo de desarrollo así como el mejoramiento en la calidad de los productos.

El campo satelital no puede estar ajeno a todas estas tendencias. Particularmente, en el caso del subsistema de control de orientación, la búsqueda y desarrollo de técnicas que permitan la aceleración del desarrollo tanto del software como del hardware asociado con un alto nivel de confianza y con la capacidad de ser validado en tierra, justo antes de ser puesto en órbita, lleva a la implementación de técnicas de cosimulación como *hardware in the loop* para la validación y evaluación de los diversos elementos que integran un esquema de control de orientación para un satélite, tales como sensores de navegación, actuadores y dispositivos unidades de procesamiento y cómputo de datos.

En México, el incipiente desarrollo de ciertas áreas tecnológicas tal como la espacial, si bien han tenido algunos aciertos y avances significativos en campos bien definidos como por ejemplo en el campo de cohetes que durante los años 60 y 70s alcanzó grandes avances con el desarrollo de estos que alcanzaron grandes alturas sin destruirse o más recientemente en la integración de los satélites UNAMSAT A y B, que aunque fueron sólo ensamblados en México con tecnología extranjera, marcaron un referente en cuanto a la experiencia ganada por los grupos participantes de aquél entonces.

Sin embargo, el vertiginoso avance de la ciencia y tecnología espacial demanda la formación de recursos humanos calificados en áreas tan complejas del desarrollo de un vehículo espacial como lo es el subsistema de control de orientación. El mercado de consumo a nivel internacional, particularmente para misiones de percepción remota y comunicaciones directivas, demanda cada día un mayor nivel de estabilidad y capacidad de apuntamiento de las plataformas satelitales.

En el Grupo de Desarrollo de Sistemas Aeroespaciales del Instituto de Ingeniería, UNAM (GDSA-UNAM), desde 2009 se ha venido trabajando intensamente de forma multi-institucional y como parte de un trabajo de investigación doctoral, en la integración de un sistema que permita la validación y evaluación en tierra de algoritmos y elementos (en hardware y software) de esquemas de control de orientación en tres ejes para satélites pequeños, con la ventaja de integrar toda la lógica de control en una arquitectura de cómputo dedicada en un dispositivo reconfigurable FPGA.

Este trabajo de tesis es una vertiente de desarrollo de aquél trabajo doctoral, y debe verse como una de las primeras iniciativas reales en México de desarrollo de tecnología satelital que muestra en primera aproximación, el comportamiento particular de un sistema ADCS, las maniobras de estabilización y apuntamiento hacia objetivos específicos.

Los alcances de la temática abordada en esta tesis al momento son vastos y excederían el esquema tradicional de un trabajo de titulación. Sin embargo, como primera aproximación en la integración de un sistema de simulación y validación en tierra, se muestran resultados que sin bien, aún están sujetos a una serie de ajustes y cambios, representan un avance importante en el desarrollo de tecnología de estabilización y control de apuntamiento en México, máxime donde actualmente a nivel mundial sólo países tan selectos como la India y algunos otros de la Unión Europea han tenido gran injerencia, y que invierten varios miles de dólares en su investigación y desarrollo cada año.

El contenido de la tesis está estructurado de la siguiente forma:

Capitulo 1: Se abordan las generalidades de las plataformas ADCS o de simulación satelital, sus principales características de simulación de vuelo, acoplamiento, movimiento, instrumentación, entre otras.

Capítulo 2: Trata de las generalidades las plataformas de desarrollo FPGA, sobre su arquitectura, su configuración, algunos de sus componentes y diferencias de fabricación, así como de los fabricantes de algunas plataformas de desarrollo que las contienen.

Capítulo 3: Se exponen los elementos básicos para el estudio y análisis del modelo matemático de la plataforma de simulación para la validación de esquemas de control, y también, se plantea el desarrollo de la ecuación de movimiento de la MSA con base en los elementos de estudio expuestos, esto es la dinámica del sistema.

***Capítulo 4**:* Se propone una discusión para abordar las diferentes estrategias y propuestas de implementación e integración del sistema de simulación para la validación de esquemas de control y orientación, y se resumen en dos vertientes: la primera propone integrar toda la arquitectura de cómputo sobre un solo dispositivo FPGA; y la segunda propone utilizar las técnicas HIL para simular el coprocesamiento con ayuda de una computadora externa, permitiendo una ejecución del sistema como si estuviese totalmente integrado a bordo.

***Capítulo 5**:* Se abordan los detalles del primer segmento del sistema que será implementado en dos bloques principales, con base en las técnicas HIL, el cual comprende el módulo de adquisición de datos (AD) y el control de actuadores, los cuáles han sido embebidos en la plataforma FPGA, se explican las configuraciones realizadas, los diagramas de flujo de los algoritmos implementados, y se expone una revisión de todo el segmentos y sus elementos que lo componen.

***Capítulo 6**:* Se explican los detalles del segundo segmento del sistema, e cual consiste en la implementación de los algoritmos de determinación y corrección de la orientación (EKG y TRAID), y la ley de control que con base en ella podrán determinarse los ciclos de trabajo de las ruedas inerciales y el sentido de giro de las mismas.

***Capítulo 7**:* Una vez realizada la verificación de los componentes y que hayan sido validados en integrados, en este capítulo se muestran los resultados experimentales de las pruebas realizadas y sus resultados. Se discuten de forma breve estos resultados obtenidos a partir de las gráficas obtenidas del comportamiento del sistema completo en funcionamiento.

***Capítulo 8**:* Finalmente, en este capítulo, se realiza una discusión de los resultados obtenidos y de los esperados, de los alcances de esta primera aproximación, así como de las recomendaciones que se hacen para la continuación del trabajo, así como un panorama de lo que sería el trabajo futuro.

Capítulo 1

Introducción a las plataformas experimentales para satélites y su instrumentación

Este capítulo aborda una introducción acerca de los proyectos de desarrollo de plataformas experimentales de bajo costo para validación de esquemas de control de orientación para satélites pequeños en diferentes universidades y centros de investigación alrededor del mundo. Se describen diversos aspectos muy generales de estas plataformas y de su instrumentación, así como algunas de las características de los módulos que las componen y funciones adicionales al control y la estabilización. El objetivo del capítulo es presentar un panorama general sobre el desarrollo de estos sistemas, de tal forma, que pueda hacerse una relación de los proyectos y sus logros con base en el trabajo continuo de las diversas instituciones, para que de esa forma este trabajo tenga un factor de comparación en esta primera aproximación del sistema que se pretende desarrollar, y además, tener un respaldo de autoridad de los esquemas de trabajo para realizar la implementación de sistemas, básicamente de control.

1.1 Introducción

Actualmente el desarrollo de sistema de simulación con técnicas de aceleración de implementación y obtención de resultados, verbigracia, *hardware in the loop*, ha permitido a diversas instituciones científicas, universitarias y de otros rubros elaborar y validar esquemas de control de diversa complejidad con buenos resultados en un corto plazo. La continuidad del trabajo y las abundantes herramientas que ofrecen muchos fabricantes de dispositivos complejos de diversas capacidades e innumerables aplicaciones por un costo relativamente bajo, ha puesto al alcance la tecnología necesaria para integrar sistemas aeroespaciales, promoviendo mayor competencia en este campo, y permitiendo que naciones aún sin infraestructura para lanzamientos reales, puedan realizar experimentos con condiciones muy similares a las del espacio donde se valide el funcionamiento de estos sistemas. Permitiendo, también, que diversas naciones cooperen en proyectos conjuntos con la finalidad de probar de forma real lo que desarrollaron y validaron en tierra.

En este capítulo se exponen proyectos que son referentes en este campo de desarrollo de simuladores aeroespaciales, enfatizando que sólo son para ejemplificar el crecimiento del estudio y desarrollo para la integración de estos sistemas.

1.2 Plataformas experimentales para aplicaciones espaciales

Las plataformas experimentales para validación de esquemas de control de orientación son ampliamente utilizadas en la industria aeroespacial, así como en universidades y centros de investigación. Muchas de ellas tuvieron su origen en Estados Unidos, donde se construyó el primer simulador de control de orientación en el Centro de Investigación Ames de la NASA en 1959.

Estas plataformas experimentales permiten a los investigadores probar en Tierra —en un ambiente que simula con gran aproximación las condiciones espaciales, como la ausencia de fricción— los algoritmos de un Sistema de Control y Determinación de Orientación para un satélites (*Attitude Determination Control System*, ADCS); de igual forma facilitan la simulación y análisis de la dinámica del satélite por medio de bancos de pruebas que permiten el planteamiento de mejores estrategias de control [1].

Diversos tipos y formas de las plataformas se han desarrollado a lo largo de la historia del diseño de los sistemas aeroespaciales, con ciertas características que los han convertido en

un referente en su campo. Por ello, a continuación se describirán los tipos o formas más conocidas de las plataformas utilizadas en el desarrollo de sistemas aeroespaciales, y donde normalmente residen la mayor parte de elementos, considerados a bordo del sistema. En ellas aparece un sistema de referencia que corresponde a la orientación de un cuerpo en tres ejes, donde se hace uso de los denominados ángulos de navegación, cada uno de los cuales representa maniobras consecutivas en las direcciones de los respectivos ejes de rotación del marco de referencia **X**, **Y** y **Z**. Cada uno de éstos ángulos se denomina: ***yaw, pitch*** y ***roll,*** o guiñada, cabeceo y alabeo como se traducen al español, los cuales son movimientos de rotación respecto a cada uno de los ejes **X**, **Y, Z**, respectivamente, y serán explicados a detalle en otro capítulo.

Los diseños o formas básicas los que predominan en el campo de las plataformas experimentales, se han clasificado de la siguiente forma [2]:

a) **Plataforma de Sobremesa**. En esta plataforma los componentes deben estar distribuidos en la parte superior de la plataforma para simplificar el análisis de movimiento, según se muestra en la Figura 1.1.

Figura 1.1 Plataforma de Sobremesa.

b) **Plataforma de Sombrilla**. En este diseño se mantiene el centro de masa del sistema muy cerca del centro de rotación, para suspender cuidadosamente los componentes debajo de la plataforma, como se observa en la Figura 1.2.

Figura 1.2 Plataforma de Sombrilla.

c) **Plataforma de Pesa (Mancuerna).** Este diseño reduce la interferencia de la estructura colocando la carga útil del simulador lo más lejos posible del centro de rotación (Figura 1.3).

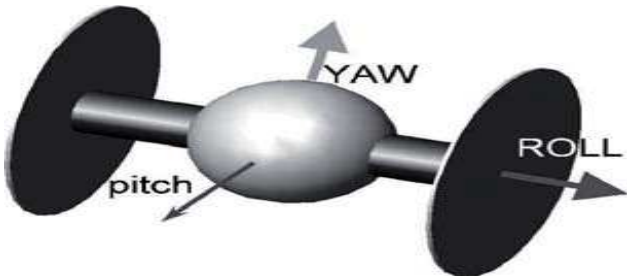


Figura 1.3 plataforma de Pesa (Mancuerna).

Con base en las características de cada una de las plataformas, los diseñadores del sistema de control eligen cuál es aquella que cumple los requerimientos de los objetivos que se persiguen y la complejidad del sistema de orientación y estabilización.

Enseguida se presentan algunos de los sistemas aeroespaciales que implementan maniobras de control en tres ejes, además de otras funcionalidades, que han sido desarrollados por universidades y centros de investigación, y muestran la integración de tres elementos comunes en ellos: sensores, dispositivos de procesamiento y actuadores. De igual forma, se verán las formas de las plataformas y en algunos casos habrá diferentes formas con diversos fines y con mayor complejidad para su integración, es decir, un mismo sistema está integrado a su vez por más de una plataforma.

1.2.1 Lanzador Automático Orbital de Picosatélite (*Orbital Picosatellite Automated Launcher*, OPAL)

El sistema de control de posición OPAL (Figura 1.4) fue el primer simulador con una plataforma de forma esférica, o de sobremesa, sin clasificación y sin derechos de propiedad. Fue desarrollada en Stanford en 1975. En 1996 fue modificada para usar dos pares de bobinas magnéticas y un magnetómetro de 3 ejes. Las pruebas de validación se ejecutaron auxiliadas de un modelo en computadora [3].

Figura 1.4 Modelo virtual del OPAL.

1.2.2 Simulador del Sistema de Control de Posición para un Vehículo Espacial Distribuido del Tecnológico de Virginia (*Whorl I, Whorl II*),*Virginia Tech Distributed Spacecraft Attitude Control SystemSimulator* (DSACSS)

Este sistema de simulación consta de 2 tipos de plataforma, independientes entre sí, que integradas constituyen un simulador para la validación de un sistema de control de orientación de formación en vuelo. La primera de ellas es un modelo basado en una plataforma de sobremesa y la segunda está basada en el modelo de mancuerna. Son nombradas Whorl I (Figura 1.5) y Whorl II (Figura 1.6), respectivamente. Entre sus capacidades destacan el trabajo con técnicas de almacenamiento de energía y opciones de control que incluyen control de posición acoplado y compensación no lineal de un sistema *under-actuated*, es decir, es un sistema de dos grados de libertad. Además, el software principal está escrito con base al modelo de programación orientado a objetos [4].

Figura 1.5 Instrumentación de la Whorl I y Whorl II, de derecha a izquierda.

1.2.3 Simulador Dinámico para la Orientación de un Vehículo Espacial del Instituto Politécnico del Estado de California. *Cal Poly Spacecraft Attitude Dynamic Simulator* (CP/SADS)

Este simulador se compone de cuatro dispositivos intercambiables de ruedas inerciales, montados sobre una plataforma de sobremesa semiesférica (Figura 1.7). Puede realizar maniobras de rotación de 360° en el eje **Z** y ± 30° en el eje **X**. En el eje **Y** con puede alcanzar una velocidad de 0.1 *rad/s* y 0.1 *rad/s²* de aceleración angular [5]. Además cuenta un módulo de comunicaciones inalámbricas a través de una plataforma de desarrollo comercial (*PC/104 on-board computer*), la cual resuelve sin problemas las necesidades de adquisición y transmisión de señales de los actuadores y sensores con la unidad de procesamiento. En las Figuras 1.7 y 1.8 se muestran otras características.

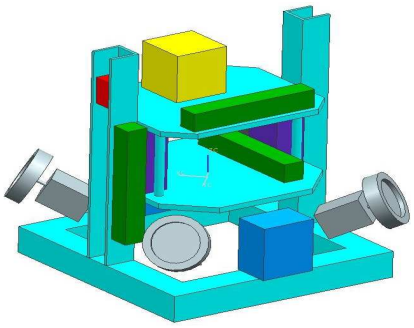


Figura 1.7 Modelo virtual de CP/SAD.

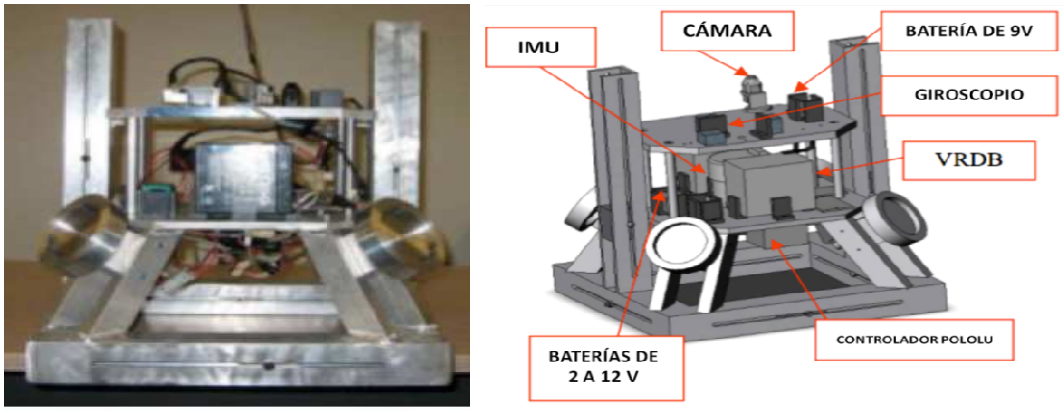


Figura 1.8 Vista frontal de CP/SADS y su instrumentación.

1.2.4 Plataforma Experimental Con 5 Grados de Libertad Para Vehículos Espaciales de Encuentro y Acoplamiento del Instituto de Tecnología de Georgia. *A 5-dof Experimental Platform for Autonomous Spacecraft Rendezvous and Docking of Georgia Institute of Technology*

El Instituto de Tecnología de Georgia desarrolló un sistema experimental a partir de un sistema anterior el cual tenía 3 grados de libertad (*3-dof Integrated Attitude Control System*, IACS), agregando dos grados de libertad adicionales para los movimientos de encuentro y acoplamiento; esto significa que tiene dos grados de movimiento de traslación a lo largo de los ejes **X** y **Y**, más tres grados de movimiento de rotación sobre los tres ejes correspondientes a **X**, **Y** y **Z**) [6]. También utiliza dos tipos de plataforma para facilitar operaciones sin fricción: se integra por tres sistemas lineales basados en plataformas de sobremesa y una más semiesférica, las primeras hacen posible un movimiento de traslación casi completo y sin fricción del sistema sobre un piso plano de epoxi, mientras la plataforma semiesférica es utilizada para habilitar el movimiento rotacional sin fricción del vehículo espacial con respecto a un pedestal de soporte. En las siguientes figusras se muestra el sistema integrado, así como algunas características y partes importantes del mismo.

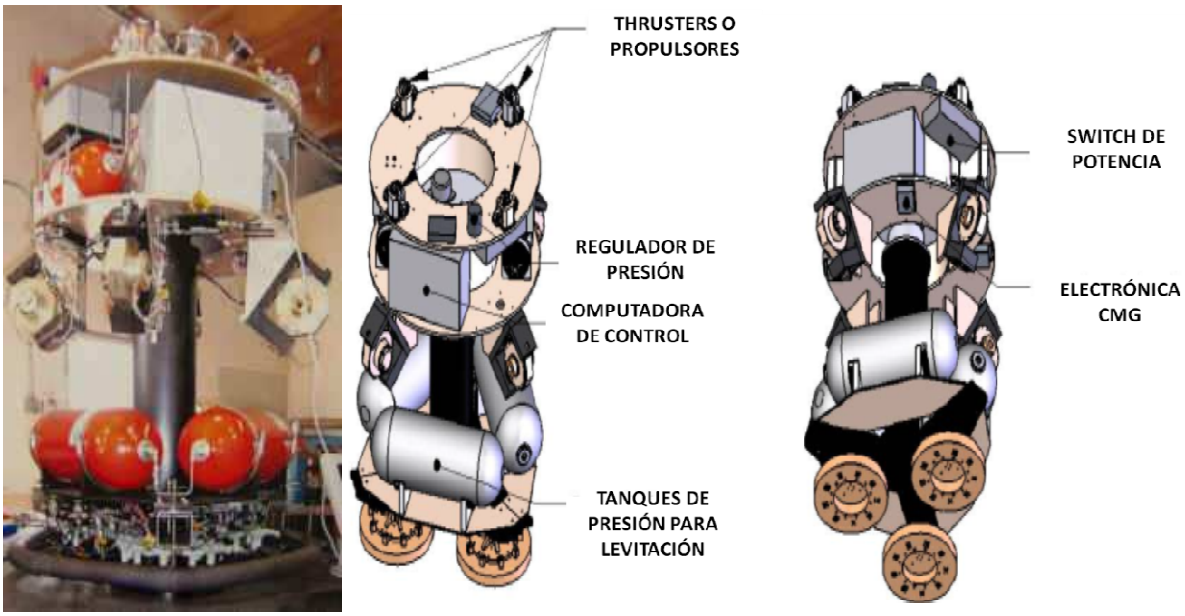


Figura 1.9 Ilustraciones del modelo del *5-dof Spacecraft Simulator for Autonomous Rendezvous and Docking (SSARDN).*

1.3 Instrumentación: sensores, actuadores y procesadores

En repetidas ocasiones la realización de una medición requiere la intervención de varios instrumentos, algunos de ellos generan estímulos sobre el dispositivo que pretende medir, mientras que otros están dispuestos para recoger las respuestas a estos estímulos. El conjunto de instrumentos que permiten la realización de las mediciones se denomina sistema de instrumentación. Todo sistema de instrumentación está integrado de una serie de instrumentos o dispositivos (electrónicos, eléctricos y/o mecánicos), un sistema de interconexión de estos instrumentos o dispositivos, y un controlador inteligente que administra el funcionamiento de todo el sistema y genera los comandos de control (plataforma de cómputo) para que una medición se realice correctamente.

La instrumentación de un sistema de simulación para validación de control es una parte fundamental y común en dichos sistemas, y hay una gran cantidad de dispositivos que componen la instrumentación, no obstante, en esta sección sólo se abordaran tres categorías muy comunes: los sensores, actuadores y procesadores. En el caso específico de dichos sistemas, es posible identificar los dispositivos que conforman la instrumentación, como son los sensores de navegación, el módulo de transducción (externo o interno a la plataforma de cómputo a bordo), el dispositivo de procesamiento (basado en un dispositivo de arquitectura fija o reconfigurable) y los actuadores (activos o pasivos, basados en ruedas inerciales, bobinas de par magnético u otras técnicas). En función del esquema de trabajo que se adopte para la implementación del sistema, es posible apoyarse en técnicas experimentales que permitan acelerar la realización de pruebas y la obtención de resultados, ya sea utilizando una unidad de procesamiento a bordo, implementado los algoritmos de control a utilizar, o bien, utilizando esquemas de cosimulación, los cuales permiten la distribución de los diferentes bloques del sistema en dos secciones de cómputo y procesamiento diferentes: como un dispositivo lógico programable o reconfigurable a bordo y una computadora personal (PC) externa donde habitualmente se ejecutan los algoritmos de control.

Existen técnicas de cosimulación denominadas *hardware in the loop* (HIL), que son útiles para la evaluación y definición de la instrumentación (sensores, elementos de cómputo y actuadores), donde gran parte de ella puede ser operada casi en tiempo real, como si el sistema estuviera ya integrado en su totalidad. De esa forma, las técnicas HIL, además de permitir la aceleración en la integración de sistemas así como en la obtención de resultados, hacen posible la visualización del sistema como si estuviera funcionando completamente.

En los siguientes apartados se mostrarán las características técnicas y cualitativas de algunos elementos de instrumentación como los sensores de navegación, procesadores y actuadores, principalmente.

1.3.1 Sensores de navegación

Los sensores de navegación son una categoría de instrumentación muy amplia, y de gran variedad en el mercado, ya que de acuerdo a su nivel de sofisticación aumentan su costo y mejoran sus características técnicas. Estos dispositivos se encargan de generar datos suficientes para obtener la posición del vehículo espacial, a partir de determinaciones directas de la velocidad y aceleración, o bien, por medio de la determinación indirecta utilizando para ello el registro de variables a su alrededor, como el campo magnético o la posición del sol y de las estrellas, según el tipo de dispositivo. Pueden adquirirse como unidades compuestas por diversos sensores usando combinaciones de giróscopos, acelerómetros, magnetómetros u otros, integrados en una sola unidad denominada ***Inertial Measurement Unit*** (IMU), Unidad de Mediciones Inerciales. Otros sensores utilizados son los compases y los sistemas de posicionamiento global GPS (*Global Position System*).

A continuación, se presentan algunos modelos de sensores de navegación de bajo costo y de un desempeño medio, los cuales, debido a que muchos desarrollos experimentales en el campo aeroespacial no cuentan con una inversión muy grande para adquirir sensores de mejores características, en muchos de estos proyectos se recurre a la utilización de técnicas filtros digitales e implementación de algoritmos de estimación y corrección que puedan reducir ruido y suavizar y corregir las señales de los dispositivos, en específico, de las lecturas obtenidas, y de esta manera mejorar la eficiencia del sistema de control sin tener que invertir en dispositivos muy costosos para obtener buenos resultados en los experimentos.

Crossbow IMU

Se trata de un una unidad de mediciones inerciales, la cual integra un giróscopo de estado sólido (Figura 1.10) de la compañía *Crossbow Technologies* de 6 grados de libertad (6DOF), destinado a aplicaciones de vuelo tales como control UAV (***Unmaned Aerial Vehicle***), vehículos no tripulados, ***avionics*** y plataformas de estabilización y control [7].

Está compuesto por giróscopos de bajo costo de producción, lo cual repercute en un desempeño relativamente bajo. Los datos son transmitidos utilizando el protocolo de comunicaciones seriales RS-232 y se trata de un dispositivo de bajo consumo de potencia, con una corriente alrededor de 200 [mA].

Figura 1.10 Unidad de medición inercial de Crossbow Technologies.

Acelerómetro SCA3000-E01

Es un acelerómetro de 3 ejes de muy bajo consumo de potencia con interfaz SPI (***Serial Peripheral Interface***) para la transmisión de datos (Figura 1.11). Es capaz de almacenar hasta 64 muestras para cada uno de los 3 ejes en la salida de datos de aceleración [8]. Es un sensor de bajo costo y presenta gran velocidad en la transmisión de adquisición de datos, así como un grado de fiabilidad y sensibilidad en sus mediciones, pues la hoja de datos especifica un margen de error en la medición de ±3 grados decimales.

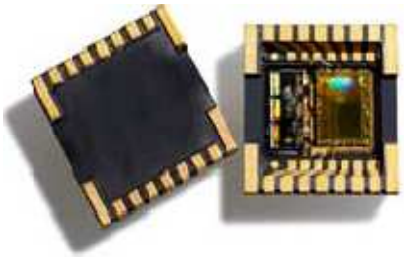


Figura 1.11 Acelerómetro de 3 ejes de bajo consumo de potencia.

Magnetómetro HMC6052

Es un sensor *magneto-resistivo*, utilizado en aplicaciones de conteo y determinación del sentido de rotación, en 2 ejes, más amplificadores con soporte para medición de campo magnético y medición del compás [9]. Es un sensor de estado sólido que no requiere de otros componentes para la obtención de señales, la hoja de datos indica un margen de medición de campo magnético de ±2 [Gauss]. Cabe mencionar que este sensor requiere que su consumo de potencia está en función con la magnitud del campo magnético que mide, por lo que puede elevarse si este es muy bajo.

Figura 1.12 Magnetómetro analógico de dos ejes.

1.3.2 Procesadores

Los procesadores son los cerebros de los sistemas de control para determinar la orientación. Son dispositivos en los que se ejecuta el software o firmware necesario para interaccionar, manipular y procesar los datos de los sensores y devolver las señales de control necesarias para llevar a cabo las diferentes maniobras de control de orientación del sistema de simulación aeroespacial. De acuerdo a sus características: arquitectura, velocidad, conjunto de instrucciones, escala de integración, etcétera, pueden llevar a cabo muchos procesos concurrentes, tener la capacidad para manejar la comunicación con diversas interfaces o puertos, u otras conexiones, todo esto dependerá del rendimiento y robustez del dispositivo. El objetivo de este apartado es exponer diversos dispositivos populares en el desarrollo de sistemas de control de orientación para vehículos aeroespaciales, así como resaltar las diferencias conceptuales que permitirán discernir entre lo que es un procesador y una unidad de procesamiento.

Las tendencias actuales apuntan al uso dominante del procesamiento en un procesador tradicional (***hard processor***), aunque el procesamiento en una unidad de procesamiento

(***soft processor***) también tiene un área amplia de aplicaciones y su uso se va extendiendo en diversos campos de investigación, ambos conceptos tienen diferencias muy marcadas. En el primero se trata de una arquitectura fija, implementada por el fabricante y no modificable; se tienen áreas específicas para memoria de datos, memoria de programación y registros. En el segundo, se tiene un esquema llamado regularmente *súper función*, y es una plataforma de elementos lógicos reconfigurables, en donde puede incluirse una arquitectura personalizada generada y descargable a través de un software desde una computadora.

Microprocesador *dsPIC33FJ256GP506*

Se trata de un *hard processor* DSPIC de 16 bits de alto desempeño de arquitectura *Harvard* modificada (Figura 1.13), que trabaja a una velocidad de 20 millones de instrucciones por segundo, cuenta también con memoria RAM estática y un conjunto base de 83 instrucciones. Además, cuenta con 2 unidades UART (***Universal Asynchronous Receiver Transmitter)*** para el flujo datos [10].

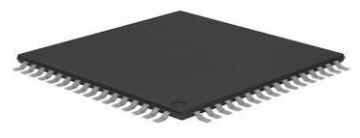


Figura 1.13 Encapsulado del DSPIC de Microchip.

Procesador Intel Pentium M

El procesador Intel Pentium M (Figura 1.14) utiliza una microarquitectura de alto desempeño y bajo consumo de energía, de uso muy amplio en aplicaciones de comercio y también de automatización, con soporte para *chipsets* Intel E7501 e Intel 855GME. Se trata de un procesador de un solo núcleo desarrollado inicialmente para computadoras portátiles, capaz de alcanzar velocidades de procesamiento de hasta 1.7 GHz, e incluye memoria caché de nivel 2 de 1 o 2 MB. Tiene un consumo de hasta 1.484 [V], este procesador se incluye en la arquitectura del sistema descrito anteriormente de *Instituto Politécnico de California* (*Cal Poly*) [11].

Figura 14 Procesador Intel Pentium M.

MicroBlaze Soft Processor Core

MicroBlaze es un *soft processor* de 32 bits, de arquitectura *Harvard* RISC (*Reduced Instruction Set Computer*) optimizado para aplicaciones embebidas [12]. Es la emulación de un *hard processor* en un núcleo embebido sobre una arquitectura de hardware reconfigurable. MicroBlaze ofrece las características de un procesador de arquitectura fija pero tiene la capacidad de ser reconfigurado parcial o totalmente (figura 1.15), además de que se pueden agregar nuevos núcleos o funciones a la arquitectura base. Tiene un costo relativamente bajo con respecto a los *hard processors* en su categoría.

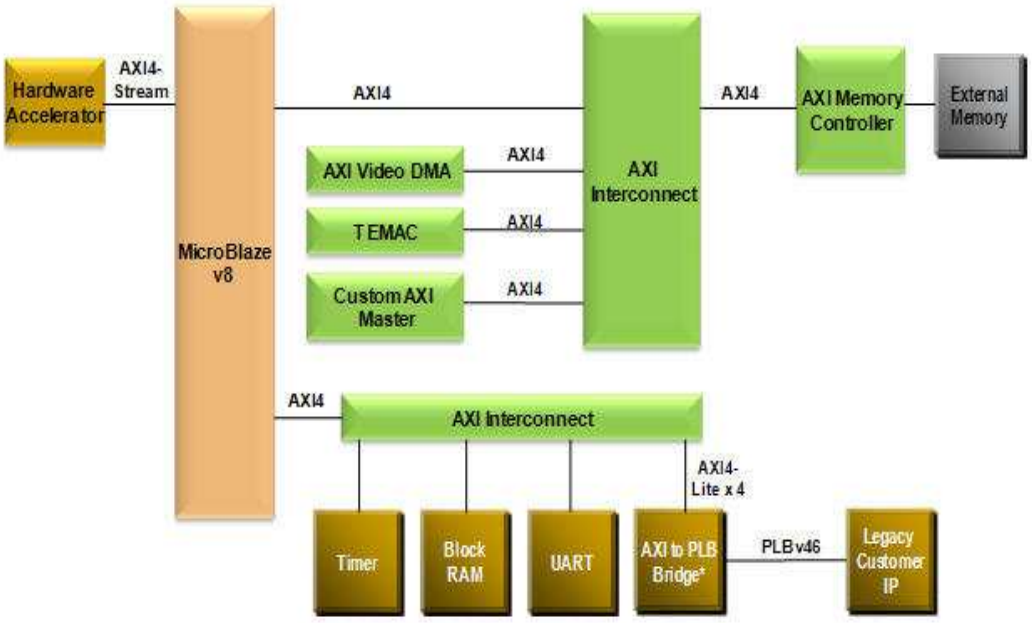


Figura 1.15 Diagrama de arquitectura del MicroBlaze.

LEON Soft Processor Core

El *soft processor* LEON es una implementación de código abierto en VHDL (*Verilog Hardware Description Language*), de arquitectura SPARC V8 (*Version 8 of the Scalable Processor ARChitecture*), el cual es un procesador de 32 bits desarrollado para aplicaciones embebidas (Figura 1.16) [13], de capacidad reconfigurable también, y compatible con algunas tarjetas de desarrollo comerciales como VIRTEX. La versión LEON 2F desarrollada por Jiri Gaisler de la Agencia Espacial Europea (ESA) es un núcleo basado en el estándar de un procesador de 32 bits de la IEEE-1754 (SPARC V8) y sólo esta licenciado para programas que estén dentro del marco de investigación de la ESA.

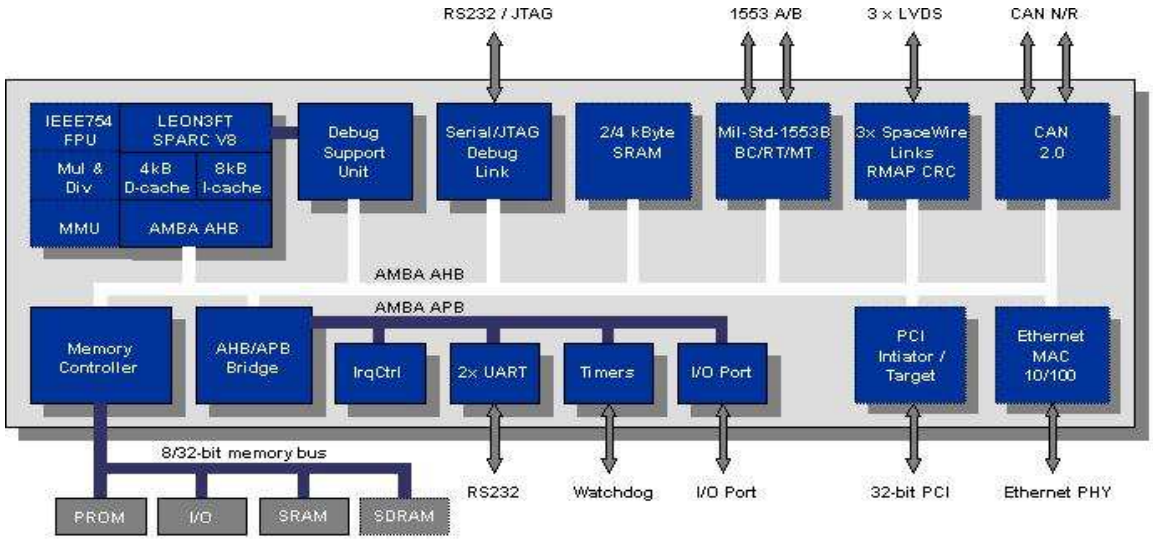


Figura 1.16 Diagrama de arquitectura del LEON3FT-RTAX Fault-tolerant Processor.

1.3.3 Actuadores

Los actuadores son los dispositivos que hacen posibles las maniobras en los sistemas de control, pues son los encargados de interpretar las órdenes de algún controlador para realizar las maniobras de estabilización y orientación de la plataforma; esto en el caso de sistemas aeroespaciales. La definición de *actuador* es muy amplia (pues hay una gran variedad de tipos), pero en este apartado se resume como: un dispositivo capaz de

transformar la energía eléctrica, mecánica o hidráulica en la activación de un proceso con la finalidad de generar un efecto sobre un proceso automatizado [14]. Además, esta sección sólo se enfocara en aquellos actuadores más utilizados por su costo bajo en los sistemas de control de orientación y estabilización como son las ruedas inerciales y las bobinas magnéticas.

Ruedas inerciales

Las ruedas inerciales son dispositivos utilizados para realizar maniobras de control, y se apoyan con el uso de motores de corriente directa que aprovechan la energía eléctrica para hacer girar las ruedas y producir un efecto en todo el sistema. Las instrucciones que reciben se generan en un controlador o procesador, y son calculadas por un algoritmo de control, que actúa en consecuencia al movimiento de la plataforma. Un motor está fijo a la plataforma y el eje de motor está unido a un volante. Cuando un voltaje de control es aplicado al motor, este genera un par haciendo girar la rueda en una dirección y la plataforma en otra. De ahí el término de rueda inercial. Dado que el torque es interno al sistema aeroespacial, la rueda inercial no puede cambiar el momento angular inercial total del sistema. Si el torque externo a la plataforma es periódico (con una frecuencia suficientemente alta) con respecto al sistema de referencia inercial, entonces la rueda inercial puede controlar completamente la plataforma [15].

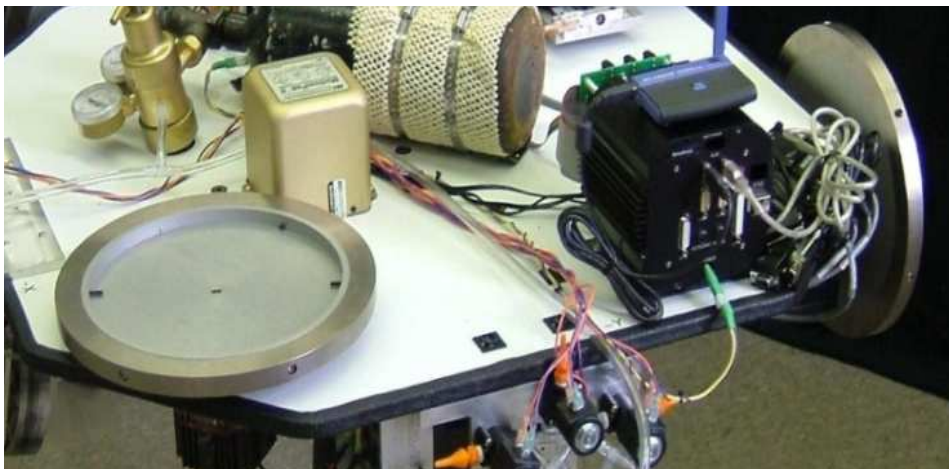


Figura 1.17 Ruedas inerciales de la plataforma Whorl I.

Bobinas de par magnético

Los sistemas de control con bobinas magnéticas pueden ser utilizados eficazmente para llevar a cabo maniobras de apuntamiento, estabilización y orientación; son de bajo peso, no tienen partes móviles, y son relativamente simples. Además, la torsión magnética es muy atractiva para aplicaciones espaciales, sin embargo, la precisión final de apuntamiento se limita a ± 3 ° de arco y la fuerza del par de control depende de la magnitud de los tres componentes vectoriales del campo magnético [16].

Figura 1.18 Bobinas magnéticas de una plataforma ADCS desarrolladas en el IG-UNAM.

El diseño y construcción de sistemas experimentales de simulación para validación de esquemas de control y apuntamiento es imprescindible para el avance del estudio, desarrollo, y aplicaciones en el campo satelital. Por medio de ellas se puede validar el diseño de un sistema que persigue la estabilización y control con base en sus características físicas, como la complejidad del movimiento y forma de la plataforma, y la implementación de la lógica para su funcionamiento, como los esquemas de validación de control y filtros de corrección. Además, puede tener funciones adicionales, como comunicaciones con otros sistemas, acoplamiento, captura y análisis de imágenes, entre otras. La experimentación en Tierra puede lograr condiciones muy cercanas a las que se tienen en el espacio considerando no sólo condiciones y fuerzas presentes al realizar el modelo matemático, sino también incluyendo instrumentación adecuada para lograr tales objetivos, cuyos componentes pueden ser adquiridos comercialmente, o bien, ser de diseño y fabricación propios si las ventajas de esto se imponen en el cotejamiento.

También, a pesar de incluir elementos de instrumentación e desempeño medio, se pueden integrar esquemas de corrección y filtrado para obtener un diseño bastante eficiente y efectivo con elementos de menor costo. Por otro lado, se puede acelerar la obtención de resultados y validación de algoritmos, mediante técnicas de cosimulación, como HIL.

Se abordaron de forma breve los elementos de instrumentación, como los sensores de navegación, que realizan las mediciones necesarias para determinar la posición del vehículo, y que normalmente se incluyen diversos tipos en una unidad de mediciones inerciales (IMU) compuestas por magnetómetros, acelerómetros y giróscopos. Se presentaron los clasificación de procesadores en: de arquitectura rígida, *hard processor*, veloces, no reconfigurables en vuelo y uso muy extendido aún en aplicaciones aeroespaciales; y por otro lado, los *soft processor*, definidos en unidades de procesamiento, en las cuales se agregan como súper funciones, y están basados en arquitecturas rígidas, pero pueden ser reconfigurables en vuelo, y de uso cada vez más extendido.

Finalmente, se abordaron los actuadores, donde sólo se presentaron las ruedas inerciales y las bobinas de par magnético, de las cuales, su finalidad es dotar al sistema de la capacidad para realizar diversas maniobras que cumplan los fines del diseño del sistema desarrollado, como la estabilización, apuntamiento, acoplamiento, etc. En el siguiente capítulo se abordan las generalidades sobre FPGA, que es una unidad de procesamiento reconfigurable, que e incluye en el desarrollo de esta tesis como dispositivo principal.

Capítulo 2

Generalidades sobre FPGAs

Desde los teléfonos inteligentes hasta equipos médicos, desde sistemas de comunicaciones como las microondas hasta los sistemas vehiculares de frenado antibloqueo, los modernos sistemas embebidos han provocado el desarrollo de novedosas plataformas computacionales que han llegado a convertirse en parte importante de las sociedades actuales. En este sentido, la industria emplea una gama amplia de herramientas y tecnología para la integración de los sistemas computacionales incluyendo componentes en software y hardware. Uno de esos componentes, la FPGA, se ha posicionado de una forma importante en el mercado, conservando una tendencia creciente en su consumo. De forma práctica, una FPGA puede ser vista como un pizarrón en blanco, sobre la cual puede ser configurado cualquier circuito digital. Una FPGA provee de una plataforma de hardware programable al desarrollador de sistemas embebidos.

En este capítulo se abordan aspectos generales sobre los dispositivos programables FPGAs tales como la arquitectura del dispositivo, componentes, los modelos de programación, software de desarrollo para aplicaciones específicas y de propósito general; así como algunas herramientas de programación, validación y *depuración* de proyectos, etc.

2.1 Introducción

Los dispositivos FPGA han tenido su origen a mediados de la década de los 80 cuando Xilinx incorporó un producto de dos tecnologías diferentes: los dispositivos lógicos programables (PLD) y los circuitos integrados de aplicación específica (ASIC), dando como resultado los FPGA (*Field Programmable Gate Array*), que básicamente son dispositivos que no son programados por el fabricante, como puede ser el caso de un microprocesador, sino por un usuario final. Actualmente han sido utilizados para innumerables aplicaciones en diversos campos, a pesar de que originalmente su uso era muy reducido y se enfocaba más a las telecomunicaciones y papeles secundarios, ahora se incluyen en aplicaciones tanto científicas como tecnológicas de uso residencial, comercial e industrial, y esto ha sido por su gran flexibilidad y reconfiguración incluso cuando el sistema desarrollado ya se encuentra en su forma embebida. En los siguiente puntos a tratar de este capítulo se describen las características generales de las FPGA, desde cómo está compuesta la arquitectura de estos dispositivos, pasando por las herramientas disponibles para su programación, hasta las plataformas que las incluyen donde se integran con más elementos y dispositivos, como los puertos de comunicaciones, y que funcionan como plataformas de desarrollo para aplicaciones personalizadas.

2.2 Arquitectura de un dispositivo FPGA

Un FPGA es un dispositivo lógico que contiene una matriz bidimensional de celdas lógicas genéricas e interruptores programables. Una celda lógica puede ser configurada para realizar una función sencilla, y un interruptor programable puede ser personalizado para proveer interconexiones entre celdas lógicas [17]. Estas características le permiten a un dispositivo programable FPGA ser reconfigurado a través de la programación por un usuario final e incluso cuando ya se encuentra embebido. Actualmente los dispositivos comerciales FPGA están basados en las tecnologías de SRAM (*Static Random Access Memory*) o *antifuse* [18] que son las tecnologías más populares pero no las únicas, también existen ROM, EPROM EEPROM, *fusible link*, y FLASH, no obstante, para los fines expositivos del capítulo, sólo se describirán las dos primeras. Los bloques SRAM se utilizan para controlar los interruptores de los transistores de paso de los nodos de compuerta y para controlar las líneas de selección de los multiplexores que manejan las entradas a los bloques lógicos [19].

En la Figura 2.1 se muestra su arquitectura de bloques, así como algunos de sus componentes.

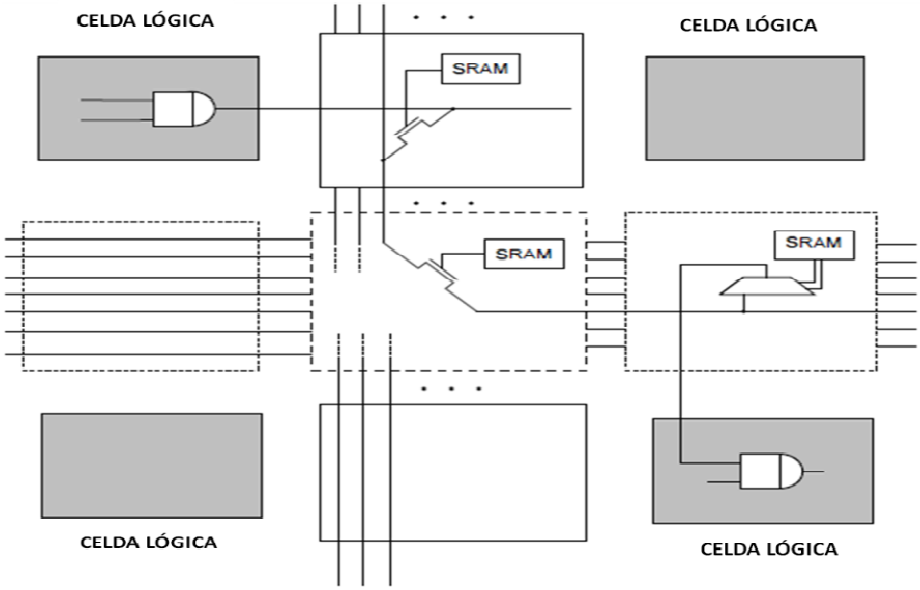


Figura 2.1 Bloques SRAM de control.

Por otro lado está el tipo de interruptor programable *antifuse*, y se trata de un circuito abierto que asume baja resistencia sólo cuando es programado. Los antifuses son construidos usando tecnología CMOS, lo que los hace adecuados para los dispositivos programables FPGA. Suelen ser colocados en dos líneas de interconexión, y se componen de tres capas: dos capas conductoras, una inferior y una superior, y un aislante intermedio [20] (Figura 2.2).

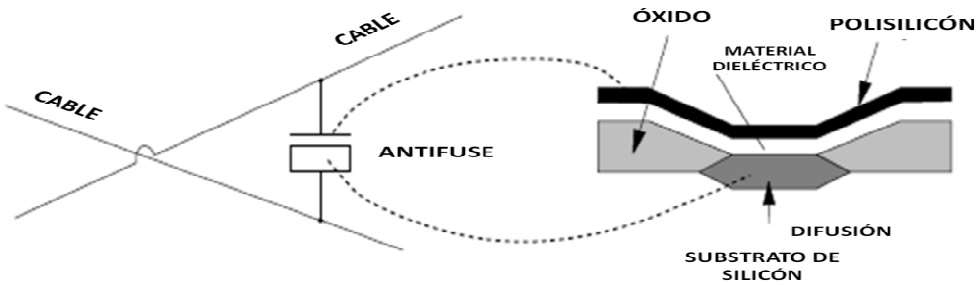


Figura 2.2 Estructura antifuse de una FPGA de la compañía Microsemi.

Los dispositivos reconfigurables cada vez tienen un mayor uso en aplicaciones industriales, de investigación y de diseño y crece su escala de integración y procesamiento de forma constante, empero, cabe mencionar que hay otros dispositivos reconfigurables aparte de las FPGA, los CPLD (*Complex Programmable Logic Device*), que se caracterizan por utilizar bloques lógicos programables similares a los de los dispositivos FPGA pero en una arquitectura más rígida y menos compleja. En esta tesis se empleará para las validaciones experimentales una plataforma comercial de desarrollo que integra una FPGA de Xilinx.

2.2.1 La familia de FPGA SPARTAN.

Xilinx es una compañía que fabrica FPGAs y sus productos se basan en interruptores programables con bloques SRAM. Cuenta con diversas categorías de FPGAs, las cuales ha dividido en *familias* o *series*, la primera de ellas fue la *XC2000 series*, lanzada en 1985 [21]. La familia de FPGA SPARTAN fue una de las primeras series de Xilinx de producción masiva y bajo costo, en la cual se logró integrar hasta 40,000 compuertas y que incluía memoria RAM en el chip [22]. En la Figura 2.3 se muestra algunas características en cuanto a compuertas, donde destaca el número de celdas lógicas así como el intervalo de compuertas típico para algunas de las familias de dispositivos.

Dispositivo	Celdas Lógicas	Máx. núm. de compuer-tas de sistema	Intervalo típico de compuertas (Lógico y RAM)	Matriz de CLB	Total de CLB	No. De Flip-Flops	No. Máx. de E/S de usuario
XCS05 & XCS05XL	238	5,000	2,000 - 5,000	10 x 10	100	360	77
XCS10 & XCS10XL	466	10,000	3,000 - 10,000	14 x 14	196	616	112
XCS20 & XCS20XL	950	20,000	7,000 - 20,000	20 x 20	400	1,120	160
XCS30 & XCS30XL	1368	30,000	10,000 - 30,000	24 x 24	576	1,536	192
XCS40 & XCS40XL	1862	40,000	13,000 - 40,000	28 x 28	784	2,016	205

Figura 2.3 Características de las series SPARTAN y SPARTAN-XL.

Los bloques lógicos programables (CLB, C*onfigurable Logic Blocks*) de los que se compone una FPGA son los elementos básicos que le dan la capacidad para ser reprogramada y reconfigurada. Por medio de los CLBS, se implementa la mayor parte de la lógica en una FPGA.

Se trata de bloques interconectados por canales de enrutamiento y que están rodeados de bloques programables de entrada/salida (IOB, *Programable Input/Output Blocks*). Los CLB pueden ser usados para implementar diversas funciones, esto quiere decir que tienen componentes combinacionales que permiten implementar funciones muy sencillas o complejas, además cada CLB contiene un par de *flip flops* que le permiten almacenar las salidas de las funciones en caso de una realimentación o comunicación con bloques adyacentes. Los canales de enrutamiento que rodean a los CLB como se muestra en la Figura 2.4 son derivados de tres tipos de interconexiones: longitud sencilla, longitud doble y *palangres* (tipo cordel). En la intersección de cada canal de enrutamiento horizontal y vertical hay una matriz de dirección de señales llamada Matriz de Interruptor Programable (PSM, *Programmable Switch Matrix*) [23].

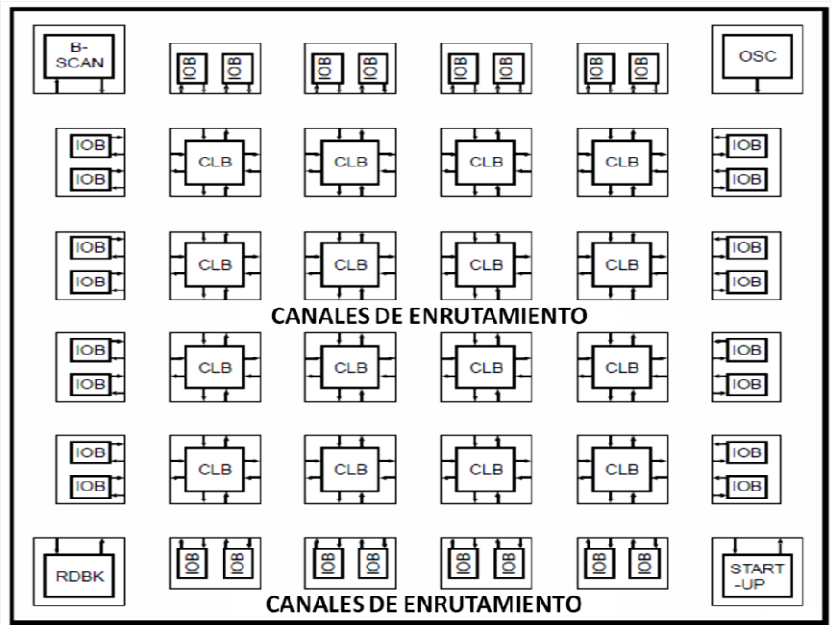


Figura 2.4 Diagrama Básico de un FPGA de Xilinx.

La arquitectura de la FPGA SPARTAN es simétrica como puede verse en la Figura 2.4, pero en el mercado de dispositivos reconfigurables hay diversos tipos de arquitecturas de acuerdo a los fabricantes, las otras arquitecturas son: Basada en canales, Mar de puertas y PLD jerárquica. Como se ha expuesto, los dispositivos FPGA permiten implementar soluciones personalizadas o desarrollar sistemas embebidos conocidos como *a la* medida, y además permiten hacerlo en un periodo de tiempo relativamente menor, acelerando el desarrollo y la implementación de sistemas y arquitecturas computacionales.

Es por esta razón que el uso de las técnicas *hardware in the loop* con estos dispositivos permite obtener resultados del sistema aún cuando no esté totalmente integrado, pero funcionando como tal, y, además, el desarrollo total del mismo considera un dispositivo de este tipo y no sólo se utilizado para validar pruebas experimentales aunque se trate de una primera aproximación. Por medio de *hardware in the loop* es posible la evaluación de elementos de hardware y esquemas de procesamiento antes de ser incluidos en una arquitectura de cómputo definitiva. Los detalles de esta técnica, en la que está basado el grueso del contenido experimental de esta tesis serán explicados en capítulos posteriores.

2.3 Modos de configuración para el FPGA

Para configurar y programar un dispositivo FPGA se debe seguir un protocolo de comunicación entre una computadora o un dispositivo de procesamiento que cuente con los recursos de software y hardware suficientes, así como una interfaz que actúe como vía entre la computadora y el dispositivo. Para llevar a cabo una reconfiguración total o parcial del dispositivo FPGA dependerá de la sofisticación del producto (recursos principalmente en hardware con los que cuenta el dispositivo) y las herramientas de software y hardware que ofrece el fabricante tanto en la plataforma de desarrollo que contiene a la FPGA como el software disponible para que la computadora pueda generar los *bitstreams* necesarios para llevar a cabo esas tareas.

Otro elemento que influye en dicha configuración es el modelo *Master-Slave* que tiene que ver con la transferencia de los datos en la operación de reconfiguración a través de los intermediarios que intervienen en este flujo, como pueden ser la memoria de datos y la caché de reconfiguración. Para descargar un flujo de datos (*bitstream* o flujo de bits) a la FPGA hay que elegir una forma de hacerlo, para ello hay tres maneras conocidas de realizarlo. La más común es aquella en la que el flujo de bits es transferido a través de un cable especial que permite el acoplamiento eléctrico para el manejo de señales de datos entre la computadora y la FPGA, ejecutando un software en la computadora para gestionar el flujo deseado a través del cable.

La segunda forma es usar un microcontrolador en la plataforma de desarrollo con las instrucciones adecuadas, un firmware, de tal forma que pueda programar al dispositivo. La tercera forma es la menos habitual ya que consiste de un dispositivo externo, vendido por separado por el fabricante o un tercero, y que se trata de una PROM de *arranque* conectada a la FPGA, la cual programa al dispositivo al encenderse la plataforma de desarrollo.

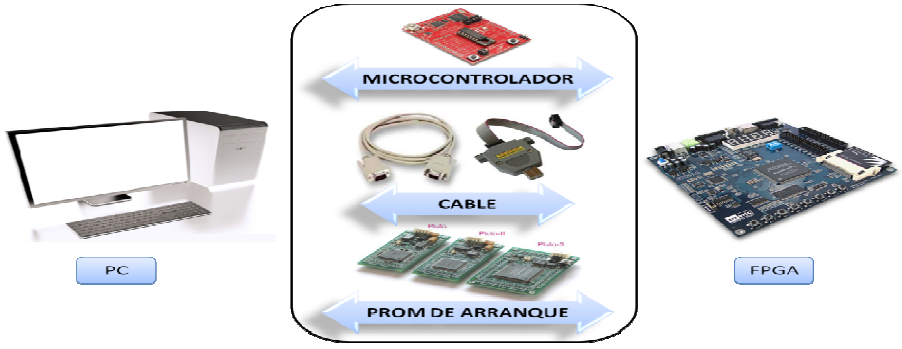


Figura 2.5 Principales formas de descargar un *bitstream* de una PC a una FPGA.

De estos métodos, el primero es el más sencillo y rápido para programar la FPGA, por ello es la forma más habitual de programar en una FPGA, es decir transferir el flujo de bits hacia el dispositivo por medio de un cable especial, esta forma de hacerlo tiene protocolos de comunicación asociados, los cuales serán descritos a continuación.

2.3.1 Configuración por interfaz JTAG (*Joint Test Action Group*)

JTAG es un estándar de comunicación y transferencia de datos, comúnmente utilizado para la configuración y programación de dispositivos lógicos programables. El nombre JTAG es el acrónimo de sus creadores, Joint Test Action Group, fue lanzado al mercado a mediados de la década de 1980; llegó a ser el estándar 1149.1 por la IEEE en 1990 para el puerto de prueba de acceso y límite de exploración [24]. El estándar se enfoca en asegurar la compatibilidad entre los Circuitos Integrados que lo cumplan. En la FPGAs se encargan de diversas funciones:

- *Programación*. descargar los contenidos de un archivo en un formato especial, dependiendo del fabricante (JEDEC, BIT, etc.) hacia los elementos lógicos de programación del dispositivo.
- *Verificación*. Volver a leer el contenido descargado al dispositivo y comparar el contenido con el archivo original.
- *Borrar*. Limpiar la información de configuración del dispositivo.
- *Pruebas funcionales*. Realiza pruebas de comparación de los resultados obtenidos contra los valores esperados, reportando las diferencias encontradas a los usuarios.

Puede llegar a realizar una mayor cantidad de funciones adicionales, pero dependerá del fabricante de la FPGA. El esquema de trabajo de la interfaz consiste de 5 señales principales:

TCK (*Test Clock*). Esta señal no tiene que ver con la señal de reloj del sistema o del dispositivo, se utiliza solamente para cargar los datos en modo de prueba del pin TMS (descrito más adelante) y los datos de prueba del pin TDI (descrito más adelante) en el flanco de subida de la señal entrante.

TMS (*Test Mode Select Input*). Esta señal controla la operación de la prueba lógica al recibir la llegada de los datos.

TDI (*Test Data Input*). Recibe los datos de entrada serial ya sea que se alimente de los registros de datos de prueba o de los de instrucciones.

TDO (*Test Data Output*). Los datos aplicados al pin TDI aparecerán en el pin TDO pero pueden ser desplazados un número de ciclos de reloj, dependiendo de la longitud del registro interno. La señal TDO es la salida del dispositivo JTAG que alimenta la entrada TDI de otro dispositivo JTAG. Esta señal es de alta impedancia.

TRST (*Test Reset*). Reiniciará asincrónicamente la lógica de prueba del JTAG. La lógica se reinicia independientemente del estado de TMS o TCK.

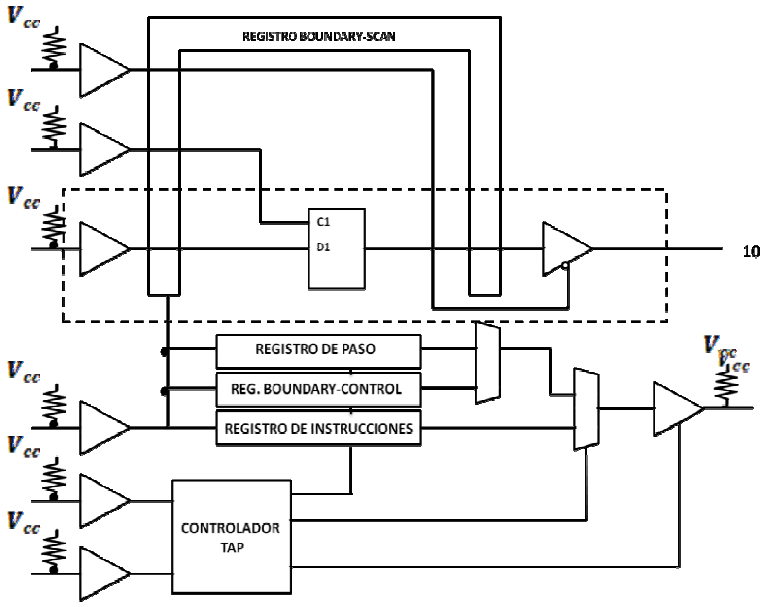


Figura 2.6 Diagrama lógico de un circuito de dispositivo JTAG.

2.3.2 Configuración por interfaz Serial Sincrónica

La interfaz serie RS-232 es la más común para realizar transmisiones de datos entre diversos dispositivos. Se trata de un estándar que constituye la tercera revisión de la antigua norma RS-232, propuesta por la EIA (Asociación de Industrias Electrónicas). La interfaz se compone de un conector DB-25 de 25 pines o un DB-9 de nueve pines, y es, en términos económicos, la más barata y de uso más extendido entre algunos dispositivos. Las señales con las que trabaja esta interfaz son digitales de +12V (0 lógico) y -12V (1 lógico) para la entrada y salida de datos, e inversamente en las señales de control. En las FPGAs se tiene una configuración de terminales diferente a la que se utiliza en dispositivos convencionales, ya que hay un proceso adicional de acoplamiento de señales que es explicado más adelante [25].

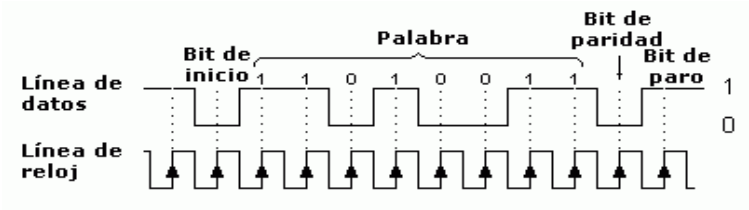


Figura 2.7 Diagrama de señales de interfaz serie.

Existen algunas señales que las interfaces utilizan en los dispositivos FPGA como identificadores para realizar diversas tareas, y a su vez son etiquetadas por los fabricantes de forma distinta pero tienen la misma función. En seguida se explican dichas señales.

Data (*data para Xilinx, data0 para Altera*). Es una entrada hacia la FPGA y corresponde a los bits de datos de configuración.

CLK (*clk para Xilinx, dclk para Altera*). Es una entrada a la FPGA. Es el reloj de configuración, y los bits de datos de configuración obedecen esta señal, desplazándose en el flanco de subida de la señal entrante.

PROG (*prog_b para Xilinx, nConfig para Altera*). Es una señal de entrada a la FPGA. Cuando acierta la señal, activo en bajo, la FPGA se reiniciará y se perderá la configuración. Si la FPGA se encuentra en *modo usuario* se detiene la operación inmediatamente, y todas las entradas y salidas regresan al *modo triestado*.

STATUS (*init_b para Xilinx, nStatus para Altera*). Se trata de una señal de salida de la FPGA e indica cuando esta está lista para comenzar el proceso de configuración.

DONE (*done para Xilinx, ConfDone para Altera*). También es una señal de salida de la FPGA, y cuando está en alta indica que la FPGA está configurada.

Como se ha descrito más arriba, la tendencia de desarrollo de dispositivos reconfigurables es la de ofrecer dispositivos más robustos y eficientes por parte de los fabricantes, también se presenta la variedad de formas en que se puede descargar el *bitstream* hacia la FPGA por medio de algunas de las interfaces más populares. Esta tendencia declina por la de utilizar la configuración por vía serial pues su fabricación es menos costosa y más imperante en el mercado electrónico, y con ello ha mejorados sus características, ofreciendo mayores velocidades de transmisión.

2.4 Software y herramientas de desarrollo para FPGA

Actualmente en el mercado existen una gran cantidad de herramientas disponibles para los desarrolladores en FPGA para algunos sistemas operativos y pueden ser de carácter gratuito o de pago. Los fabricantes ofrecen paquetes completos de desarrollo, que incluyen software y hardware para el diseño, síntesis y programación de los dispositivos. Además de esas herramientas, también ofrecen gran cantidad de documentación tales como notas de aplicación, hojas de diseño, de datos y de especificaciones; material que además de permitir conocer detalles de los dispositivos y entornos de desarrollo, facilitan el aprendizaje para la integración de sistemas embebidos en FPGA. Por otro lado, muchos desarrolladores han creado herramientas de licencia gratuita y de código abierto como respuesta a la necesidad de soluciones específicas y fomentar el compartimento con otros desarrolladores en todo el mundo.

Sin embargo, no hay un proyecto de tipo *open source*, desarrollo abierto, que tenga todas las herramientas necesarias para generar la compilación, síntesis y programación que requieren las FPGA, pero sí hay una importante aportación en software de depuración, compilación y simulación para estos dispositivos. Dentro de las herramientas comerciales (de pago) destacan las suites ISE Xilinx, MAX PLUS de Altera, Libero de Microsemi, entre otros. A lo largo de este capítulo se abordarán aspectos generales sobre las suites comerciales, solamente.

Es importante remarcar que gran parte del desarrollo para FPGA se basa en el lenguaje HDL (*Hardware Description Language*) del cual hay diversas variantes como Verilog y Abel; más adelante se abordarán detalles de modelos de programación más complejos que pueden soportar las FPGA para distintas aplicaciones.

2.4.1 Integrated Software Environment (ISE) de Xilinx

La suite ISE de Xilinx proporciona un conjunto completo de herramientas para el desarrollo de aplicaciones y sistemas embebidos en dispositivos FPGA, ofrece además el soporte necesario para las distintas familias de FPGA que fabrican. Consta de un editor, un compilador de código con soporte para C y C++, herramientas de síntesis para lenguaje descriptor de hardware, simulación, análisis y depuración, entre otras.

El software está disponible de forma gratuita en una versión llamada ISE WebPack, y la versión completa está disponible mediante la compra al fabricante, aunque también se pueden gestionar licencias para uso académico por medio de programas universitarios. A continuación se describirán los componentes que integran esta suite de desarrollo, *grosso modo* [26].

- ***ISE (Project Navigator y accesorios).*** Es el editor principal, el cual permite la edición de código VHDL (*Verilog Hardware Description Language*), la compilación, síntesis del diseño, generación del flujo de bits y programación de éste en la FPGA. ISE maneja una metodología estandarizada para la implementación de diseños en FPGA que comprende varias etapas, y este caso es común en otras suites para desarrollo en FPGA (Figura 2.8), que se resumen en: diseño inicial, que comprende una abstracción de los componentes que integran el sistema a desarrollar descritos mediante código; la síntesis, que es la generación de una *lista de redes* a partir del código VHDL, colocada en un archivo de forma no jerárquica y que describe cómo se distribuyen o *mapean* los elementos del diseño; implementación, que se enfoca en la simulación del sistema desarrollado, así como su análisis a través de un editor de forma de onda (*waveform*) interactivo o un banco de pruebas (*test bench*); y finalmente, la programación, que consiste en descargar en la FPGA el flujo de bits generado a partir del diseño completo de forma funcional y probado.

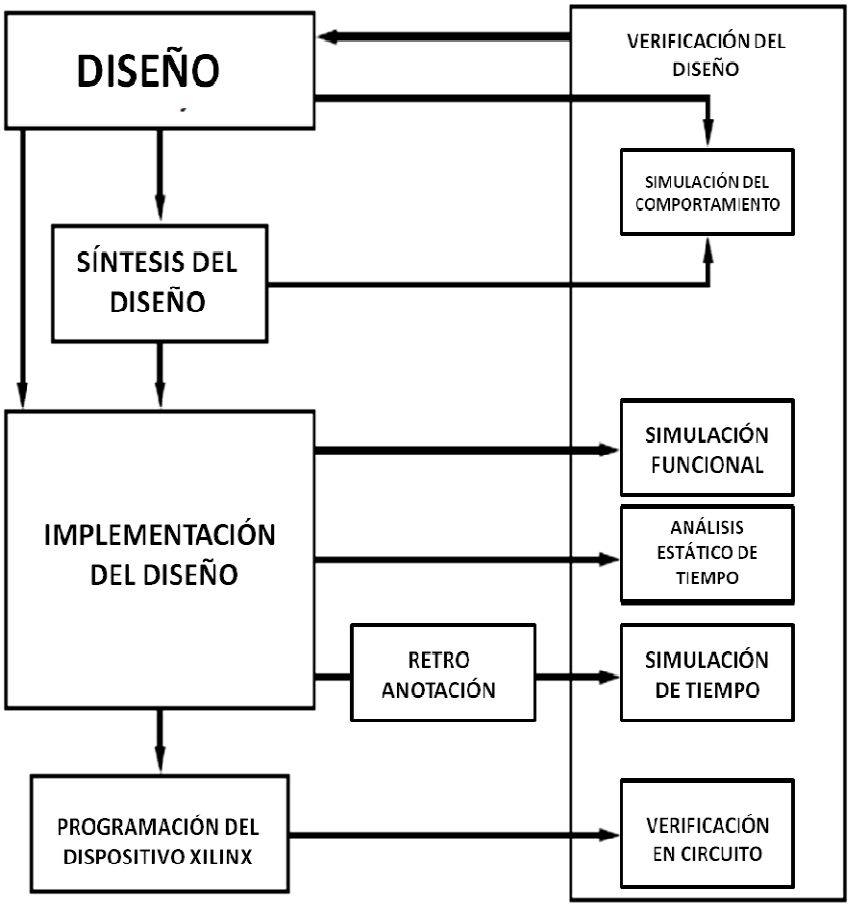


Figura 2.8 Esquema de desarrollo en ISE Xilinx.

- ***EDK* (*Embedded Development Kit*): *SDK* (*Software Development Kit*), *XPS* (*Xilinx Platform Studio*), y *Chipscope*.** EDK contiene las herramientas necesarias para implementar arquitecturas de cómputo dedicadas basadas en núcleos de procesamiento embebidos tales como *soft cores* e *IP (Intelectual Property)* de propósito específico (comunicaciones, coprocesamiento) comerciales y de diseño propio. Esta herramienta genera las porciones de hardware y software en conjunto con el procesador embebido también conocido como *soft processor*. Ofrece soporte para aplicaciones de software embebidas escritas en los lenguajes C y C++. Además, contiene un analizador de señales, depurador y monitor en tiempo real de los recursos del sistema embebido llamado ChipScope.

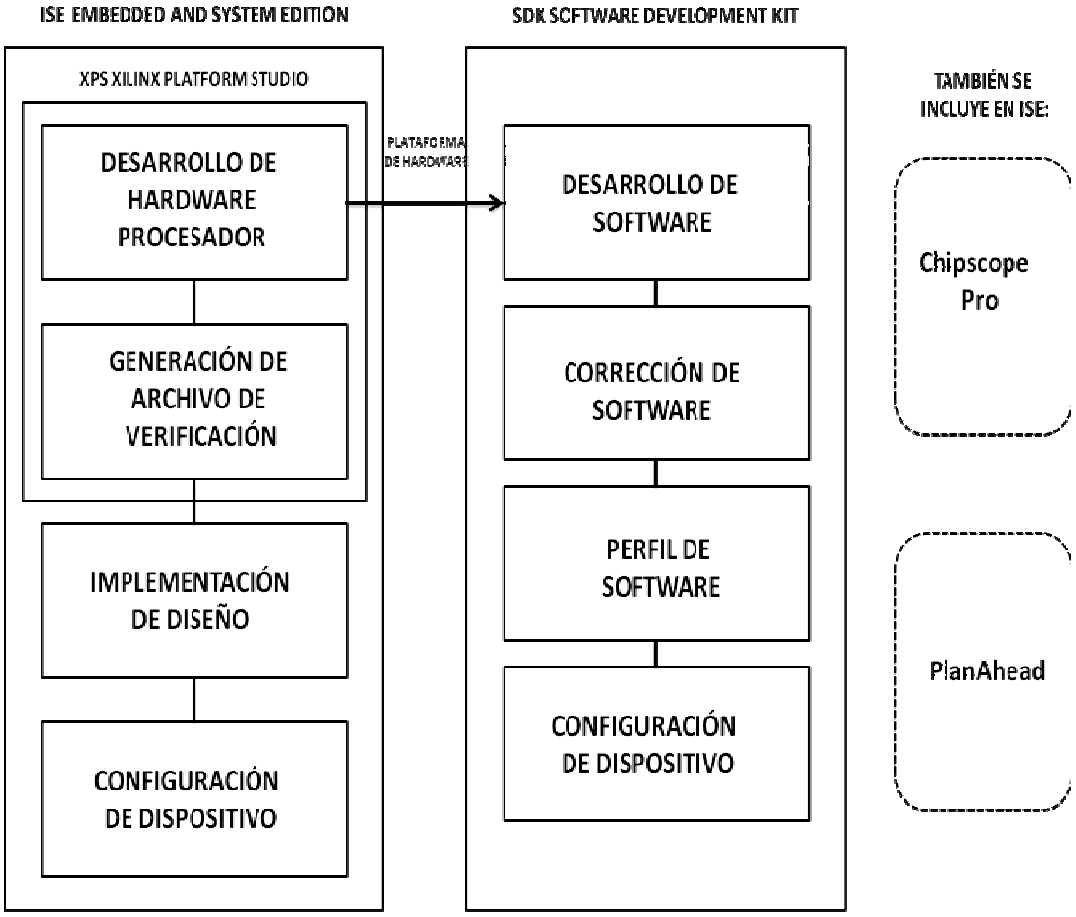


Figura 2.9 Esquema para el desarrollo de aplicaciones *softcore* o núcleos periféricos.

2.4.2 Qsys de Altera

La *suite* de software Qsys de Altera provee soporte para las plataformas FPGA, CPLD y ASIC. Se rige bajo el mismo esquema de desarrollo de aplicaciones para FPGA que se mencionó anteriormente, con un enfoque ligeramente distinto pero esencialmente estándar, esto quiere decir que hay una etapa de diseño, síntesis, implementación y programación; sólo que cada etapa y sus diversos procesos se realizan de acuerdo a las características del software que provee el fabricante, y cada tarea se lleva a cabo por separado (Figura 2.10). Qsys también ofrece una versión gratuita y una versión de pago para adquirir su conjunto de herramientas.

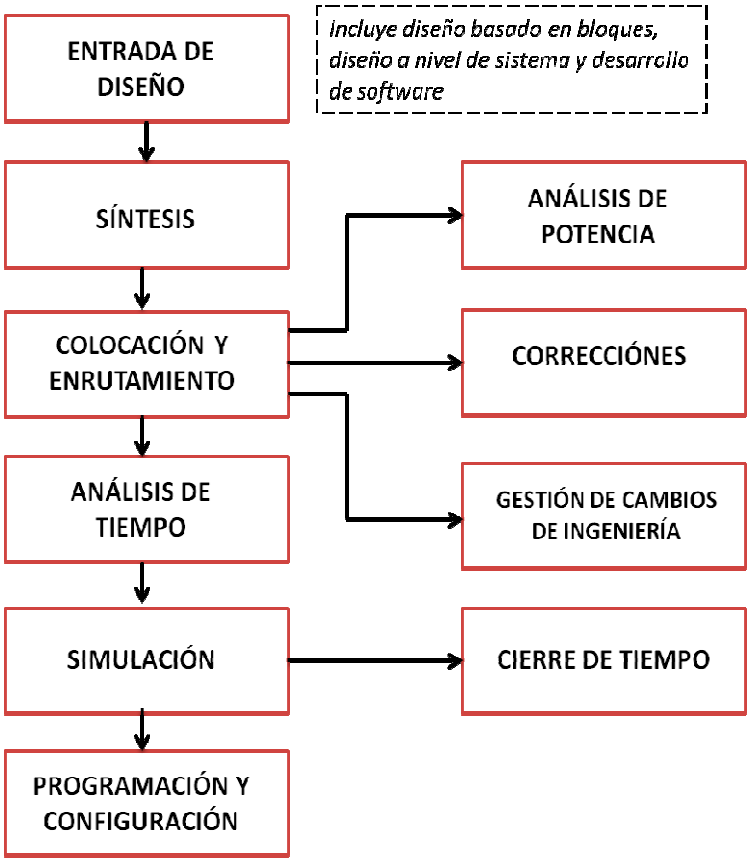


Figura 2.10 Esquema general de desarrollo en Qsys de Altera.

QSys también contiene los siguientes elementos de desarrollo [27].

- ***Quartus II y* accesorios.** Quartus II es un software que provee una interfaz gráfica que permite la edición de código HDL y su síntesis, además contiene una serie de accesorios para la simulación y el análisis de señales del diseño en tiempo real, así como la estimación de potencia para la arquitectura desarrollada, una vez que sea implementada en hardware.

- ***Nios II Embedded Design Suite*.** Esta aplicación provee las herramientas necesarias para el diseño de sistemas embebidos, tales como un editor de código con soporte para lenguaje C y C++, un depurador, compilador y enlazador de código.

- ***ModelSim***. Es una herramienta en software que administra un entorno que permite editar, compilar, simular y depurar diseños de sistemas digitales descritos en lenguaje HDL, Verilog y Systemc.

2.4.3 Suite de software Libero de Microsemi

Libero es una suite de software para el desarrollo de aplicaciones y diversos dispositivos de Microsemi, y al igual que Xilinx y Altera, utiliza el mismo esquema para la implementación de de diseños en FPGAs. Cabe señalar que Microsemi produce dispositivos FPGA de tecnología antifuse y tecnología antifuse y FPGAs flash, así como procesadores y comercializa núcleos de procesamiento (*ip cores*) de propósito específico. En este diagrama se muestra el esquema de diseño de Microsemi (Figura 2.11).

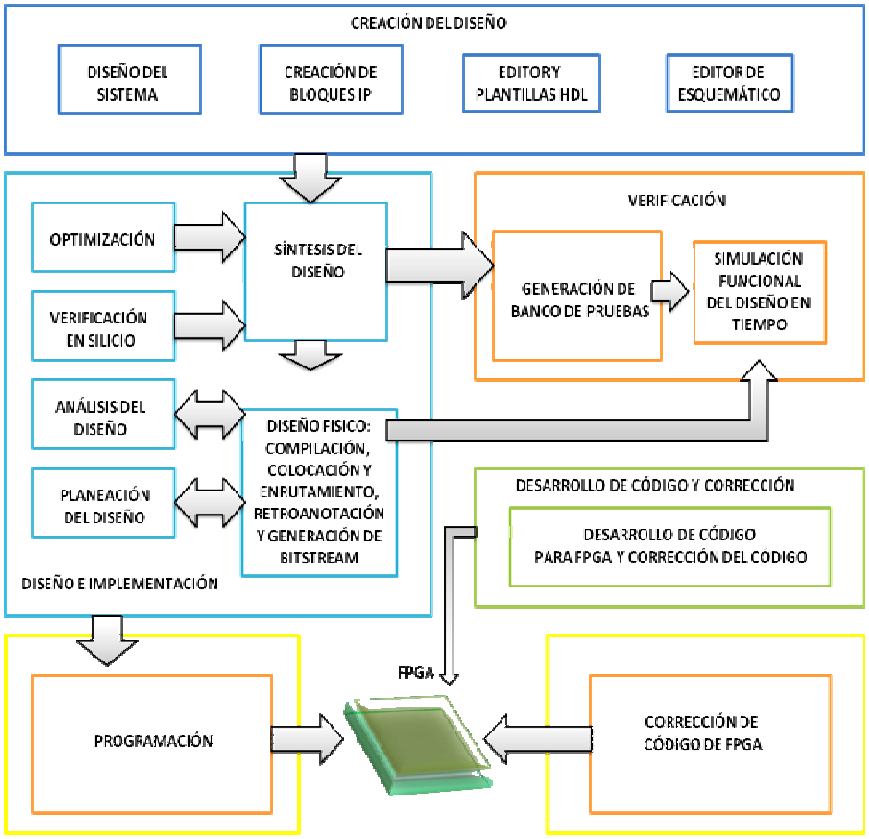


Figura 2.11 Esquema de diseño en Libero de Microsemi.

Los componentes de Libero son los siguientes [28]:

- ***SmartDesign y accesorios*.** Esta aplicación ofrece las herramientas para la edición de programas en código HDL y su posterior síntesis, también comprende la creación de núcleos personalizados y ofrece plantillas para éstos, contiene también un editor de diagramas esquemáticos, los cuales pueden ser sintetizados de forma similar a los programas escritos en lenguaje HDL.

- ***Symplify AE y accesorios de implementación de diseños.*** Este conjunto de herramientas ofrece el software necesario para la implementación del diseño que abarca la síntesis de diseños propios en *SmartDesign*, así como aquellos ofrecidos por terceros. Además contiene accesorios que se encargan de generar la lista de redes (netlist) y *mapear* los elementos del diseño de forma lógico, así como interconectar dichos elementos del diseño.
- ***FlashPro y accesorios***. Este software se encarga de realizar la tarea de programación del dispositivo. Microsemi proporciona adicionalmente herramientas para la *depuración* del diseño para sus diversos productos y tecnologías. También proporciona una consola para la *depuración* del código como parte de un desarrollo más robusto.

En este breve resumen de las características de las *suites* de desarrollo para FPGA se han expuesto claramente que comparten un esquema de trabajo estándar, al cual le han agregado un enfoque ligeramente distinto, con etapas adicionales, pero que intrínsecamente, llevan a cabo tareas en común para la programación del sistema embebido en el dispositivo. También se comentaron las herramientas de desarrollo, para la edición, síntesis, programación, análisis, mejoramiento y corrección del proyecto, y de las cuales, algunas dependerán de los recursos y características técnicas de la plataforma FPGA. De igual forma, se ha explicado que hay diversas herramientas para los desarrolladores aparte del software y hardware, y es la documentación, muy útil para adquirir un mayor conocimiento de las herramientas disponibles así como para acelerar el aprendizaje para integrar los sistemas y bloques de hardware y software de un determinado proyecto . Estas herramientas son muy importantes para lograr buen éxito en el desarrollo del sistema que se pretenda integrar.

2.5 Hardware de desarrollo para FPGAs

Este apartado incluye la descripción de algunas de los sistemas comerciales para FPGA más importantes del mercado. Los mismos fabricantes de las *suites* de software descritas en el apartado anterior, son los mismos que dominan el mercado de hardware para FPGAs: Altera, Xilinx y Microsemi. Se presentarán algunas características generales de algunas series de plataformas de dichos fabricantes. También es importante mencionar que para el desarrollo de esta tesis se ha seleccionado una plataforma de Xilinx, particularmente por la relación costo-beneficio que ofrece y en el sentido de la cantidad de soporte que hay en la red, su facilidad de uso y de las herramientas de desarrollo disponibles al momento de la realización experimental de esta tesis. Este hardware incluye un dispositivo principal: un FPGA, que además cuenta con más elementos para interconexión, interfaces y puertos de comunicación, así como dispositivos de salida y de entrada.

2.5.1 SPARTAN 3E-Board de Xilinx

La tarjeta de desarrollo FPGA SPARTAN 3E se caracteriza por la rapidez de implementación del diseño y menor consumo de potencia frente a otras de características similares. Cuenta con diversas interfaces de hardware que permiten la interacción con dispositivos y equipos externos, entre ellas destacan los puertos RS-232 (DCE Y DTE), puerto Ethernet, salida VGA y puerto PS/2. Incluye recursos de depuración tales como *push buttons, dip switches, leds* y un *display LCD de doble renglón*; la versión de esta plataforma de desarrollo utilizada está basada en el FPGA XC3S500E, el cual cuenta con alrededor de 500,000 compuertas lógicas programables disponibles; 4 [MB] de memoria *flash;* 16 [MB] de memoria RAM y soporta los sistemas embebidos PicoBlaze y MicroBlaze. Tiene conectado un oscilador a 50 [Mhz], aunque también cuenta con dos conectores físicos para proporcionar señales de reloj de diferente origen y frecuencia. El desarrollo en esta tarjeta es transferible a otros dispositivos de la misma familia que cuenten con los recursos suficientes para soportar al *soft core* Microblaze [29]. Para la realización de los experimentos de esta tesis, se utilizará esta tarjeta de desarrollo, así como su ambiente y herramientas en software asociadas, descritos brevemente en otros apartados de este capítulo.

Figura 2.12 Vista superior de una tarjeta de desarrollo SPARTAN 3E de Xilinx.

2.5.2 Cyclone II de Altera

La tarjeta de desarrollo Cyclone II Starter Development ofrece características integradas que permiten al usuario desarrollar y probar diseños que van desde circuitos digitales sencillos hasta complejos proyectos multimedia, para ello ofrece según el fabricante una interfaz sencilla, software de control residente o controladores de memoria flash, SRAM o DRAM. Los recursos que posee le permiten soportar gran cantidad de aplicaciones, incluyendo sistemas embebidos. Cuenta con una memoria estática RAM de 8 [MB], interfaces Serial RS-232, VGA, PS/2, un modulo doble de 40 pines y uno SD/MMC. Tiene hasta más de 60,000 elementos lógicos programables y la frecuencia del reloj del sistema alcanza los 20 [MHz] [30].

Figura 2.13 Vista superior de la placa Cyclone II de Altera.

2.5.3 Igloo de Microsemi

La plataforma IGLOO de Microsemi es una placa que contiene un dispositivo de nanotecnología FPGA de 250, 000 compuertas. Contiene una memoria estática RAM y una memoria no volátil (NVM) para la configuración del FPGA, y cuenta tanto con una interfaz serial USB como una LCPS (*Low-Cost Programming Stick*), con módulos para la medición de corriente, así como para pruebas de las terminales de entrada/salida, un oscilador a 20 [MHz], así como recursos de depuración y validación como *push buttons*, *dip switches* y *leds.* Se caracteriza por su bajo consumo de potencia intercambiable a niveles de 1.5, 2.5, y 3.3 [V] [31].

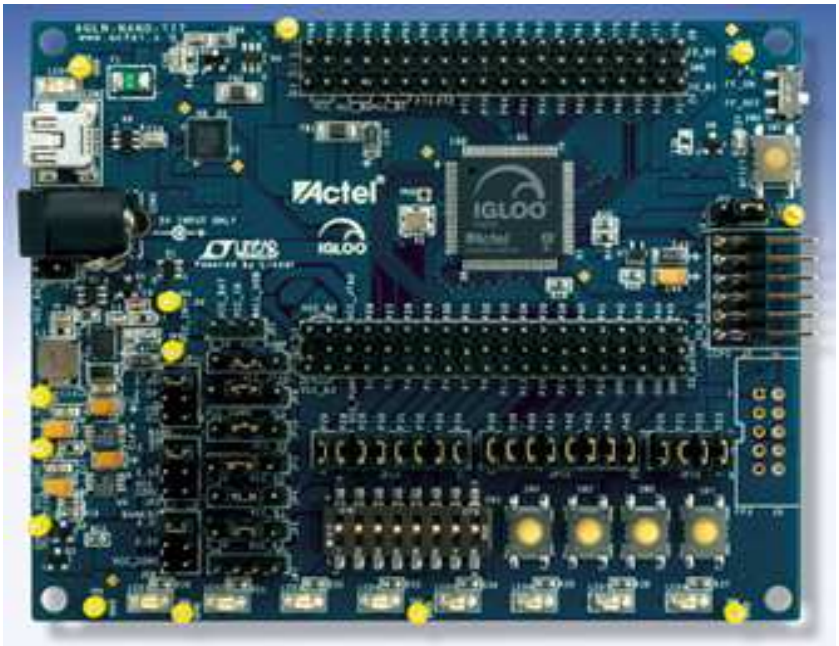


Figura 2.14 Vista general de la placa IGLOO de Microsemi

En el desarrollo de esta tesis se propone un esquema de trabajo basado en la técnica *hardware in the loop*, y parte de este trabajo es embeber una arquitectura en una plataforma de desarrollo FPGA. Por esto, se ha presentado de forma general una descripción de diversas familias, fabricantes y plataformas de desarrollo FPGA en las que se muestran características técnicas relevantes, la composición y tecnologías FPGA, el consumo de potencia, escalas de integración, así como la programación a través de plataformas de desarrollo que las incluyen como unidad de procesamiento, que a su vez cuentan con puertos de comunicación, elementos de validación, conjuntos de memorias, etcétera. Por otra parte, se ha complementado la información de esas plataformas con un resumen del software y las herramientas que ofrecen los fabricantes para el desarrollo del diseño, implementación, corrección y simulación de la arquitectura; estas herramientas tienen un costo para el desarrollador que pretenda adquirir la suite completa de herramientas, y hay versiones académicas gratuitas. Muchas de esas herramientas se utilizarán para el desarrollo de la primera etapa de este trabajo que consiste en la adquisición de datos de los sensores de navegación a través de una plataforma de desarrollo SPARTAN 3E. Con esto finaliza la parte introductoria y descriptiva de elementos básicos a tomar en cuenta para el desarrollo del sistema de simulación satelital para la validación de esquemas de control que se pretende integrar, donde estos capítulos forman parte del planteamiento de ese sistema, y a continuación, se hará un análisis cinemático y dinámico de él, donde se definirá la ecuación de movimiento, que permitirá determinar las variables que pueden manipularse para lograr el control del sistema.

Capítulo 3

Cinemática y dinámica de la Mesa Suspendida en Aire

El desarrollo de este capítulo abordará el análisis de la cinemática y la dinámica del sistema de simulación de vuelo satelital, llamado Mesa Suspendida en Aire (MSA), con que cuenta el Instituto de Ingeniería de la UNAM. La MSA es una plataforma experimental que permite la validación y verificación en Tierra del desempeño de los algoritmos de control de orientación en tres ejes para vehículos espaciales, particularmente de satélites pequeños. El planteamiento del análisis de movimiento de la MSA, se basa en la conceptualización del simulador como un cuerpo rígido. El desarrollo de la ecuación dinámica que describe a la MSA parte de un análisis de la cinemática donde se analiza la interacción de los marcos de referencias inerciales y móviles con respecto a cada uno de los componentes del simulador (plataforma y actuadores), para dar paso posteriormente al análisis dinámico. Mediante el uso del concepto de momento angular se analizarán las aportaciones al movimiento de la plataforma, por una parte de la propia plataforma del simulador y por otra de los actuadores basados en ruedas inerciales instalados a bordo de ella.

3.1 Introducción

En los capítulos anteriores se expuso un breve panorama de los sistemas de simulación satelital que se han convertido en referentes para otros desarrollos en ese mismo campo de investigación, presentando sus principales características de hardware y software, así como la capacidad de maniobras que pueden realizar, y por otro lado, se realizó una revisión general de los dispositivos que conforman el sistema de instrumentación, como son los procesadores, los sensores de navegación y los actuadores; todo esto fue a manera de planteamiento, ahora, como parte del análisis del problema se ha retomado un modelo matemático y partir de él se desarrollará la ecuación dinámica, con la cual podrán determinarse las variables que pueden manipularse para los objetivos del control.

En este capítulo se introduce el concepto de MSA, y con él se referirá a la plataforma de simulación con sus elementos a bordo, siendo sólo una parte del sistema completo denominado sistema de simulación satelital para validación de esquemas de control, no obstante, el objetivo a mediano plazo es que la misma MSA contenga al sistema completo, integrando el concepto bajo una misma denominación. Antes de desarrollar el planteamiento del análisis matemático se expondrán los elementos básicos para introducirse al estudio de la cinemática y dinámica de un cuerpo rígido.

El capítulo, entre otras cosas, se enfocará en plantear los conceptos fundamentales para abordar la teoría de la dinámica del cuerpo rígido, con la cual se pretende desarrollar el modelo matemático del movimiento de la MSA, tomando en cuenta los componentes físicos que la integran. Del análisis matemático del movimiento de la mesa se debe tener como resultado un modelo, ecuaciones, que describa el comportamiento del sistema. En ese contexto se abordará someramente las definiciones de conceptos básicos para un mejor entendimiento de la cinemática y dinámica del cuerpo rígido y posteriormente se desarrollará el modelo de la MSA tomando como referencia esos conceptos.

3.2 Representación de la orientación de un cuerpo rígido

Este capítulo trata del planteamiento del modelo matemático para obtener la ecuación de movimiento de la MSA que forma parte del sistema de simulación satelital para la validación de esquemas de control, pero antes de abordar dicho modelo, se hará un breve repaso de los elementos necesarios a manera de introducción.

Para entender los apartados posteriores, se iniciará por exponer lo que es un cuerpo rígido y que un concepto ideal del que parte todo el análisis. Por ellos se debe definir lo que idealmente es un *cuerpo rígido*. El cuerpo rígido es un caso especial de los sistemas que están constituidos por muchas partículas, es un cuerpo en el cual las distancias entre todos sus componentes permanecen constantes bajo la aplicación de una o varias fuerzas o momentos, como se muestra en la Figura 3.1. Por lo tanto, un cuerpo rígido conserva su forma durante todo su movimiento [32].

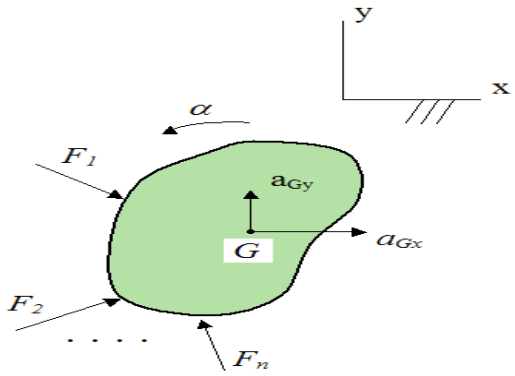


Figura 3.1 Conceptualización de un cuerpo rígido.

Para este concepto es posible distinguir dos tipos de movimiento de un cuerpo rígido: **traslación** y **rotación**. El primero se produce cuando todas las partículas describen trayectorias paralelas de tal modo que las líneas que unen dos puntos cualesquiera del cuerpo permanecen siempre paralelas a su posición inicial. El segundo describe trayectorias circulares alrededor de una línea llamada eje de rotación; en particular este movimiento es objeto de estudio para los fines de modelo de la MSA, ya que como se verá más adelante, la MSA estará sujeta a movimientos de este tipo, donde no hay desplazamiento lineal de la MSA, lo cual es de particular interés para este trabajo; ambos se muestran en la Figura 3.2 a) y b).

a) b)

Figura 3.2 Tipos de movimiento de un cuerpo rígido a) Traslación b) Rotación.

Una vez que se ha definido el concepto de un cuerpo ideal y sin variaciones en cuanto a la distancia entre las partículas que lo componen, se deben tener en cuenta más elementos que permiten el estudio del movimiento. Por ello, se necesitarían definir puntos para determinar la posición y orientación del cuerpo, por lo que se requieren coordenadas de la orientación, algunas veces referidas como parámetros de orientación o actitud, las cuales son grupos de coordenadas que describen completamente la orientación de un cuerpo rígido relativo a algún marco o sistema de referencia. Describir la orientación de un objeto relativo a un marco de referencia difiere en la forma fundamental de describir la correspondiente posición espacial relativa de un punto. Cabe señalar que, en el ámbito que aborda esta tesis, aunque un vehículo espacial no es perfectamente rígido, sobre todo en el caso de un satélite equipado con ruedas inerciales, el modelo del cuerpo rígido es una muy buena aproximación para estudiar la dinámica de la orientación.

La búsqueda para la mejor descripción de la orientación de un cuerpo rígido es muy importante. Una correcta selección de la representación del sistema de coordenadas para describir las coordenadas de la orientación puede simplificar de forma importante las matemáticas y evitar ciertas dificultades, tales como singularidades matemáticas y geométricas. Contrario a esto, una mala selección puede limitar el intervalo de operación de un sistema controlado, requiriéndole operar dentro de un intervalo no singular de los parámetros de descripción de la orientación seleccionados. Entonces, para realizar esta descripción se debe describir el nuevo concepto introducido: marco de referencia.

3.2.1 Marco de referencia

El marco de referencia se utiliza comúnmente para la descripción cinemática de una partícula o de todo un cuerpo rígido, así como para el análisis de cantidades dinámicas asociadas a un cuerpo o bien, de un cuerpo o partícula relativas a otro punto de referencia. Un marco de referencia, o sistema coordenado de referencia, se define generalmente como un grupo de tres vectores unitarios, los cuales son mutuamente perpendiculares y que definen un espacio cuyo origen es definido por un punto en el que se interceptan los tres vectores. Este sistema de referencia es indefectible, ya que con base en él, se harán cálculos fundamentales que determinaran la posición del cuerpo rígido estudiado, como se ha dicho, pues es la base para realizar la descripción de la orientación [33].

Un ejemplo del sistema coordenado de referencia puede verse en la Figura 3.3, donde **S** y **S'** se mueven a un velocidad relativa $\vec{u}$, en el que la posición de **S** se ha definido arbitrariamente y permanece en posición fija respecto de **S'**, independiente de la velocidad, tal que **S'** es un sistema móvil que proyecta su posición sobre el primer sistema.

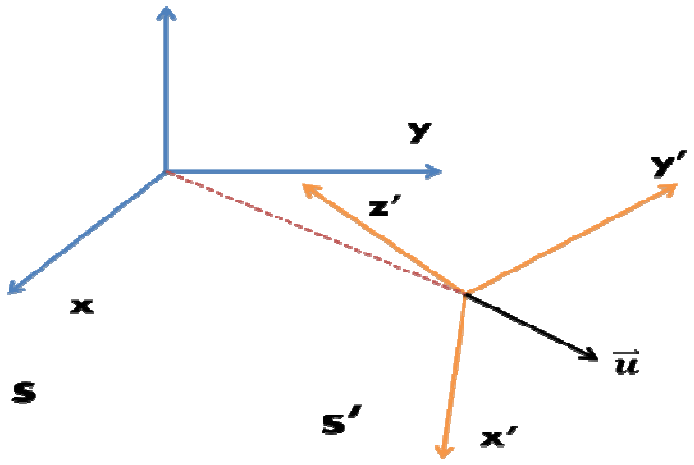


Figura 3.3 Dos marcos de referencia moviéndose con una velocidad relativa $\vec{u}$.

Como pudo verse de forma gráfica el sistema coordenado de referencia es un auxiliar que permite proyectar el movimiento de un sistema móvil con respecto a uno fijo. Pero, para completar la descripción, se deben trazar vectores, los cuales indican la posición de los objetos en el espacio del sistema coordenado.

3.2.2 Vectores

Ya se ha definido el sistema de referencia donde se puede describir la posición de un cuerpo, ahora toca describir las componentes de esa posición: la entidad básica de la dinámica es el vector. Un vector tiene dos propiedades: magnitud y dirección, y es representado por magnitudes a lo largo de cualesquiera de los tres ejes mutuamente perpendiculares, llamado en conjunto marco de referencia o sistema de coordenadas, como se ha visto. Cada eje de un marco de referencia coordenado está representado por un vector unitario, definido como un vector de magnitud igual a uno [34].

Consideremos entonces, verbi gratia, un vector **A** escrito en términos de sus magnitudes a lo largo de un marco de referencia consistente de una triada de vectores unitarios, **i**, **j**, **k** como se muestra en esta ecuación: **B = B $_x$i + B $_y$j + B $_z$k;** o simplemente como: **B =** $\{B_x B_y B_z\}$, y como se observa en la Figura 3.4 donde el vector marca su posición reflejando su trayectoria en los planos correspondientes entre los ejes, además se observan los ángulos trazados, pero por ahora, no se definirán aún, pues no son parte de lo que se trata de explicar ; por último la punta de flecha del vector indica su dirección y su magnitud estará dada por la distancia entre el origen en este caso y el punto **B.**

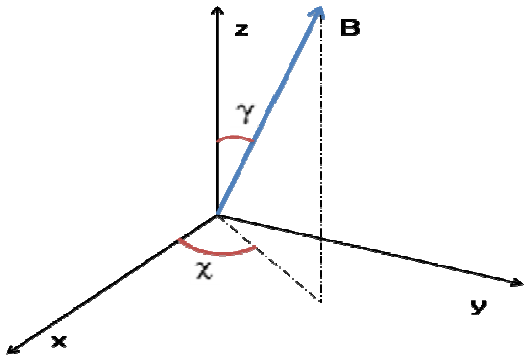


Figura 3.4 Ejemplo de un vector B en un marco de referencia xyz.

Con fundamento en estas definiciones básicas con representación geométrica, se pueden abordar nuevos conceptos que forman parte del bagaje necesario para la obtención del modelo de la MSA. En seguida se abordará el concepto de *matriz de rotación* que conjunta los conceptos anteriores.

3.2.3 Matriz de rotación

Lo que sigue ahora es definir lo que es una matriz de rotación. Esta es una matriz ortogonal cuyo determinante es igual a 1, lo cual significa que la inversa de esa matriz es igual a la transposición de la misma. Visto de otro modo, existe un grupo especial ortogonal de todas las matrices de rotación de 3x3 (en este caso) que es denotado por ***SO(3)***. Por lo tanto, si una matriz de rotación $\boldsymbol{R} \in \boldsymbol{SO}(\mathbf{3})$, entonces:

$$\det \boldsymbol{R} = \pm \mathbf{1} \quad \boldsymbol{y} \quad \boldsymbol{R^{-1}} = \boldsymbol{R^{T}} \qquad \textbf{(3.1)}$$

Conceptualmente se define a una matriz de rotación como aquella que establece la actitud u orientación de un cuerpo rígido, al ser la matriz que al premultiplicarse por un vector expresado en las coordenadas de un sistema de referencia determinado, produce el mismo vector expresado en un sistema de coordenadas fijo al cuerpo como se vio en los sistemas de referencia fijo y móvil. Formalizando las cosas, si $\boldsymbol{z}$, donde $\boldsymbol{z} \in \boldsymbol{R^3}$, es un vector en un sistema de referencia determinado, y $\boldsymbol{z'}$, donde $\boldsymbol{z'} \in \boldsymbol{R^3}$, es el mismo vector expresado en un sistema de referencia fijo al cuerpo, entonces se tienen las siguientes igualdades:

$$\mathbf{z' = Rz} \qquad \mathbf{y} \qquad \mathbf{z = R^T z'} \qquad \textbf{(3.2)}$$

Además, las matrices de rotación para las cuales **det *R = 1*** son llamadas propias, mientras que aquellas para las cuales **det *R = -1*** son llamadas impropias. La rotación de las matrices impropias es llamada *rotoinversión,* y consiste en la rotación seguida de una operación de inversión. En el espacio euclidiano se enunciarían tres matrices, aunque para los fines del capítulo, se representarán una matriz de forma general como sigue:

$$\boldsymbol{R} = [\boldsymbol{r_1}, \boldsymbol{r_2}, \boldsymbol{r_3}] = \begin{bmatrix} \boldsymbol{r_{11}} & \boldsymbol{r_{12}} & \boldsymbol{r_{13}} \\ \boldsymbol{r_{21}} & \boldsymbol{r_{22}} & \boldsymbol{r_{23}} \\ \boldsymbol{r_{31}} & \boldsymbol{r_{32}} & \boldsymbol{r_{33}} \end{bmatrix} \qquad \textbf{(3.3)}$$

Cabe señalar que la interpretación que tiene la matriz de rotación **R** [35], puede estar en uno de estos tres sentidos:

1) Como representación de la orientación de un cuerpo rígido
2) Para la descripción de la rotación un cuerpo rígido
3) Como herramienta para realizar la transformación de un sistema coordenado

En este trabajo, las tres interpretaciones tienen mucho que ver con lo que se planteará de manera formal más adelante. En resumen, los tres conceptos definidos en lo que va del estudio introductorio de este capítulo: sistema de referencia, vectores, y matriz de rotación, son la base de lo que sigue: los métodos para encontrar la mejor representación e la orientación, pues con ellos se justifica el análisis de la MSA.

3.2.4 Ángulos de Euler

Una de las formas más comunes de representar la actitud de un cuerpo rígido es un grupo de tres ángulos conocidos como ángulos de Euler. Este método representa una forma sencilla y relativamente fácil de obtener resultados deseados para representar la actitud u orientación de un cuerpo rígido [36]. Algunos de los grupos de ángulos de Euler son ampliamente usados en aeronáutica y tienen nombres específicos tales como *roll, pitch* y *yaw*, o guiñada, cabeceo y alabeo, como se les denomina en español, respectivamente.

En la Figura 3.5 se muestra su representación gráfica en un sistema coordenado de referencia, especificando los movimientos en cada unos de los ejes del sistema: guiñada es el movimiento en el eje **Y**, cabeceo en el eje **X** y alabeo en el eje **Z**. A pesar de sus características y facilidades tiene ciertas desventajas en cuanto a la representación que se busca para la MSA.

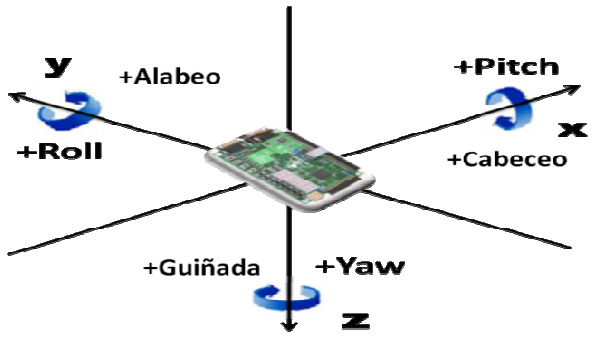


Figura 3.5 Representación gráfica de los ángulos de Euler.

Las principales desventajas que presenta esta representación de orientación del cuerpo rígido, están relacionadas con los siguientes detalles:

1) Ciertas funciones asociadas con la determinación de los ángulos de Euler presentan singularidades matemáticas, es decir, puntos para los cuales la función deja de ser continua o bien, no está definida.

2) Son menos precisos que los cuaterniones unitarios, de los que se definirán en el siguiente apartado, cuando son utilizados para integrar cambios de orientación sobre un periodo de tiempo.

3.2.5 Cuaterniones

Los cuaterniones fueron presentados como una extensión de los números complejos por William Rowan en 1843, y son utilizados de forma amplia en las áreas de aeronáutica, robótica y animaciones virtuales. En esta tesis, mediante este método se puede obtener la orientación de un objeto alrededor de un eje. Como se dijo, un eje es un vector unitario, representado en esta sección como **λ,** con componentes en un sistema coordenado de referencia ***X,Y,Z***, donde, a diferencia del método de ángulos de Euler, en los cuaterniones, además, se requiere de cuatro parámetros para describir la orientación respecto a un sistema coordenado de referencia: los tres componentes del vector unitario **λ** y un ángulo de rotación **θ** (Figura 3.6) [37].

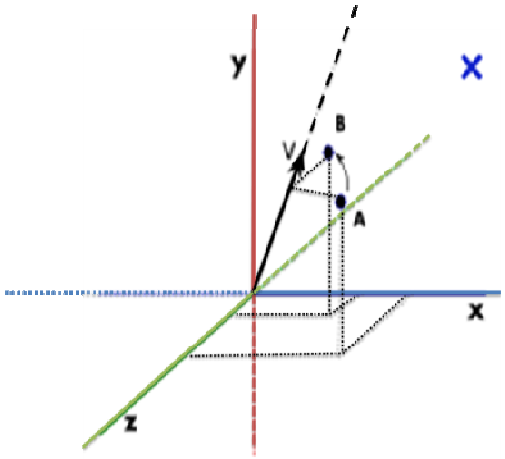


Figura 3.6 Representación gráfica de un cuaternión.

Un cuaternión, el cual se representará con la letra **q,** es un número que consta de dos partes, una parte escalar y una vectorial: **η** y **ε**, respectivamente, como se muestra a continuación:

$$q = [\eta \varepsilon_1 \varepsilon_2 \varepsilon_3] \quad \textbf{(3.4)}$$

Ahora bien, para obtener el cuaternión unitario se tiene lo siguiente:

$$\eta^2 + \varepsilon_1^2 + \varepsilon_2^2 + \varepsilon_4^2 = q^T q = 1 \quad \textbf{(3.5)}$$

Donde:

$$\eta = cos\left[\frac{\theta}{2}\right] \varepsilon = [\varepsilon_1 \varepsilon_2 \varepsilon_4] = \sin\left[\frac{\theta}{2}\right] \lambda \quad \textbf{(3.6)}$$

Las expresiones mostradas en **(3.6)** denotan la posibilidad de obtener, a partir de los parámetros de Euler, los componentes de un cuaternión. Esto es importante, pues en algunos procesos es necesario expresar la orientación de un cuerpo en términos de cuaterniones teniendo los parámetros de Euler a través esta expresión es posible realizar una conversión sencilla entre estas dos formas diferentes de expresar la actitud de un cuerpo rígido.Ahora, el producto de dos cuaterniones se describe como se muestra en **(7)**,

$$\boldsymbol{q_1} \otimes \boldsymbol{q_2} = \begin{bmatrix} \eta_a - \varepsilon_1 - \varepsilon_2 - \varepsilon_3 \\ \varepsilon_1 \eta_a - \varepsilon_3 \varepsilon_2 \\ \varepsilon_2 \varepsilon_3 \eta_a - \varepsilon_1 \\ \varepsilon_3 - \varepsilon_2 \varepsilon_1 \eta_a \end{bmatrix}_a \begin{bmatrix} \eta_b \\ \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \end{bmatrix}_b \tag{3.7}$$

Se puede expresar como:

$$\boldsymbol{q_1} \otimes \boldsymbol{q_2} = \begin{bmatrix} \eta_a \eta_b - \varepsilon_a^T \varepsilon_b \\ \eta_a \varepsilon_b + \eta_b \varepsilon_a + \varepsilon_a \times \varepsilon_b \end{bmatrix} \tag{3.8}$$

Una vez que han sido obtenidas las ecuaciones necesarias para la representación de la orientación, con base en los conceptos de cuaternión y ángulo de Euler y sus generalidades *grosso modo* en cuanto a la representación de un cuerpo rígido en el espacio y ,además, mencionado las ventajas y practicidad que cada una de ellas presenta, en los apartados siguientes se continúa con la teoría de la cinemática y dinámica del cuerpo rígido, lo cual permitirá, más adelante, el planteamiento del modelo dinámico de la MSA, el cual será de alta utilidad para describir y entender, por medio de cantidades dinámicas, el principio de funcionamiento de la plataforma de simulación satelital.

3.3 Introducción a la cinemática del cuerpo rígido

La cinemática estudia el movimiento de una partícula o un sistema de partículas, relativo a la posición, la velocidad, la aceleración y el tiempo, sin tener en cuenta las fuerzas externas que originaron el movimiento. A partir de la conceptualización de la MSA como un cuerpo rígido en el espacio, abordado en apartados anteriores, se hará una introducción breve de la cinemática del cuerpo rígido tomando como punto de partida la descripción del movimiento de rotación.

El referente principal que permite abordar la cinemática de un cuerpo rígido la podemos encontrar en el teorema de Chasles (1793-1880). Dicho teorema establece que el movimiento de un cuerpo rígido puede ser descrito por medio del desplazamiento de cualquier punto del cuerpo (un punto base) más una rotación en torno a un único eje a través de ese punto, es decir, que el movimiento en general de un cuerpo rígido es el vector que se obtiene como resultado de una traslación y una rotación puras. Por lo tanto, en cualquier instante un cuerpo rígido en estado general de movimiento tiene una vector de velocidad angular cuya dirección es aquella del eje instantáneo de rotación.

En la Figura 3.7, se puede observar un cuerpo rígido en movimiento y su eje instantáneo de rotación, en dicha figura se define la dirección del vector de velocidad angular absoluta, que denotaremos con $\boldsymbol{\omega}$. Los ejes **X, Y, Z** son fijos e integran el marco de referencia inercial de referencia. Los vectores de posición $\boldsymbol{R_A}$ **y** $\boldsymbol{R_B}$ de dos puntos del cuerpo rígido son medidos con respecto al marco de referencia inercial.

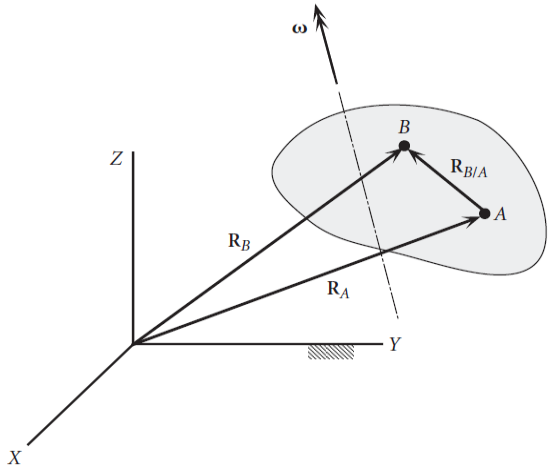


Figura 3.7 Cuerpo rígido y sus ejes instantáneos de rotación.

Ahora bien, si se tiene un vector $\boldsymbol{R_{B/A}}$ que como se observa en la misma Figura 3.2.1 que va del punto **A** al punto **B** es el vector de posición de **B** relativo a **A**. Ya que el cuerpo es rígido $\boldsymbol{R_{B/A}}$ tiene una magnitud constante aun cuando su dirección está continuamente cambiando. Se observa que:

$$\boldsymbol{R_B = R_A + R_{B/A}} \qquad \textbf{(3.9)}$$

Derivando la ecuación **(9)** con respecto al tiempo, se tiene:

$$\dot{\boldsymbol{R}}_{\boldsymbol{B}} = \dot{\boldsymbol{R}}_{\boldsymbol{A}} + \frac{d\boldsymbol{R}_{\boldsymbol{B/A}}}{dt} \qquad \textbf{(3.10)}$$

Los vectores $\boldsymbol{R_A}$ y $\boldsymbol{R_B}$ son las velocidades absolutas $\mathbf{v_A}$ y $\mathbf{v_B}$ de los puntos **A** y **B**. Debido a que la magnitud de $\boldsymbol{R_{B/A}}$ no cambia, su derivada con respecto al tiempo está dada por la siguiente ecuación:

$$\frac{d\boldsymbol{R}_{\boldsymbol{B/A}}}{dt} = \boldsymbol{\omega} \times \boldsymbol{R}_{\boldsymbol{B/A}} \qquad \textbf{(3.11)}$$

Por lo que la ecuación **(3.11)**, correspondiente a la velocidad relativa queda del modo siguiente:

$$\boldsymbol{V_B} = \boldsymbol{V_A} + \boldsymbol{\omega} \times \boldsymbol{R_{B/A}} \qquad \textbf{(3.12)}$$

Además, si se obtiene la aceleración correspondiente con una segunda derivada, quedaría $\ddot{\boldsymbol{R}}_{\boldsymbol{A}}$ y $\ddot{\boldsymbol{R}}_{\boldsymbol{B}}$ como las aceleraciones absolutas respectivas a $\boldsymbol{a_A}$ y $\boldsymbol{a_B}$ de dos puntos del cuerpo rígido [43], quedando la siguiente ecuación:

$$\frac{d^2\boldsymbol{R}_{\boldsymbol{B/A}}}{dt^2} = \boldsymbol{\alpha} \times \boldsymbol{R}_{\boldsymbol{B/A}} + \boldsymbol{\omega} \times (\boldsymbol{\omega} \times \boldsymbol{R}_{\boldsymbol{B/A}}) \qquad \textbf{(3.13)}$$

Donde α es la aceleración angular, tal que $\boldsymbol{\alpha} = \boldsymbol{d\omega/dt}$, quedando de la siguiente forma:

$$\boldsymbol{a_B} = \boldsymbol{a_A} + \boldsymbol{\alpha} \times \boldsymbol{R_{B/A}} + \boldsymbol{\omega} \times (\boldsymbol{\omega} \times \boldsymbol{R_{B/A}}) \qquad \textbf{(3.14)}$$

En esta primera parte del estudio del movimiento de un cuerpo rígido, que aborda el movimiento sin considerar a fuerzas externas sobre el cuerpo o sistema de partículas. Lo que sigue, será el estudio del movimiento del cuerpo en rotación, de lo que se deducirá que la dinámica del cuerpo rígido ya considera la cinemática, en este caso, enfocado al movimiento rotacional.

3.4 Introducción a la dinámica del cuerpo rígido

El estudio de la dinámica de un cuerpo considera las causas que provocaron el movimiento del mismo, a diferencia de la cinemática en la que no es así. Continuando en la misma línea de estudio, y siendo el mismo caso que se pretende analizar, se deben considerar algunos elementos adicionales, los cuales serán explicados brevemente.

El movimiento rotacional puede definirse como el movimiento de un cuerpo rígido, en este caso, alrededor de un eje **Z** con una velocidad angular $\boldsymbol{\omega}$, donde cada una de sus partículas describirá una órbita circular con centro en dicho eje. En términos matemáticos, si se considera una partícula $\boldsymbol{A_i}$ que describe un círculo de radio $\boldsymbol{R_i = A_iB_i}$, con velocidad de $\boldsymbol{v_i = \omega \times r_i}$, donde $\boldsymbol{r_i}$ es el centro de masa del cuerpo o el punto de referencia de un sistema inercial, y se considera que la velocidad $\boldsymbol{\omega}$ es la misma para todas las partículas del cuerpo rígido, puede decirse entonces que el momento angular de la partícula $\boldsymbol{A_i}$con respecto al origen es [38]:

$$\boldsymbol{L_i = m_i r_i \times v_i}$$

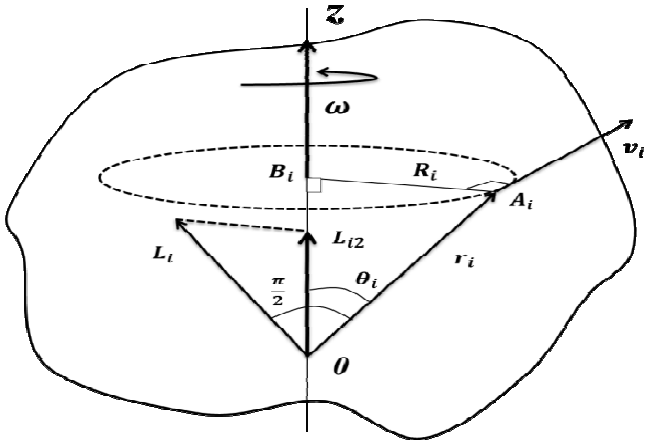


Figura 3.8 Representación geométrica del momento angular.

En la Figura 3.8 puede observarse que la dirección de momento angular $\boldsymbol{L_i}$ es perpendicular al plano trazado por los vectores $\boldsymbol{r_i}$ y $\boldsymbol{v_i}$, y está situado en el plano trazado por $\boldsymbol{r_i}$ y el eje **Z,** dibuja un ángulo $\boldsymbol{\pi/_2 - \theta_i}$ con ese eje, entonces si la magnitud de $\boldsymbol{L_i}$ es $\boldsymbol{m_i r_i v_i}$ y su componente paralela al eje **Z** es $\boldsymbol{L_{iz} = L_i cos(\pi/_2 - \theta_i)}$, puede escribirse como:

$$L_{iz} = m_i(r_i\, sen\theta_i)(\,\omega R_i) = m_i R_i^2 \omega = I\omega \qquad \textbf{(3.15)}$$

De la ecuación anterior puede se puede obtener el momento angular total del cuerpo rodante a lo largo del eje de rotación **Z** como $(\sum_i m_i\, R_i^2)\boldsymbol{\omega}$. Con base en la obtención del momento angular, puede enunciarse también lo que se denomina *momento de inercia* que como puede verse en la siguiente ecuación es parte intrínseca del momento angular, expresado como la suma para cada partícula del producto de su masa por el cuadrado de su distancia al eje:

$$I = \; m_1 R_1^2 + \; m_2 R_2^2 + \; m_3 R_3^2 + \cdots = \; \sum_i m_i\, R_i^2) \qquad \textbf{(3.16)}$$

Esto puede aplicarse para las tres direcciones mutuamente perpendiculares para las cuales el momento angular es paralelo al eje de rotación o ejes de inercia X_1, Y_2, Z_3, y sus momentos se nombran como *momentos principales de inercia*, nominados como I_1, I_2, I_3.

Ahora bien, hay una relación entre el momento angular total de un sistema de partículas y el *torque total* de las fuerzas aplicadas a dicho sistema en cuanto a que éstos se calculan con respecto a un punto de referencia en un sistema inercial:

$$\frac{dL}{dt} = \; \tau \qquad \textbf{(3.17)}$$

Donde τ representa el torque total, tal que: $\tau = \; \sum_i \tau_i$, si esta definición se aplica en el caso de un cuerpo rígido que rota alrededor de un eje principal que contiene un punto fijo en un sistema inercial, si se tiene que $L = \; I\omega$, en consecuencia el torque externo τ debe ser considerado respecto al punto fijo mencionado, entonces:

$$\frac{d(I\omega)}{dt} = \; \tau \qquad \textbf{(3.18)}$$

Pero si el eje permanece fijo con respecto al cuerpo rígido, entonces el momento de inercia será constante, quedando:

$$I\frac{d\omega}{dt} = \tau \qquad \textbf{(3.19)}$$

Con este estudio breve se finaliza la parte teórica de la introducción de la dinámica del cuerpo rígido. Ahora se ingresará a la parte del modelado de la Mesa Suspendida en Aire, tomando en consideración todos los elementos revisados anteriormente para poder desarrollar el análisis del modelo matemático que implica las características y elementos físicos a bordo de la MSA.

3.5 Planteamiento de la cinemática y dinámica de la Mesa Suspendida en Aire: Desarrollo del modelado matemático del sistema, Modelo dinámico

El modelo dinámico de la MSA tiene como objetivo describir mediante una ecuación el movimiento de dicha plataforma, esto considera las causas que provocan su movimiento y los elementos que habrá a bordo, es decir, el análisis implica que la MSA sea considerada como un cuerpo rígido, lo que permitirá simplificar dicho análisis. Antes de comenzar este análisis de movimiento se dará un breve revisión de los componentes de la MSA que se toman en consideración.

El simulador de vuelo satelital es un sistema que contiene a la MSA como plataforma que lleva diversos dispositivos a bordo para realizar maniobra y otras tareas relacionadas; de esa forma está compuesta por un disco de MDF (*Medium Density Fibreboard*) o Tablero de Fibra de Densidad Media de 50 cm de diámetro y 1 cm de altura; cuenta con tres ruedas inerciales (una por cada eje cartesiano), tiene también tres motores de corriente directa (DC), una fuente de alimentación de energía, *drivers* reguladores de voltajes para los motores, un procesador de datos, sensores y un dispositivo transmisor de radio frecuencia (RF). Estos elementos se pueden observar en las Figuras 3.9 y 3.10.

Todos los elementos mencionados están montados sobre la MSA y tienen la característica de representar algunos de los subsistemas esenciales que componen a un satélite de vuelo, lo cual se debe considerar en el desarrollo del modelado matemático de la mesa, ya que la masa posee inherentemente inercia.

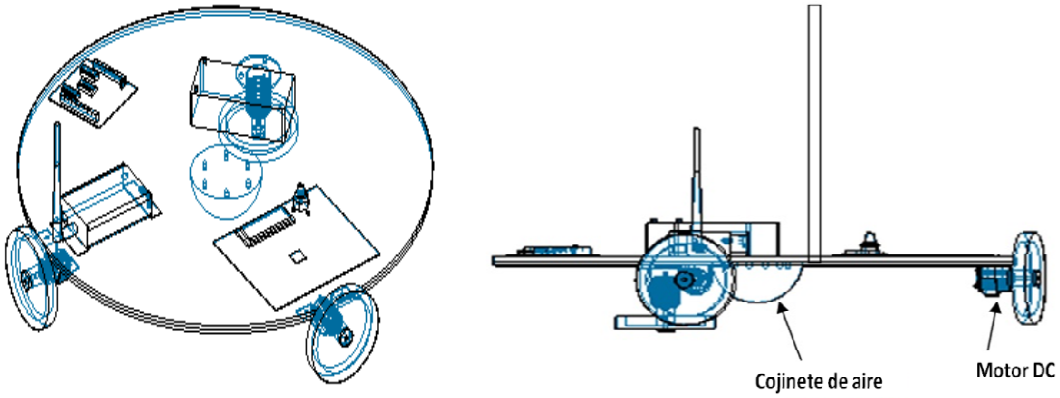


Figura 3.9 y 3.10 Vistas superior y lateral de la MSA y sus componentes.

Como pudo verse en las figuras anteriores, dados los componentes y la forma de la plataforma, se deben caracterizar las partes principales que tendrán movimiento, esto es, la plataforma y las ruedas inerciales; donde, en el caso de estas últimas, puede analizarse una sola vez por simplificación, y extrapolar los resultados con las dos ruedas restantes. Inicialmente se debe conceptualizar vectorialmente este sistema de componentes, de tal forma que pueda establecerse el planteamiento del problema, como lo muestra la siguiente figura:

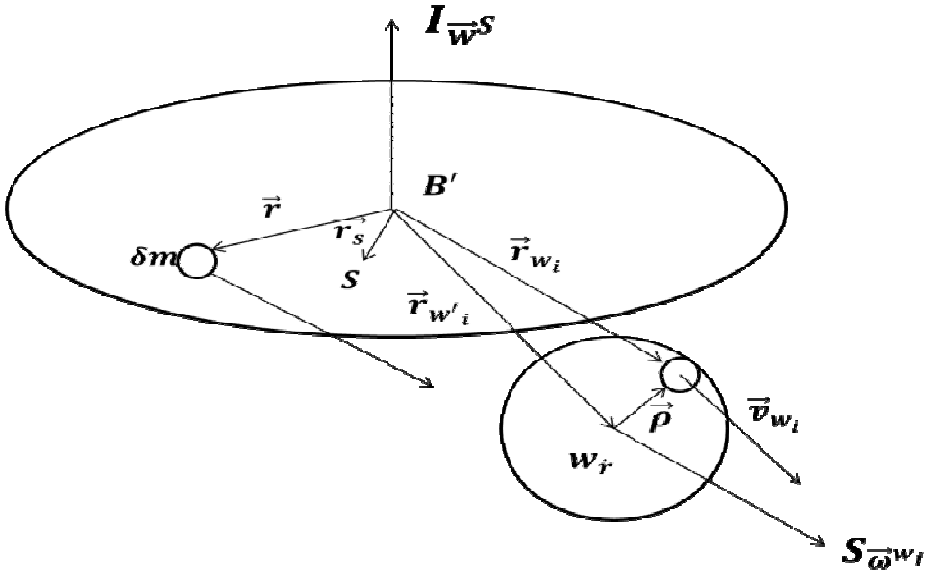


Figura 3.11 Concepto vectorial de las componentes de la MSA.

Donde: **B'** es el centro de masa de la mesa; **S'** es el punto donde está el centro de masa de la plataforma del simulador; **I** es el sistema de referencia inercial y **S** es el sistema de referencia cuerpo.

Por otra parte, hay algunas consideraciones importantes en la Figura 3.11: puede observarse que se modela con solo una rueda inercial, además se toma en cuenta una diferencial de masa $\boldsymbol{\delta m}$, la cual debe ser considerada para el desarrollo matemático posterior; el centro de masa de la mesa no coincide con el centro geométrico de la misma, generando automáticamente un vector de posición **rs** entre ambos; se aprecia también la dirección de la velocidad angular tanto de la mesa como de la rueda, cuya característica es que no corresponde al sistema de coordenadas inercial sino al sistema de coordenadas móvil. Por lo tanto, partiendo del momento de una diferencial de masa $\boldsymbol{\delta m}$, tenemos que su velocidad está dada por:

$$\vec{v} = \vec{\omega}_S^I \times \vec{r} \qquad \textbf{(3.20)}$$

Donde $\vec{r}$ es el vector de posición desde **B'**.

La velocidad de una diferencial de masa **dm** de la rueda está dada por:

$$\overrightarrow{v_{wi}} = \vec{\omega}_S^I \times \overrightarrow{r_{wi}} + \vec{\omega}_{wi}^S \times \vec{\rho} \qquad \textbf{(3.21)}$$

Y dado que es un sistema mecánico rotacional, el modelo matemático resultará de determinar en primera instancia todo el momento angular que afecta a dicho sistema, siendo este análisis el objetivo de la cinemática, el cual es el siguiente:

$$\vec{H} = \int_S \vec{r} \times \left(\vec{\omega}_S^I \times \vec{r}\right)\boldsymbol{\delta m} + \sum_{i=1}^{3} \int_{wi} \overrightarrow{r_{wi}} \times \left(\vec{\omega}_S^I \times \overrightarrow{r_{wi}} + \vec{\omega}_{wi}^S \times \overrightarrow{\rho}\right)\boldsymbol{\delta} \qquad \textbf{(3.22)}$$

En la ecuación **(3.22)** se incluye el símbolo de la sumatoria, representada por la letra sigma, la cual alude el análisis correspondiente a las tres ruedas inerciales o actuadores. La ecuación anterior puede desarrollarse por separado, es decir, se operará cada uno de sus miembros por separado.

En su forma general, se puede escribir como:

$$\vec{H} = \vec{Hm} + \vec{Hr} \quad \textbf{(3.23)}$$

Donde $\vec{Hm}$ es el momento angular de la mesa y $\vec{Hr}$ es el momento angular de la rueda, definidas respectivamente como:

$$\vec{Hm} = \int_S \vec{r} \times \left(\vec{\omega}_S^I \times \vec{r}\right)\delta \quad \textbf{(3.24)}$$

$$\sum_{i=1}^{3} \int_{wi} \overrightarrow{r_{wi}} \times \left(\vec{\omega}_S^I \times \overrightarrow{r_{wi}} + \vec{\omega}_{wi}^S \times \overrightarrow{\rho}\right)\delta \quad \textbf{(3.25)}$$

Desarrollando **(25)**:

$$\vec{Hr} = \sum_{i=1}^{3} \int_{wi} \overrightarrow{r_{wi}} \times \left(\vec{\omega}_S^I \times \overrightarrow{r_{wi}}\right)\delta m + \sum_{i=1}^{3} \int_{wi} \overrightarrow{r_{wi}} \times \left(\vec{\omega}_{wi}^S \times \vec{\rho}\right)\delta m \quad \textbf{(3.26)}$$

Simplificando $\vec{Hr}$ y analizando cada uno de sus miembros por separado, se tiene:

$$\vec{Hr} = \vec{Hr1} + \vec{Hr2} \quad \textbf{(3.27)}$$

$$\vec{Hr1} = \sum_{i=1}^{3} \int_{wi} \overrightarrow{r_{wi}} \times \left(\vec{\omega}_S^I \times \overrightarrow{r_{wi}}\right)\delta m \quad \textbf{(3.28)}$$

$$\vec{Hr2} = \sum_{i=1}^{3} \int_{wi} \overrightarrow{r_{wi}} \times \left(\vec{\omega}_{wi}^S \times \vec{\rho}\right)\delta m \quad \textbf{(3.29)}$$

Considerando la figura conceptualizada en vectores, se aprecia que el vector $\overrightarrow{r_{wi}}$ posee dos componentes, $\overrightarrow{r_{wi}}*$y $\vec{\rho}$. Así, tenemos que:

$$\overrightarrow{r_{wi}} = \overrightarrow{r_{wi}}* + \vec{\rho} \quad \textbf{(3.30)}$$

Sustituyendo **(30)** en **(29)**:

$$\vec{Hr2} = \sum_{i=1}^{3} \int_{wi} (\overrightarrow{r_{wi}^{*}} + \vec{\rho}) \times (\vec{\omega}_{wi}^{S} \times \vec{\rho}) \delta m \quad \textbf{(3.31)}$$

Desarrollando:

$$\vec{Hr2} = \sum_{i=1}^{3} \int_{wi} \overrightarrow{r_{wi}^{*}} \times (\vec{\omega}_{wi}^{S} \times \vec{\rho}) \delta m + \sum_{i=1}^{3} \int_{wi} \vec{\rho} \times (\vec{\omega}_{wi}^{S} \times \vec{\rho}) \delta m \quad \textbf{(3.32)}$$

Como la variable en este caso es $\vec{\rho}$, la ecuación anterior puede escribirse como:

$$\vec{Hr2} = \sum_{i=1}^{3} \int_{wi} \overrightarrow{r_{wi}^{*}} \times (\vec{\omega}_{wi}^{S} \times \int \vec{\rho}) \delta m + \sum_{i=1}^{3} \int_{wi} \vec{\rho} \times (\vec{\omega}_{wi}^{S} \times \vec{\rho}) \delta m \quad \textbf{(3.33)}$$

Por definición, el vector de posición del centro de masa relativo a si mismo es cero, esto es:

$$\int_{wi} \vec{\rho}\, \delta m = 0 \quad \textbf{(3.34)}$$

Lo que se expresa como:

$$\vec{Hr2} = \sum_{i=1}^{3} \int_{wi} \vec{\rho} \times (\vec{\omega}_{wi}^{S} \times \vec{\rho}) \delta m \quad \textbf{(3.35)}$$

Tomando las ecuaciones anteriores **(3.28)** y **(3.29)**, se tiene que ambas integrales son de línea, además de que poseen la propiedad aditiva con respecto al camino de integración, en este caso delimitado por $\boldsymbol{\omega}$. Así, se tiene que:

$$\vec{Hmt} = \vec{Hm} + \vec{Hr1} \quad \textbf{(3.36)}$$

Donde:

- $\vec{\boldsymbol{Hmt}}$ es el momento angular total de la mesa
- $\vec{\boldsymbol{Hm}}$ es el momento angular de la mesa sin considerar la rueda inercial
- $\vec{\boldsymbol{Hr1}}$ es el momento angular de la rueda que influye en el momento angular de la mesa

Por lo que:

$$\vec{\boldsymbol{Hmt}} = \int_{\boldsymbol{S}} \vec{\boldsymbol{r}} \times \left(\vec{\boldsymbol{\omega}}_{\boldsymbol{S}}^{\boldsymbol{I}} \times \vec{\boldsymbol{r}}\right)\boldsymbol{\delta m} + \int_{\boldsymbol{wi}} \overrightarrow{\boldsymbol{r_{wi}}} \times \left(\vec{\boldsymbol{\omega}}_{\boldsymbol{S}}^{\boldsymbol{I}} \times \overrightarrow{\boldsymbol{r_{wi}}}\right)\boldsymbol{\delta m} \quad \textbf{(3.37)}$$

Con la propiedad de *aditividad*, se tiene que:

$$\vec{\boldsymbol{Hmt}} = \int_{\boldsymbol{S+w}} \vec{\boldsymbol{r}} \times \left(\vec{\boldsymbol{\omega}}_{\boldsymbol{S}}^{\boldsymbol{I}} \times \vec{\boldsymbol{r}}\right)\boldsymbol{\delta m} \ldots (\boldsymbol{c}) \quad \textbf{(3.38)}$$

Con estas consideraciones, se puede concretar una ecuación que describa el momento angular total de la mesa.

$$\vec{\boldsymbol{H}} = \int_{\boldsymbol{S+w}} \vec{\boldsymbol{r}} \times \left(\vec{\boldsymbol{\omega}}_{\boldsymbol{S}}^{\boldsymbol{I}} \times \vec{\boldsymbol{r}}\right)\boldsymbol{\delta m} + \sum_{\boldsymbol{i=1}}^{\boldsymbol{3}} \int_{\boldsymbol{wi}} \vec{\boldsymbol{\rho}} \times \left(\vec{\boldsymbol{\omega}}_{\boldsymbol{wi}}^{\boldsymbol{S}} \times \vec{\boldsymbol{\rho}}\right)\boldsymbol{\delta m} \quad \textbf{(3.39)}$$

Hasta este punto, se ha logrado integrar en una sola ecuación todos los momentos angulares involucrados en la rotación de la plataforma satelital. Lo siguiente es descomponer las operaciones vectoriales para obtener un modelo matemático más reducido. Para ello, es conveniente emplear las propiedades vectoriales, a partir del siguiente teorema.

Se tiene el siguiente triple producto vectorial:

$$\boldsymbol{a} = \vec{\boldsymbol{b}} \times \left(\vec{\boldsymbol{c}} \times \vec{\boldsymbol{b}}\right) \quad \textbf{(3.40)}$$

Al descomponerlo, se produce la siguiente expresión:

$$\vec{\boldsymbol{b}} \times \left(\vec{\boldsymbol{c}} \times \vec{\boldsymbol{b}}\right) = \vec{\boldsymbol{c}}\left(\vec{\boldsymbol{b}} \cdot \vec{\boldsymbol{b}}\right) - \vec{\boldsymbol{b}}\left(\vec{\boldsymbol{b}} \cdot \vec{\boldsymbol{c}}\right) \quad \textbf{(3.41)}$$

Aplicando este teorema a la ecuación **(3.39)**, queda:

$$\vec{\boldsymbol{H}} = \int_{\boldsymbol{S+w}} \vec{\boldsymbol{\omega}}_{\boldsymbol{S}}^{\boldsymbol{I}} \left(\vec{\boldsymbol{r}} \cdot \vec{\boldsymbol{r}}\right)\boldsymbol{\delta m} + \sum_{\boldsymbol{i=1}}^{\boldsymbol{3}} \vec{\boldsymbol{\omega}}_{\boldsymbol{wi}}^{\boldsymbol{S}}\left(\vec{\boldsymbol{\rho}} \cdot \vec{\boldsymbol{\rho}}\right)\boldsymbol{\delta m} \quad \textbf{(3.42)}$$

Integrando:

$$\vec{\boldsymbol{H}} = \underline{\boldsymbol{I}}^{\boldsymbol{S+w/B*}} \vec{\boldsymbol{\omega}}_{\boldsymbol{S}}^{\boldsymbol{I}} + \underline{\boldsymbol{I}}^{\boldsymbol{'wi/wi*}} \vec{\boldsymbol{\omega}}_{\boldsymbol{wi}}^{\boldsymbol{S}}(\mathbf{7}) \quad \textbf{(3.43)}$$

Donde $\underline{\boldsymbol{I}}^{\boldsymbol{'wi/wi*}}$ es la diádica de inercia de la i-ésima rueda con respecto al centro de masa de la rueda **wi***. Así mismo, $\underline{\boldsymbol{I}}^{\boldsymbol{S+w/B*}}$ es la diádica de inercia para el cuerpo entero y la rueda con respecto al centro geométrico de la rueda **B'**.

Para más practicidad, se reducirán las expresiones de las diádicas de inercia.

- $\underline{\boldsymbol{I}}^{\boldsymbol{S+w/B*}} = \underline{\boldsymbol{I}}$
- $\underline{\boldsymbol{I}}^{\boldsymbol{'wi/wi*}} = \underline{\boldsymbol{I}}^{\boldsymbol{i}}$

El modelo inicial está completo, sin embargo aún falta un poco más de desarrollo para llegar a tener un modelo matemático en función de pares. Para ello es necesario auxiliarse del Teorema de Euler:

$$\vec{T} = \vec{rs} \times mg\vec{k} \quad \textbf{(3.44)}$$

Donde:

- **m** es la masa del sistema
- **g** es la constante de gravitación
- $\vec{rs}$ es el vector de posición del centro de masa respecto al centro geométrico de la mesa
- $\vec{k}$ es la dirección de la fuerza de gravedad en el eje coordenado inercial
- $\vec{T}$ será el par de perturbación que sufra el sistema

El modelo matemático de la ecuación **(3.44)** tiene la forma del teorema de Euler, aunque hace falta derivarla. No obstante, esta operación de diferenciación corresponde a un concepto denominado *derivada covariante*. Por definición, cuando en lugar de emplear una base fija en todo el dominio de un vector se usan bases móviles, como cuando se emplean coordenadas curvilíneas, la variación total de un vector dependiente del tiempo depende no solo de la variación de componentes como en el caso de la derivada ordenada, sino también el de la variación de la orientación de la base.

$$\frac{\delta}{\delta t}\boldsymbol{a}(\boldsymbol{t}) = \frac{\boldsymbol{d}}{\boldsymbol{dt}}\boldsymbol{a}(\boldsymbol{t}) + \boldsymbol{\omega}(\boldsymbol{t}) \times \boldsymbol{a}(\boldsymbol{t}) \quad \textbf{(3.45)}$$

En términos de momento angular:

$$\overrightarrow{\boldsymbol{Mnet}} = \dot{\boldsymbol{H}} + \dot{\boldsymbol{\omega}} \times \boldsymbol{H} \quad \textbf{(3.46)}$$

En este caso, el modelo matemático emplea tanto una base inercial como una móvil, así que este concepto debe ser tomado en cuenta.

Con este concepto en mente, se deriva la ecuación **(3.44)**:

$$\frac{d}{dt}\vec{H} = \frac{Sd}{dt}\left(\underline{I}\vec{\omega}_S^I\right) + \vec{\omega}_S^I \times \underline{I}\vec{\omega}_S^I + \sum_{I=1}^{3}\left(\frac{wid}{dt}\left(\underline{I^i}\vec{\omega}_{wi}^S\right) + \vec{\omega}_{wi}^I \times \underline{I^i}\vec{\omega}_{wi}^S\right) \ldots (8) \qquad \textbf{(3.47)}$$

Al desarrollar **(3.47)**, se tiene que:

$$\frac{d}{dt}\vec{H} = \vec{T} = \underline{I}\dot{\vec{\omega}}_S^I + \vec{\omega}_S^I \times \underline{I}\vec{\omega}_S^I + \sum_{I=1}^{3}\left(\underline{I^i}\dot{\vec{\omega}}_{wi}^S + \left(\vec{\omega}_S^I + \vec{\omega}_{wi}^S\right) \times \underline{I^i}\vec{\omega}_{wi}^S\right) \ldots (9) \qquad \textbf{(3.48)}$$

Donde:

$$\vec{\omega}_{wi}^I = \vec{\omega}_S^I + \vec{\omega}_{wi}^S \qquad \textbf{(3.49)}$$

Dado que el vector $\vec{\omega}_{wi}^S$ es paralelo a la componente de inercia $\underline{I^i} = I_{11}^{wi}\vec{\imath}\vec{\imath}\vec{\imath}$ de la rueda, el siguiente producto vectorial se reduce a cero:

$$\vec{\omega}_{wi}^S \times \underline{I^i}\vec{\omega}_{wi}^S = \mathbf{0} \qquad \textbf{(3.50)}$$

Con la deducción anterior en cuenta, y sustituyendo en el teorema de Euler la ecuación **(43)**, se tiene que:

$$\vec{rs} \times mg\vec{k} = \underline{I}\dot{\vec{\omega}}_S^I + \vec{\omega}_S^I \times \underline{I}\vec{\omega}_S^I + \sum_{I=1}^{3}\left(\underline{I^i}\dot{\vec{\omega}}_{wi}^S + \vec{\omega}_S^I \times \underline{I^i}\vec{\omega}_{wi}^S\right) \qquad \textbf{(3.51)}$$

Con base en este análisis y aunque aún falta desarrollar parte de la ecuación de movimiento, el objetivo de un análisis como este es poder desarrollar la ecuación completa para mapear en cada uno de los ejes la caracterización correspondiente, y a partir de ello puede realizarse un análisis adicional más exhaustivo para desarrollar esquemas más complejos de estabilidad, obtener variables de estado, pero a pesar de que no está en su forma completa, la ecuación general obtenida puede darse una idea global del movimiento de la MSA.

En el siguiente apartado se expondrán las propuestas para la implementación del sistema de simulación satelital para validación de esquemas de control, en el cual se dan detalles técnicos y lógicos de los elementos que lo componen, de igual forma, se propone una discusión breve para mostrar las ventajas y desventajas de dichas propuestas con el fin de justificar el desarrollo de este trabajo con el fin de acelerar la obtención de resultados y validar los algoritmos propuestos.

Capítulo 4

Análisis de estrategias de integración e implementación del subsistema de control de orientación

Sin lugar a duda, el proceso de implementación en hardware de algoritmos o modelos matemáticos puede llegar a ser una labor de alta complejidad. Una vez que estos algoritmos son funcionales y se validan por medio de simulaciones numéricas o se observan por medio de modelos virtuales, el siguiente paso lógico debería ser su implementación en una plataforma de desarrollo electrónica. Sin embargo, el proceso de transferencia debe contemplar un análisis de los requerimientos que imponen las partes que integran a los algoritmos, en términos de los recursos de cómputo disponibles (hardware y software). Con el advenimiento de nuevas tecnologías en dispositivos electrónicos, tales como los dispositivos lógicos programables FPGA es posible construir sistemas de cómputo a la medida de los requerimientos de las aplicaciones, por lo que resulta una plataforma muy atractiva la flexibilidad que ofrece, en términos de recursos lógicos, para la integración de sistemas de cómputo donde coexisten elementos tanto de software como de hardware. En este capítulo se abordarán los detalles de dos estrategias de diseño e implementación para la integración de un sistema de cómputo dedicado a la solución de algoritmos de control de orientación para un satélite pequeño. Una primera propuesta considera un sistema totalmente integrado sobre la FPGA, donde coexistirían componentes embebidos en hardware y software. La segunda propuesta considera el uso de la técnica de cosimulación denominada *hardware in the loop,* la cual permitirá la rápida implementación física del sistema, acelerando el tiempo de desarrollo y la evaluación de cada componente del sistema de control de orientación.

4.1 Introducción

La fase de de desarrollo del sistema de simulación satelital para validar esquemas de control incluye un análisis breve que exponga las alternativas para la implementación de toda la lógica del sistema. Para ello se proponen dos formas de hacerlo las cuales presentan ventajas y tienen ciertos costos de tiempo y complejidad que con base en el análisis se deben considerar para llevarlas a cabo. Por un lado se tiene la implementación total a bordo de la MSA, lo cual implica que un solo dispositivo se encargará de la funciones lógicas y gestionará todo el procesamiento y tareas para los dispositivos a bordo del simulador, entre otras actividades que pudiese llevar a cabo. Para los fines de este trabajo y como primera aproximación, el sistema se enfoca en las tareas principales: de adquisición de datos de los sensores de navegación, procesamiento y aplicación de algoritmos para la determinación de orientación, filtrado y corrección de señales, generación de comandos de control, escritura de comandos en registros para maniobras por medio de señales PWM.

De igual forma, el planteamiento que se propone en este capítulo incluyó una fase de análisis numérico y de simulación virtual, el cual permitió observar con gran aproximación y evaluar el desempeño de las estrategias de control utilizadas para la realización de maniobras de control de orientación de un satélite, no obstante, estas sólo fueron aproximaciones a la realidad. Esto fue logrado mediante el trabajo de investigación previo [referencia tesis Córdova], el cual fue desarrollado al interior del Grupo de Desarrollo de Sistemas Aeroespaciales del Instituto de Ingeniería de la UNAM (GDSA-IIUNAM), en le que ya se había planteado un esquema de procesamiento de señales y de control de orientación para el nanosatélite HumSat-México, actualmente en fase de desarrollo. En términos generales, dicho esquema de control contempla el flujo de operación que se muestra en la Figura 4.1.

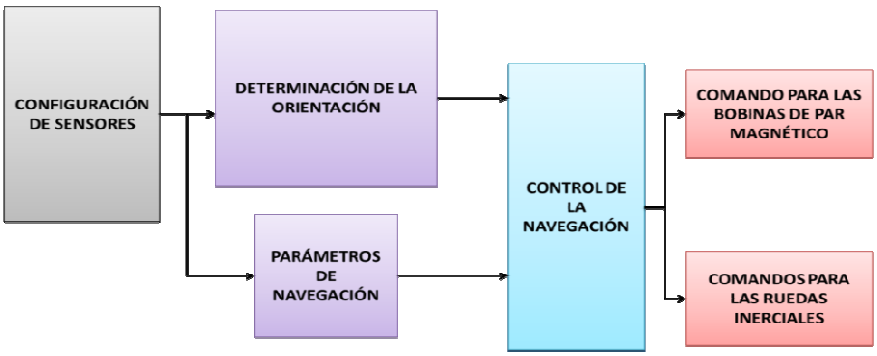


Figura 4.1 Esquema de control del nanosatélite HumSat-México.

En la figura 4.2 es posible identificar algunos de los elementos principales, tales como sensores de navegación, bloques de algoritmos de determinación, estimación y control de orientación, así como actuadores. Ese esquema de control considera el uso de actuadores activos basados en bobinas de par magnético y ruedas inerciales. Como se ha dicho, ese esquema de control fue verificado por medio de modelos de realidad virtual, como se muestra en la Figura 4.2 donde se presenta al satélite HumSat-México orbitando alrededor de la Tierra en una órbita tipo LEO (*Low Earth Orbit*) y realizando tareas de percepción remota.

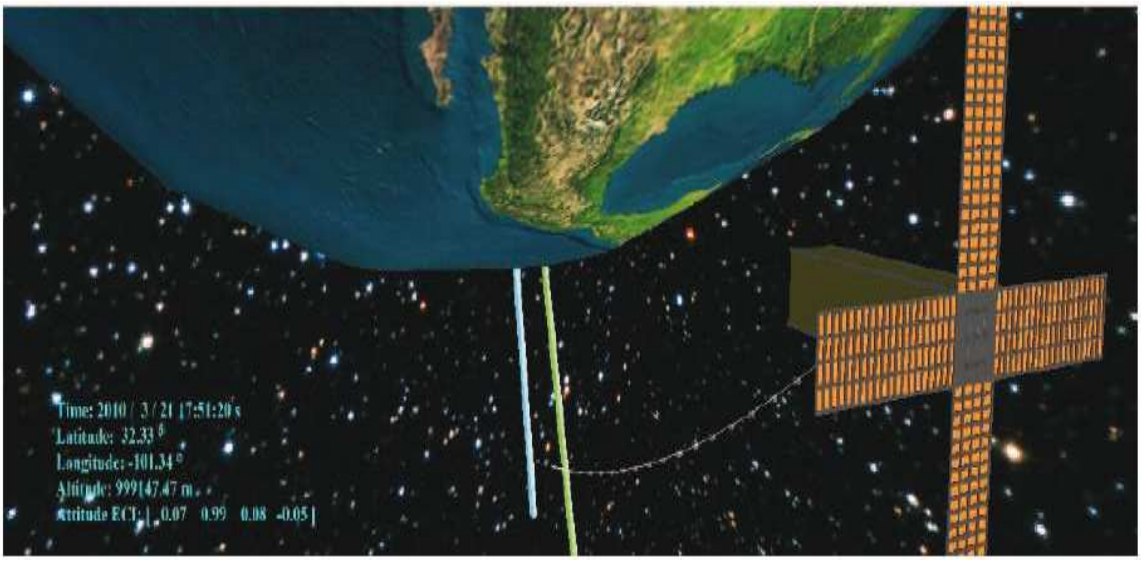


Figura 4.2 Modelo en realidad virtual del satélite HumSat-México.

Este modelo ha sido la base para el presente trabajo, el cual ha integrado esto y otros elementos para poder plantear un sistema físico, que simule con gran aproximación condiciones similares a las del espacio para validar estos esquemas propuestos. Es decir, el poder realizar en tierra una implementación práctica de esos algoritmos y poder reproducir en un ambiente simulado en el laboratorio el desempeño de tales algoritmos se vuelve una enorme y compleja labor de ingeniería que requiere de varias áreas de la misma y que representa una de las principales contribuciones de este trabajo de tesis, y que se expone a continuación.

4.2 Estrategias de implementación de algoritmos de control de orientación para satélites pequeños.

Antes de abordar en detalle cada uno de los esquemas propuestos de implementación del sistema de determinación de la orientación y control, se ofrece una revisión general de cada una de ellas, con el objetivo de tener un panorama claro al momento de realizar el repartimiento de los bloques funcionales y las diferencias en la forma de plantear la solución para atacar un solo problema. Se expondrán las ventajas y desventajas de cada una de estas técnicas de trabajo e implementación y algunas de las consideraciones que se deberán tener en cuenta para desarrollar el sistema a partir de una de ellas.

El primer esquema que se propone es que el sistema lógico sea completamente embebido sobre una plataforma de desarrollo FPGA, la cual irá a bordo de la MSA. Con el uso aquel dispositivo, se podrán explotar características particulares como el paralelismo y la flexibilidad de diseño en el planteamiento de la arquitectura de cómputo. Lo que se analiza en esta propuesta es la repartición de bloques que serán desarrollados como bloques de hardware los cuales implican mayor tiempo de desarrollo y complejidad, pero que son más veloces en su ejecución, o bloques de software que requieren menor tiempo de implementación, pero son más lentos; los bloque a desarrollar son para la adquisición de datos de los sensores de navegación, los algoritmos de determinación de actitud, filtrado utilizando EKF (*Extended Kalman Filter*), leyes de control y envío de señales de control hacia los actuadores (Figura 4.5), y además, se debe considerar que los recursos en la plataforma con que se cuenta son muy limitados en la capacidad de procesamiento y memoria, y deben optimizarse al máximo para tener un buen desempeño del sistema integrado en su conjunto.

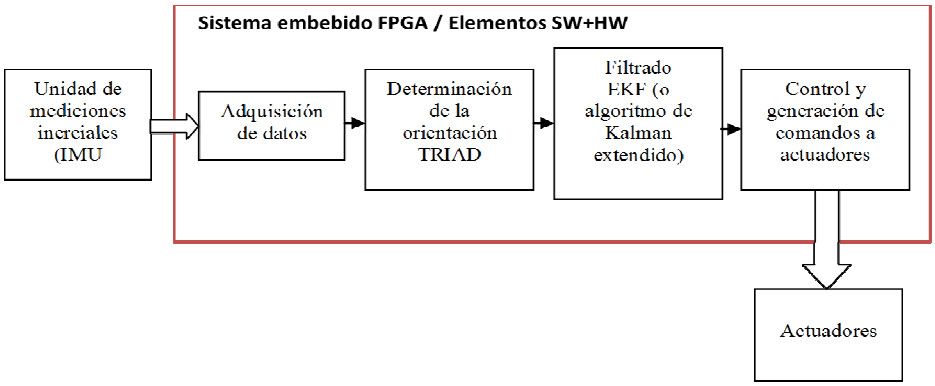


Figura 4.5 Diagrama del sistema contenido en la plataforma de desarrollo FPGA.

La segunda estrategia considera el uso de una técnica de cosimulación conocida como *hardware in the loop* (HIL), en la cual los bloques a implementar son repartidos en plataformas de procesamiento distintas; por un lado se cuenta con una plataforma FPGA donde se encuentra embebida una arquitectura de cómputo que incluye la adquisición de datos y control de actuadores; y por el otro, se implementan *scripts* de MATLAB® sobre un PC donde se realice el procesamiento en tiempo real de los algoritmos para determinar de la orientación, filtrado y estimación y generación de comandos de control (Figura 4.6). Esta técnica es una opción viable desde el punto de vista del tiempo de desarrollo, ya que la reducción en complejidad que conlleva permitiría reducir el tiempo de desarrollo para la obtención de resultados.

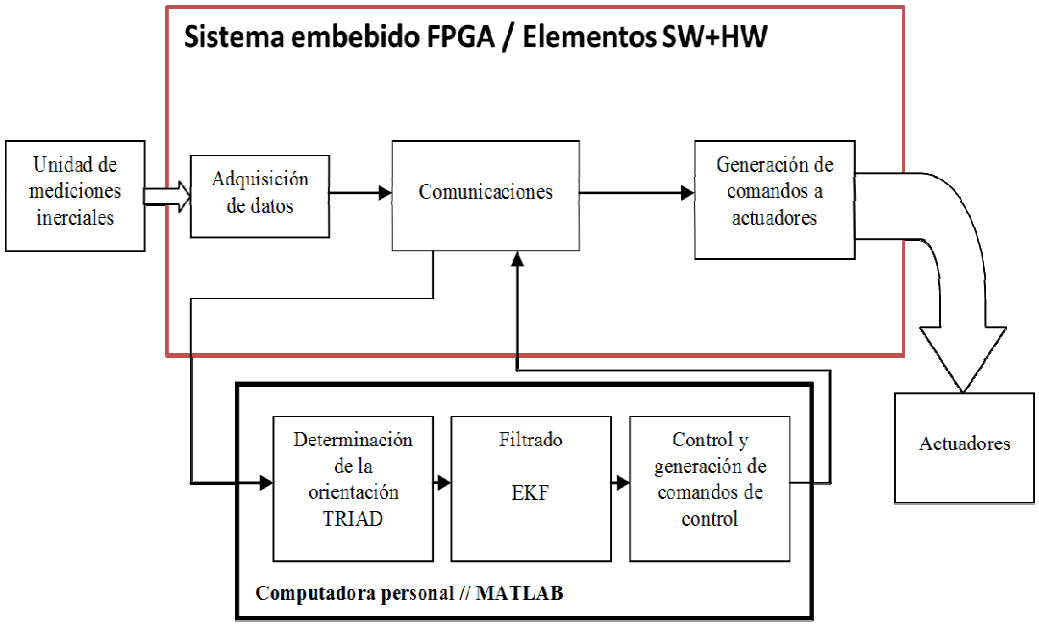


Figura 4.6 Diagrama del sistema utilizando la técnica HIL.

Entonces, de acuerdo a las propuestas, se debe decidir llevar a cabo la integración del sistema mediante una técnica tradicional, optimizando todos los recursos de que se dispone, pero considerando un tiempo más largo por la complejidad que conlleva, o se puede declinar por un desarrollo que permita mayor flexibilidad en la implementación y que pueda generar resultados de gran aproximación a lo que se espera, validando los bloques como si se tratara de un sistema como el de la primera propuesta. A continuación, se describen mayores detalles de las diferentes formas de implementación y sus esquemas de trabajo considerandos los recursos disponibles de igual forma en ambos casos.

4.3 Sistema totalmente contenido en el FPGA

Aunque se han mencionado varias ventajas de los desarrollos de arquitecturas sobre plataformas FPGA, en este caso, la arquitectura que se pretende desarrollar implica muchos bloques que deben interaccionar entre sí, se deben tener en cuenta varios aspectos, los cuales incluyen el conocimiento y uso de las aplicaciones y herramientas en software que ofrece el fabricante de FPGA, la alta complejidad en la implementación de operaciones aritméticas u otras, que no cuenten con soporte por parte de una biblioteca dentro del lenguaje de programación y deban implementarse teniendo el número de parámetros para operar, *depuración*, entre otras desventajas, como la cantidad y el tipo de operaciones aritméticas asociadas con cada uno de los algoritmos a implementar. Para poder enfrentar estos inconvenientes en la implementación, se debe realizar un análisis previo con el objetivo de identificar los bloques de mayor complejidad en cuanto operaciones, funciones de entrada o salida e iteraciones, esto permitirá decidir si los bloques serán de hardware, bloques o *cores IP*, o bloques de software, instrucciones de alto nivel en el cuerpo del programa principal. También es importante la necesidad de adicionar interfaces (y protocolos de comunicación) para la interacción con dispositivos externos. A continuación se hará un análisis de lo requerido y de lo que se dispone para poder lograr con buen éxito el funcionamiento del sistema totalmente contenido en un FPGA.

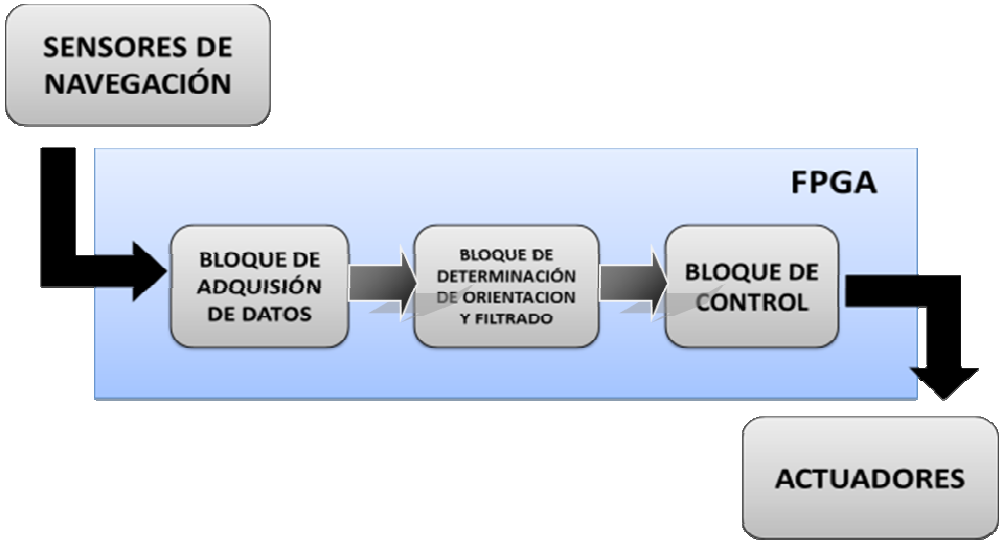


Figura 4.7 Diagrama del sistema incluido en una plataforma de desarrollo FPGA.

La arquitectura de cómputo embebida en la FPGA (Figura 4.7) se compone de tres bloques lógicos principales que serán implementados con base al análisis de repartimiento de bloques: adquisición de datos provenientes de los sensores de navegación, teniendo en cuenta el formato de la lectura de datos de los sensores y los protocolos de comunicación utilizados (I2C, SPI, RS232); el segundo bloque se refiere al procesamiento de los datos a través de algoritmos de determinación de la orientación y filtrado (TRIAD , EKF), de los cuales se obtendrá la determinación de la orientación, con base en este dato se ubicará la posición de la MSA con relación a un sistema de referencia inercial y pasará posteriormente por un algoritmo de filtrado y estimación de señales, el cual permitirá, en principio, mejorar la calidad en las señales de la orientación y posteriormente obtener información sobre la estimación de la velocidad angular de la plataforma en función de los datos obtenidos por el modelo de medición; finalmente, el tercer bloque debe calcular el torque de control en mediante una ley o ecuación de control, que generara las señales que regirán el ciclo de trabajo de los actuadores.

La arquitectura embebida en la plataforma de desarrollo FPGA tiene un núcleo base o principal, se trata de un microprocesador MicroBlaze, que como se mencionó anteriormente, se trata de emular un microprocesador tradicional, pero no es necesario tener una arquitectura de hardware rígida. Este núcleo será el encargado de gestionar y administrar toda la lógica del sistema. MicroBlaze forma parte de la gama media de procesadores embebidos de Xilinx; es un microprocesador *soft core* de propósito general, de arquitectura tipo RISC de 32 bits de características coincidentes para su uso en el sistema embebido que se propone en este apartado. Al formar parte de las herramientas de desarrollo de Xilinx, MicroBlaze cuenta con el soporte necesario para el desarrollo de diseños de mediano y alto nivel de complejidad, utilizando la suite de desarrollo ISE. Un diagrama general de la propuesta totalmente contenida en el FPGA se muestra en la Figura 4.7.

En la Figura 4.8 se muestra la estructura principal de la arquitectura embebida en la FPGA, se resalta el núcleo central, microprocesador MicroBlaze, del cual se desprenden las conexiones principales o buses, y los bloques que gestiona; los elementos externos son los sensores de navegación y las tres ruedas inerciales o actuadores. Los bloques o núcleos periféricos tienen funciones específicas, ya sea para manejo de protocolos de comunicaciones, funciones lógicas o para coprocesamiento aritmético. Los núcleos I2C, FIT e INTERRUPT CONTROLLER (INT) se utilizan para la adquisición de datos y en las comunicaciones; TRIAD y EKF corresponde a la parte de procesamiento, determinación y corrección de la orientación y estimación de la velocidad angular; y los núcleos PWM y CONTROL corresponden al manejo de señales y generación de comandos de control de los actuadores.

Se debe tener en cuenta que algunos núcleos incluidos en la arquitectura, como los que resuelven las comunicaciones (UART, I2C) y los bloques de coprocesamiento (FIT) son núcleos tipo IP (*Intelectual Property*), propiedad de Xilinx, los cuales se ofrecen dentro de la suite ISE. Y los núcleos TRIAD, CONTROL y PWM son núcleos personalizados, es decir, núcleos desarrollados independientemente para funciones específicas. Los núcleos generalmente son programas escritos en lenguaje VHDL, los cuales, después de ser sintetizados y validados por medio de un banco de pruebas (*testbench)* o sobre una plataforma FPGA, son importados al ambiente de desarrollo del sistema embebido para ser instanciados como elementos periféricos del microprocesador MicroBlaze.

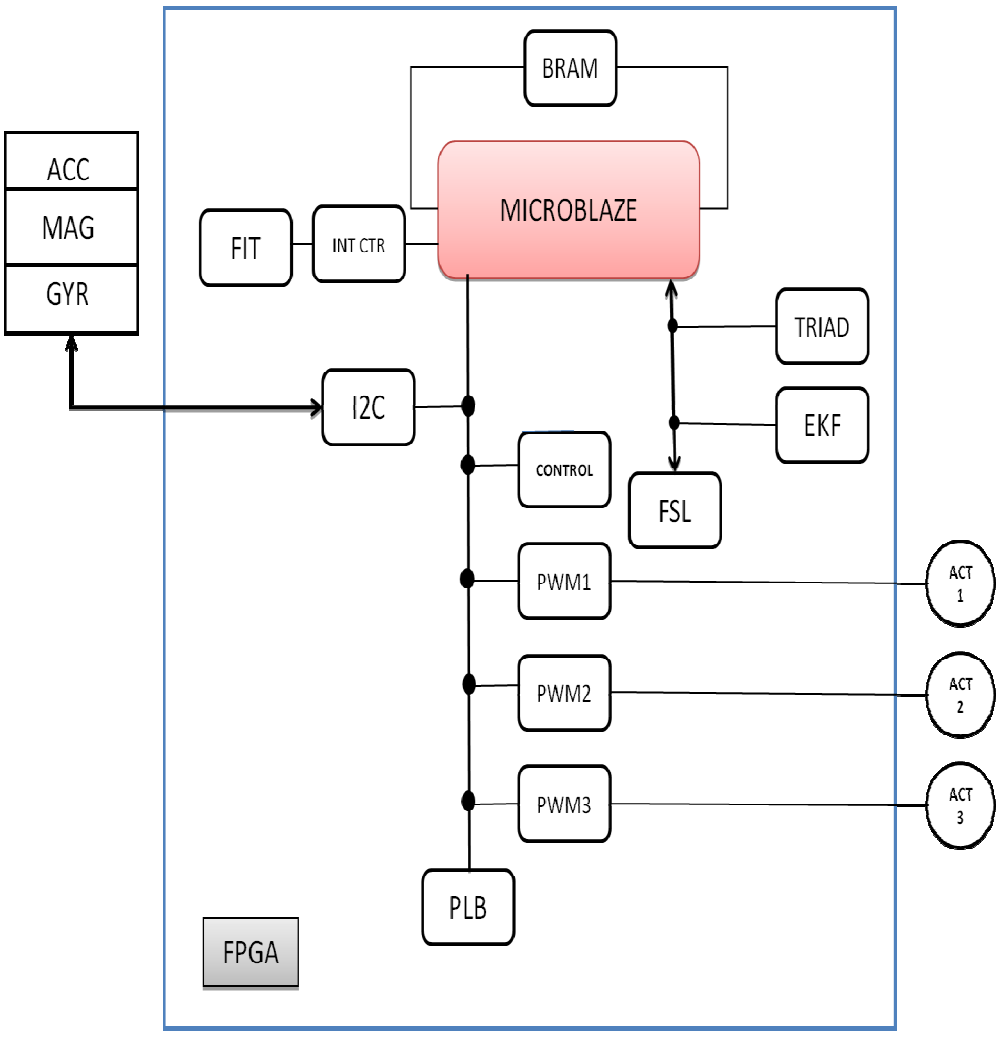


Figura 4.8. Sistema totalmente contenido en la FPGA.

En este sentido es importante destacar la importancia de las herramientas de desarrollo que ofrece el fabricante. Como se ha mencionado anteriormente, para este trabajo de tesis se han utilizado productos de Xilinx, una decisión basada en los recursos de que dispone el GDSA-IIUNAM y del trabajo doctoral del que se desprende esta tesis. El conjunto de software de desarrollo ISE posee dos herramientas importantes: el navegador de proyectos y EDK (*Embed Development Kit*), ambos necesarios para lograr la integración de los elementos del sistema que se expone. El navegador de proyectos (*Project Navigator*) permite por un lado el desarrollo y evaluación de bloques personalizados desarrollados mediante programación VHDL, Verilog o diagramas de estado. Es una herramienta de la que se han mencionado algunas de sus características en el Capítulo 2 de este trabajo, en la figura 4.8 se muestra la interfaz principal que incluye el editor, árbol de proyectos y una consola, y otras herramientas.

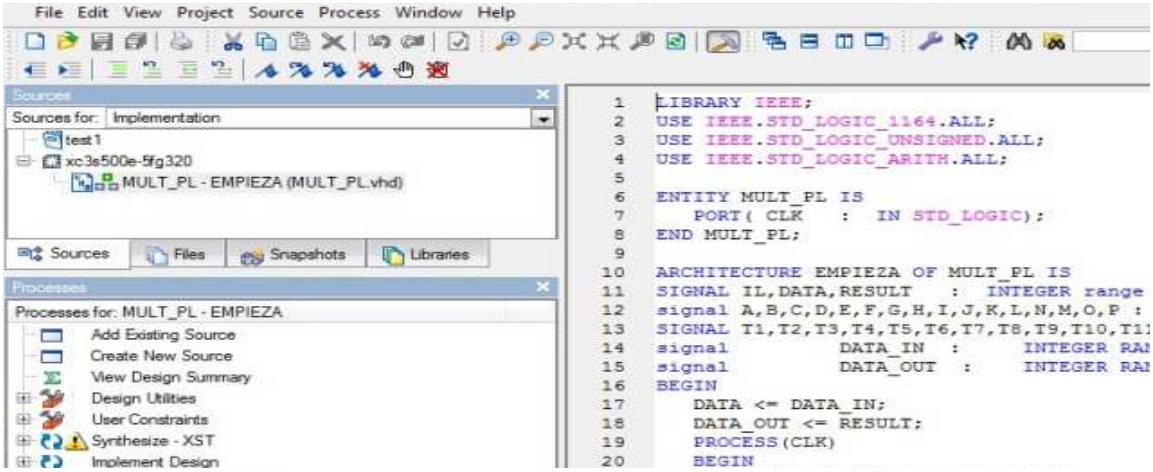


Figura 4.9 Pantalla principal de Project Navigator.

También se cuenta con la herramienta EDK, que es un ambiente de desarrollo que permite la integración de sistemas embebidos basados en las plataformas de dos tipos de procesadores de Xilinx, MicroBlaze y PowerPC, este último de la gama alta de procesadores embebidos de Xilinx. EDK es un ambiente de desarrollo que permite la integración de arquitecturas digitales creadas en VHDL como núcleos periféricos instanciados, y donde se desarrolla el software que controlará la lógica de todo el sistema y que residirá en el microprocesador con soporte para lenguaje C o C++ nativo de Xilinx. La interfaz de desarrollo principal se muestra en la Figura 4.10.

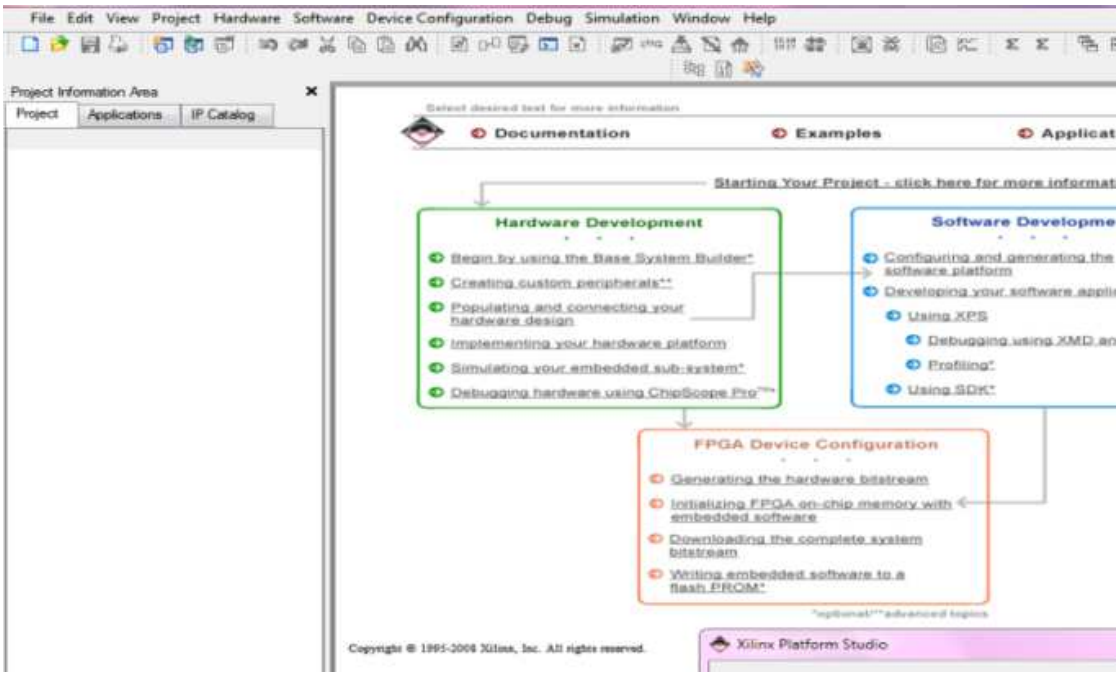
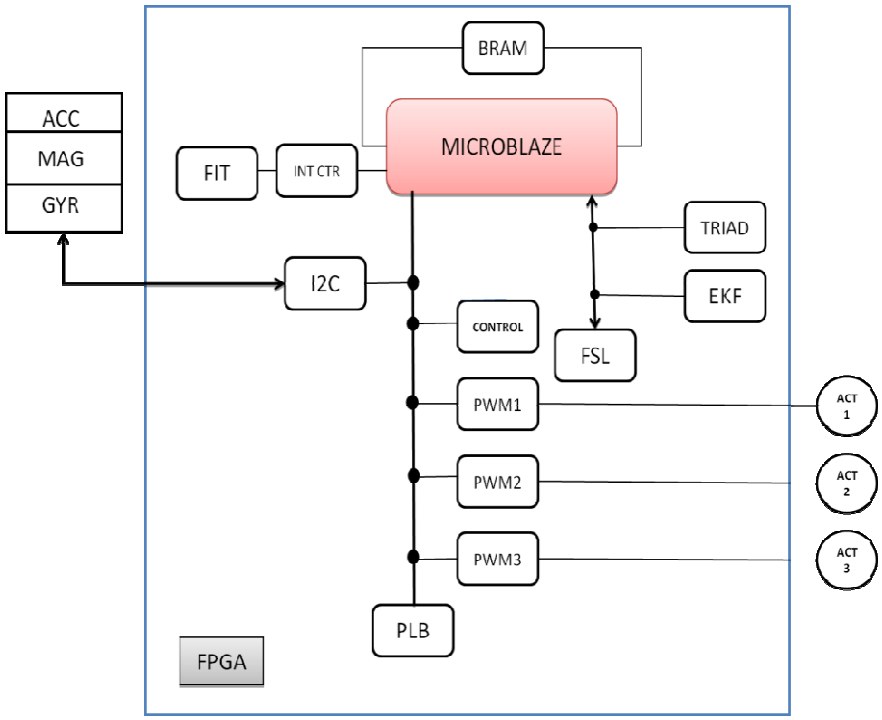


Figura 4.10 Interfaz del software de desarrollo EDK para la creación de sistemas embebidos.

El proceso de desarrollo en EDK es relativamente sencillo, parte de la creación de un proyecto base, el cual es configurado mediante un aplicación *wizard* que simplifica muchas tareas, y donde el usuario integrará cada uno de los elementos de la arquitectura, o bien, se cuenta ya con un sistema base en el que se incluye el núcleo MicroBlaze y se configura la conexión de algunos periféricos básicos. Una vez integrado todo el sistema debe ser sintetizado, de tal forma que el software EDK genere los archivos necesarios para mapear los componentes sobre los recursos del FPGA. Finalmente debe agregarse el programa principal o el firmware del microprocesador embebido, el cual coexistirá con toda la arquitectura base y sus núcleos periféricos, y que también puede llevar a cabo funciones lógicas descritas en lenguaje de alto nivel C y C++. Terminado este proceso se compila el programa principal (software), y posteriormente se realiza la integración del hardware y del firmware del microprocesador MicroBlaze, generando un archivo ***.bit*** que será descargado a la FPGA. Todos los elementos que conforman una arquitectura completa pueden tener cambios y mejoras, ya sean los núcleos u otros elementos lógicos, por lo que pueden ser replanteados, re-sintetizados y vueltos a descargar a la FPGA, tantas veces como sea necesario. En la Figura 4.11 muestra detalle de los módulos que conforman la arquitectura de cómputo, donde se señalan las conexiones, los núcleos o periféricos incluidos y el nombre de cada uno de los módulos.



4.11 Sistema embebido en FPGA y sus módulos.

4.3.1 Adquisición de datos

A continuación se describen algunas de las características de los núcleos periféricos que componen la arquitectura, y comenzará con la exposición de los periféricos para el bloque adquisición de datos, mencionando sus características más relevantes.

- **I2C**: para poder interaccionar con los sensores de navegación es necesario un núcleo para hacer uso de las líneas de comunicación de datos, es decir, manejo de señales por hardware y manejo de registros por software. El bus I2C permite esa interacción de los sensores con la plataforma de desarrollo FPGA a través de las conexiones de entrada y salida de propósito general, asociadas a un núcleo GPIO (núcleo embebido en la arquitectura base que se encarga de las conexiones entre el FPGA y las interfaces de plataforma de desarrollo) que las administra. Está conectado al bus PLB en modo *slave* de 32 bits, puede manejar datos de 32, 64 y 128 bits; se puede operar en uno de los dos modos siguientes: *slave* o *master*; puede habilitarse la generación y detección de señales de INICIO y PARADA [36]. En este trabajo los sensores serán configurados en modo *slave* [39].

- **FIT_TIMER:** antes de generar una interrupción al microprocesador debe haber una señal que indique qué periodo de tiempo o alguna referencia de tiempo para indicar cuándo debe hacerse la interrupción al microprocesador, por lo que debe considerarse agregar un núcleo que pueda generar esta señal de forma automatizada e iterativa, teniendo como base el reloj de la plataforma. Ese núcleo debe ser un temporizador (*timer* de intervalo fijo). Xilinx incorpora este núcleo temporizador listo para agregarse al microprocesador. En él, pueden ser definidos los ciclos de reloj entre interrupciones, basado en el reloj principal de la plataforma de desarrollo FPGA y que servirá como señal de control. También cuenta con reinicio de funciones (*reset*) opcional [40].

- **XPS INT CONTROLLER**: este núcleo podría ser adicional, pues su función principal es gestionar las interrupciones que son generadas, las cuales puede provenir desde 32 fuentes de interrupción diferentes, en el trabajo de tesis presente, sólo se dispondría inicialmente de una fuente proveniente de los sensores de navegación, los cuales vienen incluidos en una sola placa impresa. El controlador de interrupciones tiene prioridad entre peticiones de interrupción determinada por un vector de posición, donde el bit menos significativo (LSB, bit 0) tiene la prioridad más alta [41].

4.3.2 Buses de datos

Como se ha visto, todo el tránsito de datos requiere de canales de comunicaciones internos para que el flujo de datos opere entre los núcleos periféricos y el microprocesador embebido Microblaze. Para ellos se cuenta con dos buses diferentes los cuales se encargan de resolver las comunicaciones internas, por un lado, PLB, el cual ocupa mayor consumo de recursos y es más sencillo de configurar, y por otro, FSL, el cual requiere mayor labor en su implementación y es mucho más rápido.

- **Interfaz PLB**. La interfaz PLB (*Processor Local Bus*) es un bus de datos de 128 bits que cuenta con los recursos necesarios para la conexión de núcleos periféricos en modos *slave* y *master*. Está compuesto por una unidad de control, un temporizador con *watchdog* (mecanismo temporizador auxiliar del sistema cuando ocurre un mal funcionamiento) y un control de registro de dispositivo el cual almacena estados de error. Estas son algunas de sus características más sobresalientes: soporte de buses *master* y *slave* de 128, 64 y 32 bits; segmentación (*pipelining*) de direcciones de PLB [42]. Esta interfaz es un componente muy complejo, por lo que en este trabajo se mencionan aquellas características que sea necesario aclarar para explicar su relación con el desarrollo del sistema propuesto en este trabajo de tesis.

- **BUS FSL**. *Fast Simplex Link* (FSL) es un bus de comunicaciones unidireccional punto a punto disponible sólo para MicroBlaze, con el que se pueden realizar transferencias de datos de forma rápida entre MicroBlaze y un periférico y viceversa. En comparación con el bus PLB, FSL es un bus mucho más rápido, pero requiere de una mayor labor de implementación, ya que no cuenta con un mecanismo incluido que pueda arbitrar las comunicaciones (debe ser programado) y el canal no puede ser compartido. Cuenta con las siguientes características: el tamaño de estructuras FIFOS es de 1 hasta 8 [KB], trabajando sincrónicamente o asincrónicamente, lo que permite que el lado *master* de la interfaz y el lado *slave* puedan operar en velocidades distintas de lo que indica el reloj del sistema. A diferencia de los otros periféricos descritos en este trabajo, el fabricante indica en la hoja de datos del núcleo, que los recursos utilizados por esta interfaz dependerán de la arquitectura y los parámetros de configuración del bus. Este enlace es ideal para comunicaciones entre procesadores MicroBlaze y dispone de puertos suficientes para ocho conexiones FSL de entrada y ocho de salida [43].

4.3.3 Procesamiento

- **NÚCLEO TRIAD.** Para la determinación de la orientación se utilizó un algoritmo determinístico conocido como TRIAD. El nombre se debe a que el algoritmo está basado en la construcción de dos triadas de vectores unitarios ortonormales, usando los vectores de referencia disponibles de cada sensor de navegación. El bloque TRIAD se debe descomponer en elementos más básicos para que su implementación se torne menos compleja. Así, se debe generalizar al algoritmo como un conjunto de cuatro partes fundamentales que lo integran, de las que pueden identificarse el cálculo de vectores ortogonales y de referencia —estos vectores son calculados a partir de realizar algunas operaciones sobre los datos de referencia o teóricos—; cálculo de los vectores ortogonales de medición, correspondiente a la operación sobre los datos obtenidos por los sensores de navegación; cálculo de la matriz de rotación a partir de lo todos los vectores ortogonales en las dos partes anteriores; resultado del cálculo de la matriz de rotación se obtiene, mediante una transformación, un cuaternión correspondiente a la orientación del cuerpo (MSA). El diagrama de flujo del algoritmo TRIAD (Figura 4.11) muestra en términos generales las acciones que deben llevarse a cabo con base en su funcionamiento y etapas principales para facilitar su implementación.

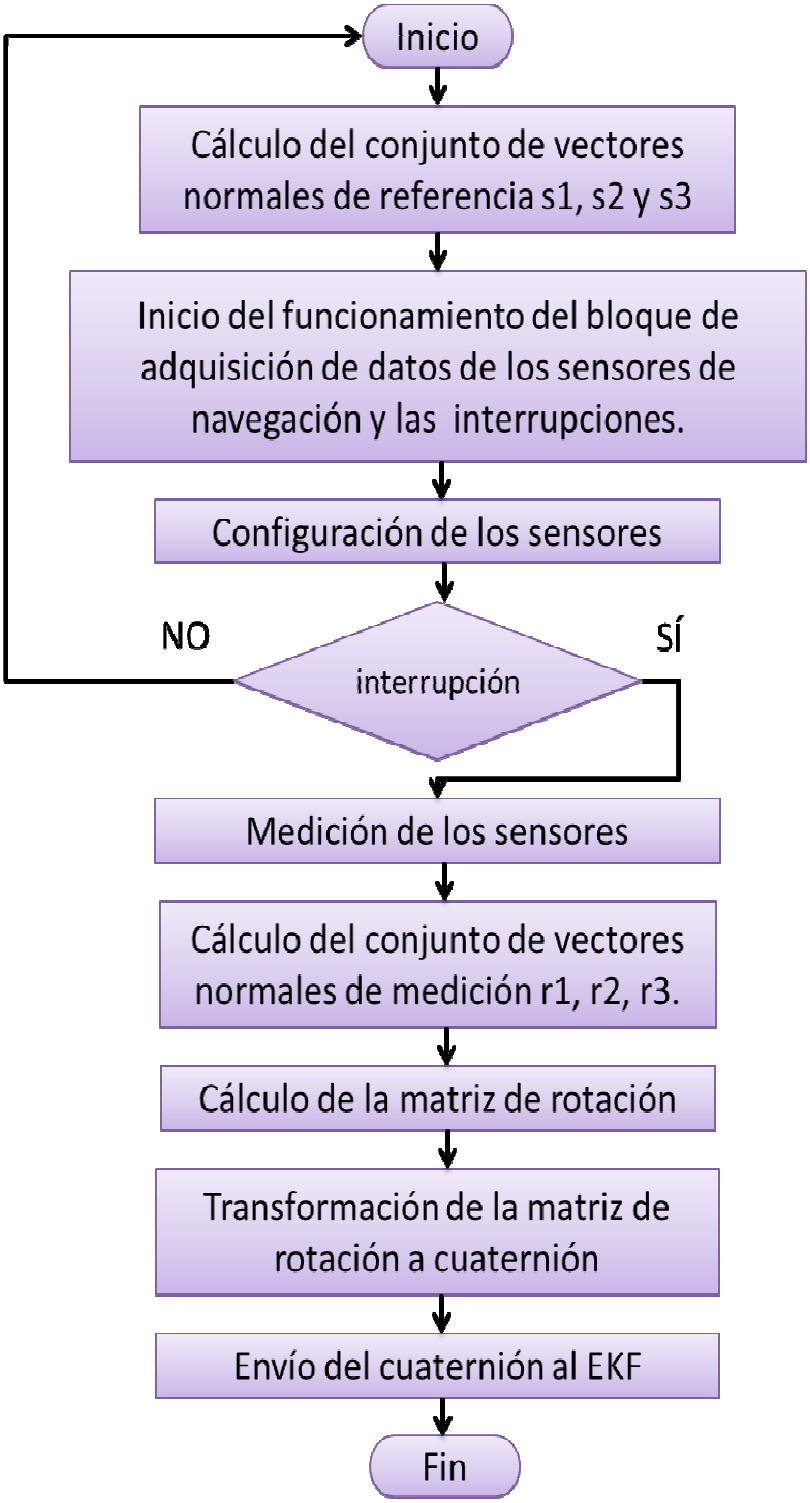


Figura 4.12 Diagrama de flujo del programa para el cálculo del cuaternión de medición a través del método TRIAD.

Por ahora, definiremos al algoritmo TRIAD como un algoritmo para determinar la orientación de un vehículo aeroespacial, la MSA en nuestro caso. Funciona principalmente con las siguientes operaciones: obtención de la norma para el cálculo de vectores unitarios, lo que a su vez se compone de operaciones más básicas con matrices como multiplicación, división, adición y obtención de la raíz cuadrada. Adicionalmente se realiza el cálculo del producto cruz y el cálculo de funciones trigonométricas como senos y cosenos para la conversión de la rotación a cuaternión.

Una vez descritas las operaciones de TRIAD, queda claro que la mejor estrategia para su implementación, antes que pensar en su programación sobre el MicroBlaze, es la creación de bloques de hardware en VHDL que nos permitan la solución de éstos cálculos. Una de las ventajas de este tipo de implementación es la reutilización de núcleos, es decir, será posible utilizar éstos núcleos para resolver operaciones aún pertenecientes a otros algoritmos optimizando recursos y liberando de complejas tareas al procesador central. Otra de las consideraciones importantes en este u otro algoritmo es cuidar la aritmética, ya que dichas operaciones podrían realizarse en punto fijo o punto flotante.

- **Bloque EKF**. Por su parte, el algoritmo EKF (Filtro de Kalman Extendido Discreto), es un algoritmo de estimación que permite determinar el estado instantáneo de un sistema dinámico perturbado por ruido blanco, mediante mediciones linealmente relacionadas con el estado pero afectadas por el ruido. El algoritmo consta de varias etapas, en la que se requiere de diversas operaciones de cierta complejidad como lo son el cálculo de matrices inversas, otras como la suma, resta y multiplicación, y también el cálculo de funciones trascendentales y raíces cuadradas. Lo que se requiere entonces, es realizar una bifurcación de las secciones que comprenden operaciones repetitivas para implementar mediante bloques de hardware sólo aquellos que aún no estén incluidos todavía, esto permitirá reutilizar aquellos bloques de operaciones ya creados y optimizar los recursos de que se dispone. El Filtro de Kalman se compone de 3 secciones (Figura 4.13): 1) Procesos de estimación, innovación y actualización de estados, el cual opera con datos del sensor giroscopio, realimentado por la ganancia del algoritmo y también realimenta el algoritmo TRIAD; 2) cálculo de la matriz de error de covarianzas y etapa de propagación, alimentaba por ganancias teóricas y constantes; 3) cálculo de la ganancia de Kalman. Todo el proceso obtiene un cuaternión equivalente al algoritmo TRIAD y una estimación de la velocidad angular, ambos, elementos necesarios para la ley de control. Hay etapas o fenómenos que no se consideran en la implementación del algoritmo, como el lapso de eclipse, es decir cuando el satélite orbita por una zona obscura.

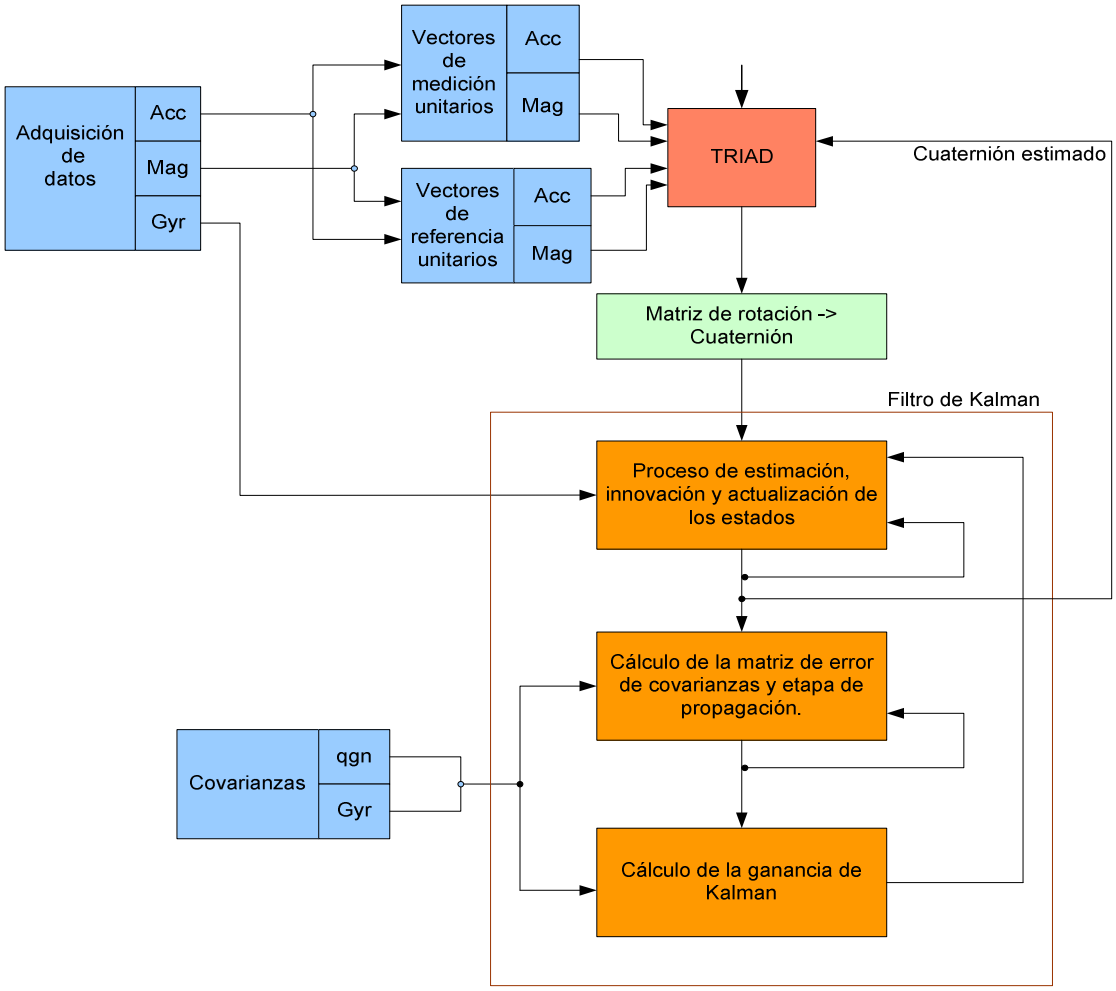


Figura 4.13 Descripción de la composición en bloques del Filtro de Kalman.

- **Núcleos de CONTROL**: El bloque de control se compone de dos secciones principales: la ley de control y la interacción con la lógica de los núcleos PWM para el control de los actuadores. La ley de control que se utilizará está definida por la ecuación 1.

$$\boldsymbol{\tau_c = -k_d \widehat{\Omega_b^i} - k_q \hat{\epsilon}} \qquad \textbf{(1)}$$

De donde $\boldsymbol{\tau_c}$ es el par de control, el cual se calcula a partir de las ganancias $\boldsymbol{k_d}$ y $\boldsymbol{k_q}$ y de los elementos $\boldsymbol{\widehat{\Omega_b^i}}$ y $\boldsymbol{\hat{\epsilon}}$, los cuales representan la velocidad angular y el cuaternión de determinación de la orientación estimados, respectivamente.

Como puede apreciarse, las operaciones que se requieren no tienen mayor complicación, a reserva de que se requieran manejar matrices de dimensiones relativamente grandes que impliquen una alta carga de procesamiento para MicroBlaze, por lo que la lógica funcional del bloque de control podría llevarse a cabo en el microprocesador. Por otro lado, los bloques necesarios para el manejo de los actuadores por medio de PWM, deben ser implementados a través de núcleos personalizados e instanciados al bus de datos del microprocesador embebido, y los cuales tendrían que responder a las instrucciones dadas desde la lógica descrita desde el programa principal (software) esto quiere decir que deben tener un comportamiento como se muestra en el diagrama de flujo de la Figura 4.14.

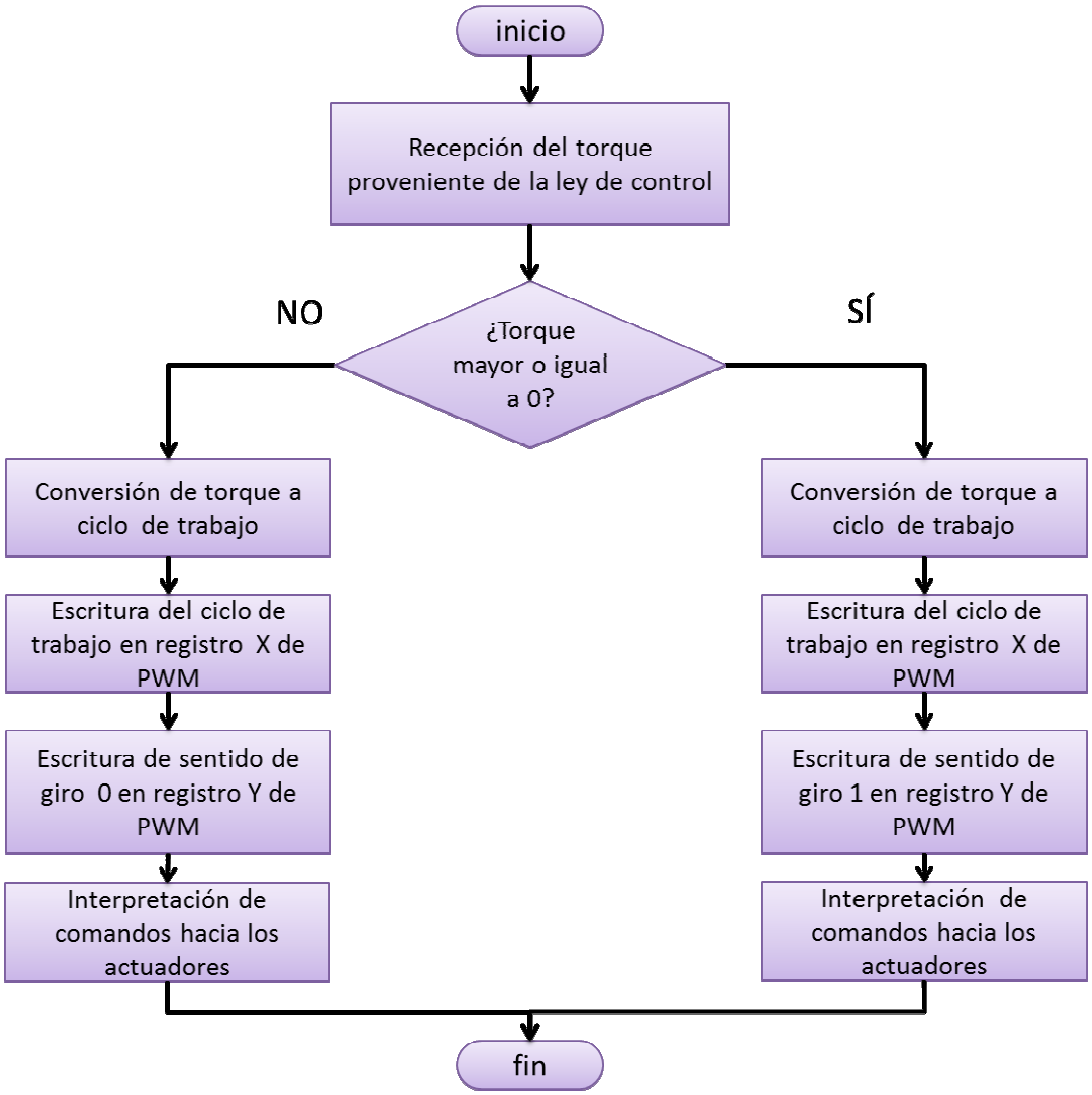


Figura 4.14 Diagrama de flujo de instrucciones para los núcleos PWM.

Los núcleos de PWM serán idénticos pero deben trabajar de acuerdo al parámetro de entrada que es la recepción del torque, la función de conversión que se indica estará en función de las características eléctricas de los actuadores. El signo del torque indica el sentido de giro para los actuadores, por ello se requiere de por lo menos dos registros de trabajo para cada uno de los núcleos, en uno se escribirá el sentido de giro mencionado y en otro el ciclo de trabajo obtenido por la función de conversión.

Cada uno de los algoritmos, tanto TRIAD como EKF, tienen asociados un cierto número de operaciones aritméticas de diferente nivel de complejidad. En la tabla de la Figura 4.15, obtenida de [tesis Paul], se muestra un análisis inicial de la cantidad de operaciones requeridas por cada algoritmo. Esto es de gran importancia, ya que a partir de los datos analizados se realizará la toma de decisiones para llevar a cabo el repartimiento en bloques de software o hardware que integrarían al sistema embebido que se implementará en la FPGA y que contendrá la lógica del subsistema de control de orientación.

Proceso	Sumas	Multiplicaciones	Divisiones	Raíz Cuadrada	Total
TRIAD	70	84	33	10	197
Innovación y actualización	165	152	13	1	331
Propagación	196	1899	6	0	2101
Ganancia de Kalman	0	546	6	0	552
Total	431	2681	58	10	3181

Figura 4.15 Resumen de las operaciones asociadas a los algoritmos TRIAD y EKF.

A continuación, se expondrá una alternativa más versátil para la implementación del sistema propuesto en este trabajo de tesis se trata de una técnica para la bifurcación del sistema que permita desarrollar un banco de pruebas, donde se tenga por un lado adquisición de datos y recepción de los mismos ya procesados y, por otro, el procesamiento, lo que no quiere decir que sea mejor, no obstante ofrece una aceleración para su implementación y obtención de resultados.

4.4 Sistema integrado a partir de la técnica *hardware in the loop* (HIL)

En la primera parte de este capítulo se hizo un análisis de los aspectos más importantes que deberán tenerse en cuenta para llevar a cabo la integración de la parte lógica de la plataforma de simulación satelital MSA y se propuso una arquitectura con la distribución de bloques que deberían estar embebidos en el FPGA; con ello se abordó la complejidad para implementar un sistema como ese con los recursos de hardware y software limitados por las características de la plataforma de desarrollo FPGA. Ahora se presentará una técnica de desarrollo mediante el uso de una técnica de cosimulación llamada *hardware in the loop*, cuyo esquema general se muestra en la figura 4.16. Esta técnica permitirá acelerar el proceso de implementación y conformar un sistema de simulación altamente flexible donde se podrán desarrollar, evaluar y optimizar algunos de los módulos que integran el esquema de control de orientación en dos plataformas diferentes: la plataforma de desarrollo FPGA y una PC.

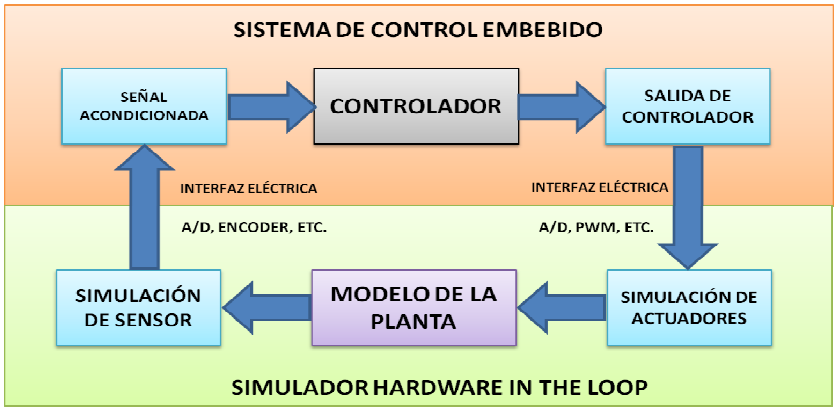


Figura 4.16 Diagrama de bloques del esquema de trabajo en HII.

La técnica HIL es considerada en muchas fuentes bibliográficas como una técnica de simulación en tiempo real, donde el concepto de *tiempo real* implica el tiempo de todo el ciclo de funcionamiento es suficiente mientras este no sea interrumpido, tal que las variables de entrada generan las salidas correspondientes, aún cuando hubiese algún retraso [44]. Esta es la base para implementar la parte lógica del sistema de simulación satelital para validar un esquema de control de la orientación, donde, no hay una simulación de los sensores o actuadores, sino que se utilizaría una PC externa con MATLAB para la ejecución de los algoritmos TRIAD, EKF y CONTROL. A continuación se expondrán los puntos que deben considerarse para llevarlo a cabo.

Con base en la técnica HIL, se tienen dos bloques principales, de los cuales uno representa la implementación física, y otro, la simulación: la plataforma de desarrollo FPGA y la PC, respectivamente. Como muestra el modelo HIL de la Figura 4.17, el sistema de cómputo estará segmentado en dos bloques, la parte de adquisición de datos provenientes de los sensores y el control de los actuadores se resolverá en la plataforma de desarrollo FPGA. El otro bloque, donde se resolverían los algoritmos de determinación de la orientación, estimación y control, serían implementados en un software de análisis matemático como MATLAB ejecutándose sobre una PC. El diagrama de bloques quedaría como se muestra en la Figura 4.17.

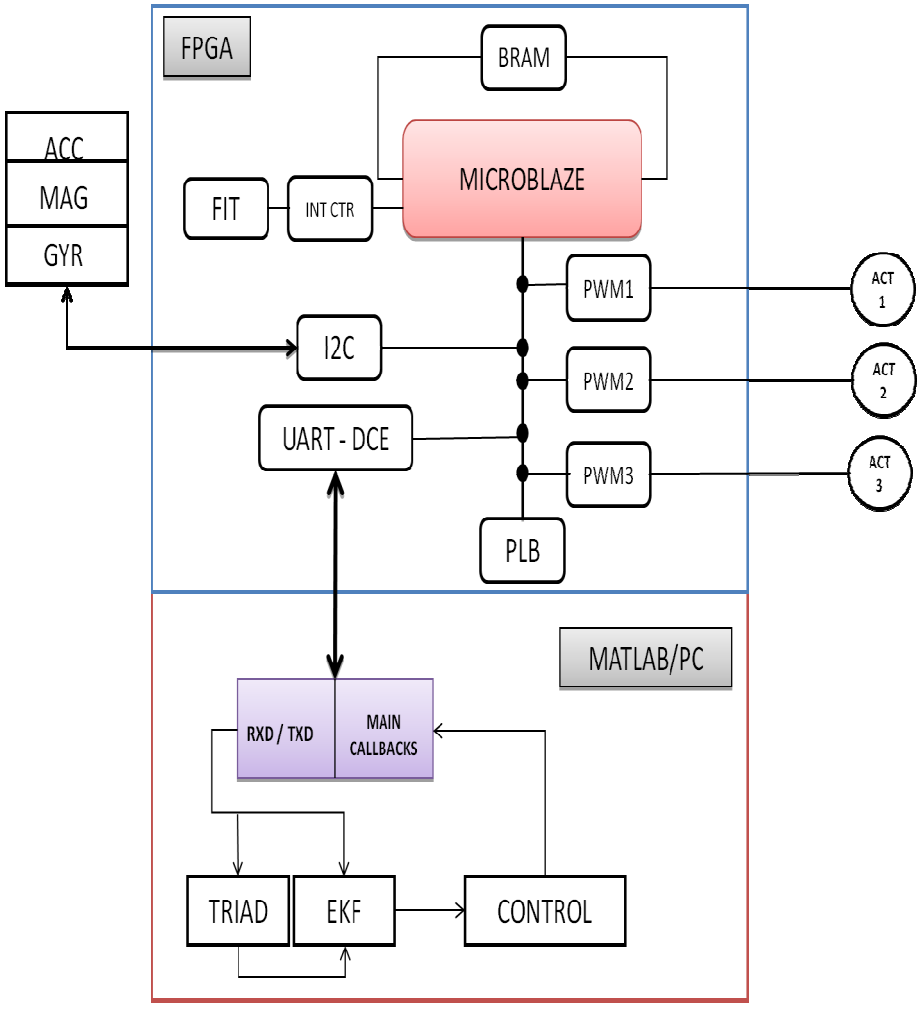


Figura 4.17 Propuesta de esquema HIL para la MSA.

A continuación se dará una revisión de cada uno de los bloques que componen esta propuesta de desarrollo para la implementación del sistema de cómputo del simulador satelital para la validación de esquemas de control.

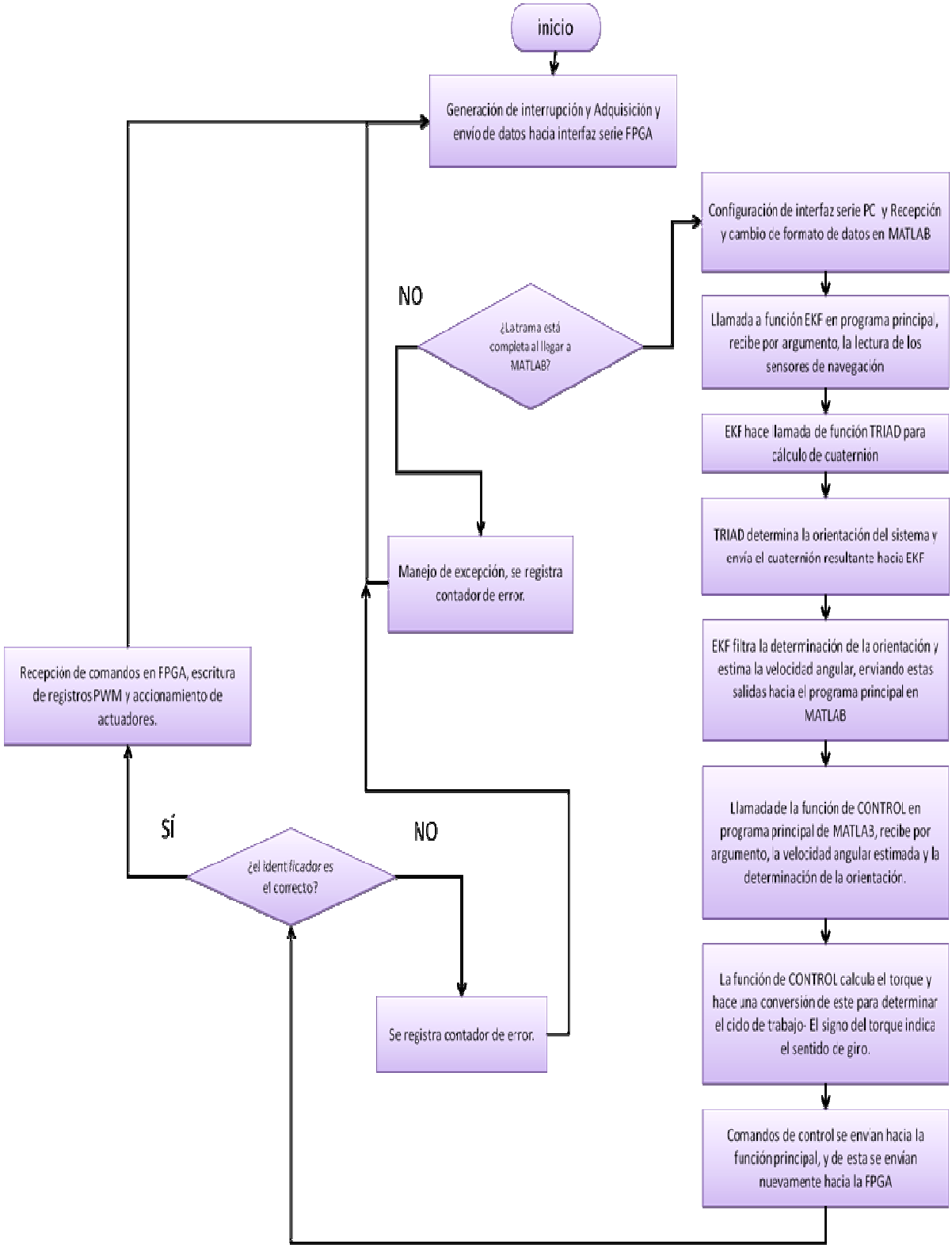
- **BLOQUE DE ADQUISICIÓN DE DATOS E INTERPRETACIÓN DE COMANDO DE CONTROL EN LA PLATAFORMA FPGA.** Al igual que en el bloque propuesto en la alternativa que considera al sistema contenido totalmente en la FPGA, el bloque de adquisición de datos se integraría en el sistema embebido que se construiría en la propia FPGA. Una modificación importante será el envío de los datos de los sensores de navegación hacia la PC, lo cual puede realizarse considerando que el microprocesador embebido tiene la capacidad para enviar los datos a través de una interfaz serial, la cual requiere de un núcleo adicional, UART/DCE, que resolverá los problemas de comunicaciones entre el microprocesador y la interfaz serial de la plataforma FPGA, donde se deberá seleccionar la velocidad de las comunicaciones coincidente con la PC y MATLAB. Por otro lado, siendo que el módulo de control será externo, ahora la plataforma de desarrollo debe resolver el problema de recepción de los comandos para el control de actuadores, e igualmente, esto se realizará por medio de núcleo de UART/DCE, aunque tendrá que definirse una lógica sencilla que permita sincronizar la recepción de los comandos y realizar la escritura de dichos datos en los registros de los núcleos PWM.

- **BLOQUE DE PROCESAMIENTO, ESTIMACIÓN Y FILTRADO, Y CONTROL EN MATLAB.** Con el uso de la técnica de cosimulación HIL uno de los principales cambios en la implementación de la lógica del sistema de control de orientación es el repartimiento de los bloques funcionales del esquema de control y su desarrollo en plataformas distintas. En el caso particular del bloque de procesamiento, en el cual se incluyen los algoritmos de determinación de la orientación, la estimación y el filtrado de parámetros de navegación, así como el desarrollo de la ley de control, los cuales se desarrollarían en MATLAB, que se ejecutaría en una PC convencional. La transferencia de datos entre la plataforma FPGA, abordada en apartados anteriores, y la PC se realizará mediante un enlace de radio frecuencia (RF), con objeto de no entorpecer el funcionamiento de la plataforma MSA. Uno de los primeros retos que representa la implementación de este bloque en MATLAB es resolver la interfaz para la adquisición de datos. La estrategia comenzaría con el manejo del puerto serie de la computadora mediante las funciones correspondientes en MATLAB. Por fortuna, existe mucha documentación al respecto, lo cual puede ayudar a encontrar una solución de forma rápida.

En cuanto a la forma de desarrollar los algoritmos en MATLAB existen dos opciones: una de ellas es hacerlo mediante la integración de scripts (archivos *.m*), los cuales contendrán funciones que describan el funcionamiento de cada algoritmo (TRIAD, EKF, Control). Mediante el uso de esta opción sería relativamente fácil la revisión, en caso de errores de programación, su diseño modular permitiría la conceptualización ordenada de cada uno de los algoritmos y el inherente entendimiento de todo el esquema integrado.

La segunda opción podría considerar el desarrollo de un sistema en SIMULINK, con esta elección se tendría la ventaja de observar mediante el uso de diagrama a bloques la lógica integrada de los algoritmos que integran al esquema de control de orientación. Particularmente para esta opción se tendría el inconveniente para el caso de las comunicaciones por medio del puerto serie, ya que los bloques existentes limitan en gran medida su uso debido a restricciones de uso del propio bloque, aunque existe la posibilidad de desarrollar un bloque personalizado, sin embargo esta opción aún representa un nivel de complejidad mayor que mediante el uso de funciones en scripts.

De forma general: el flujo de datos comenzará por la adquisición de datos de los sensores de navegación en la plataforma de desarrollo FPGA, los cuales deberán ser enviados hacia MATLAB para su posterior procesamiento. Los algoritmos en MATLAB procesan los datos, determinan la orientación del sistema, estiman la velocidad angular y filtran la salida de la representación de la determinación de orientación. Una vez hecho esto, se aplica la ley de control, teniendo como parámetros la determinación de la orientación, ya filtrada del ruido, y la estimación de la velocidad angular, para obtener el torque de control, el cual finalmente es convertirlo en ciclo de trabajo, enviando además el signo de sentido de giro hacia el sistema embebido en la plataforma de desarrollo FPGA sobre la MSA. El sistema embebido se encarga de interpretar los comandos recibidos y escribirlos en los registros correspondientes para la ejecución de los actuadores sobre el sistema físico MSA.



.**Figura 4.18 Diagrama del funcionamiento general del sistema bajo HIL.**

Mediante esta breve exposición y comparación entre las dos alternativas planteadas muestran que la complejidad del primer caso, donde todo el sistema está contenido totalmente en la plataforma FPGA es mucho mayor y además conlleva un mayor tiempo de desarrollo, prueba e implementación. Por otro lado, la alternativa mediante el uso de la técnica HIL ofrece una solución de implementación con un menor tiempo de desarrollo en la implementación, permite realizar pruebas parciales de cada uno de los bloques o módulos para validar su funcionamiento, y además acelera la obtención de resultados, estimando un menor tiempo en el aprendizaje para el uso de herramientas necesarias y adicionales para lograr la integración. Es por estas características que la primera aproximación del sistema de simulación satelital incluirá una arquitectura de cómputo basada en la técnica HIL, la cual se ha explicado brevemente en este capítulo, y posteriormente se detallará la descripción de su composición y funcionamiento.

Los siguientes capítulos se dedicarán a la parte de implementación a detalle de esta propuesta HIL, en ellos se mostrará el desarrollo del sistema y las consideraciones técnicas que hay que realizar, así como la implementación y el desarrollo de cada uno de los bloques de comunicación, procesamiento, envío y recepción de datos, mencionando además los dispositivos involucrados y presentando y discutiendo los resultados obtenidos con los comentarios generales de su funcionamiento.

Capítulo 5

Bloque de adquisición de datos y control de actuadores en el FPGA

En el capítulo anterior se expuso un análisis para elegir un esquema de trabajo que permitiera acelerar la obtención de resultados al mismo tiempo que se validen los algoritmos y funciones que se requieren para la integración del sistema de simulación satelital para la validación de esquemas de control. Con la adopción del esquema basado en las técnicas HIL, estará implementado en una plataforma de desarrollo FPGA, a bordo de la MSA, y una PC externa que realizará tareas de coprocesamiento, las cuales interaccionarán en tiempo real para determinar la orientación de la MSA y llevar a cabo el control de la misma. En el presente capítulo se expone la descripción del primer bloque que realizará los tareas y procesos a bordo de la MSA, es decir, las funciones correspondientes a la adquisición de datos de los sensores de navegación y la generación de comandos hacia los actuadores, así como parte de las comunicaciones. Se detallan los procedimientos, configuraciones, programación y los protocolos de comunicación para lograr el funcionamiento de dicho bloque. Finalmente, se aborda el uso de las herramientas de desarrollo (software) para la integración de un sistema embebido y se abordarán los aspectos más relevantes de la experiencia en el proceso de desarrollo de la arquitectura.

5.1 Introducción

Como se ha descrito en el Capítulo 4, y luego de un análisis en términos del tiempo de desarrollo contra beneficio, se concluyó y justificaron las ventajas para implementar la lógica computacional del sistema de simulación satelital utilizando la técnica HIL. El uso de esta técnica permitirá acelerar la obtención de resultados experimentales, al tiempo que se optimizan y reduce la complejidad en la implementación de la lógica de los algoritmos de control de orientación que se tienen contemplado para esta plataforma de simulación. El sistema estará contenido en dos grandes bloques: la MSA con una plataforma de desarrollo FPGA a bordo y una PC externa. En el primer bloque irá parte de la unidad de cómputo del sistema de simulación (adquisición de datos, comunicaciones, control de actuadores), y el segundo bloque, implementará la lógica de los algoritmos que integran el esquema de control. El diagrama de la propuesta de implementación se muestra en la figura 5.1.

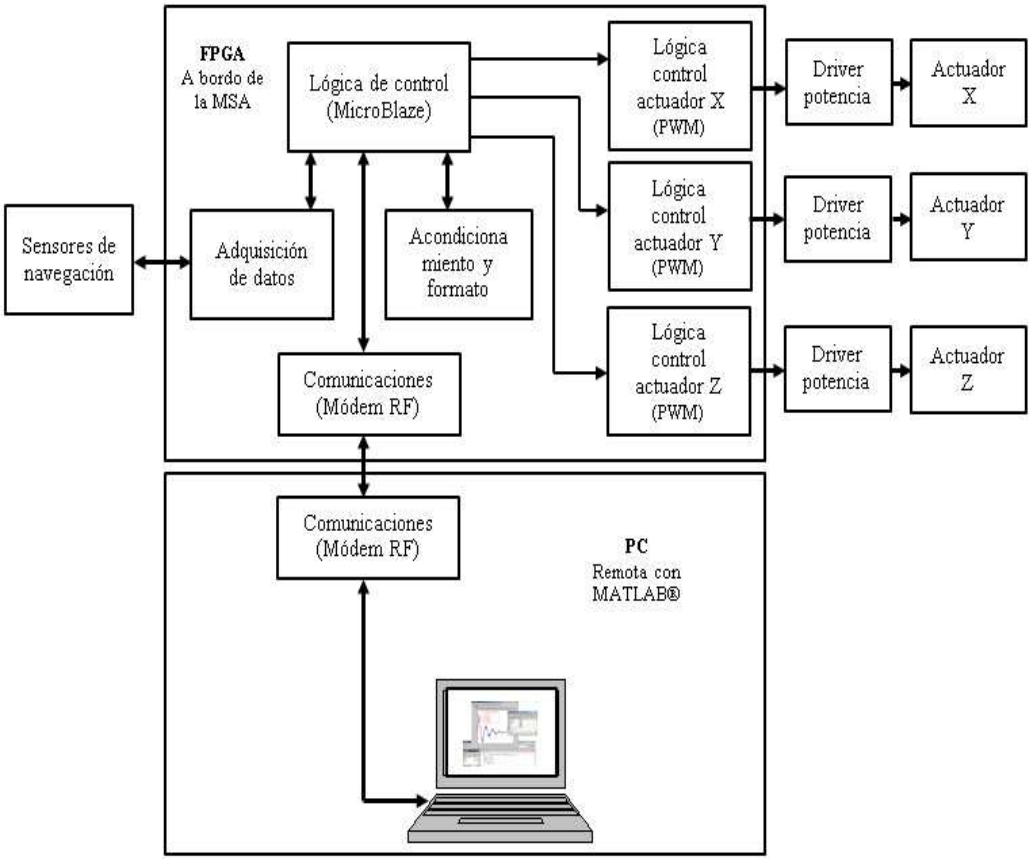


Figura 5.1. Bloques principales del sistema de control de orientación utilizando HIL.

Es importante resaltar el hecho de que uno de los objetivos subyacentes de este trabajo de tesis, previsto desarrollar con el uso de la técnica HIL, además de todas las ventajas ya mencionadas, permitirá el desarrollo de un sistema de simulación simple, de bajo costo, cuya arquitectura integrará elementos de fácil adquisición en el mercado nacional y, lo más importante de todo, representa la base para el desarrollo de tecnología mexicana de estabilización y control de apuntamiento para satélites, materia de alta complejidad y de interés para el desarrollo de misiones satelitales cada vez más sofisticadas y que, en el marco de la creación de la Agencia Espacial Mexicana, representa uno de los primeros esfuerzos tangibles y concretos de desarrollo de tecnología espacial en la UNAM y en México.

5.2 Bloque de adquisición de datos

La adquisición de datos (AD) de los sensores de navegación es el primer bloque del sistema que será implementado sobre la plataforma FPGA SPARTAN 3E. El núcleo principal del sistema es el microprocesador embebido MicroBlaze quien atiende el control de la lógica de todo el sistema a bordo de la MSA y donde se conectan, al bus PLB en este caso, los núcleos para la adquisición de datos. Estos núcleos realizan funciones específicas y generalmente son programados en lenguajes descriptores de hardware (HDL), son sintetizados y agregados al proyecto principal donde son gestionados por medio del microprocesador embebidos.

En el desarrollo de la arquitectura embebida sobre la plataforma de desarrollo FPGA, se han agregado núcleos IP para realizar la adquisición de datos de los sensores de navegación, los cuales utilizan el protocolo I2C para las comunicaciones, entre los que se cuentan: un núcleo generador de interrupciones (FIT), un núcleo controlador de interrupciones (INT CONTROLER), un núcleo para el manejo del protocolo I2C y un núcleo para el manejo de entradas y salidas de propósito general (GPIO). En la figura 5.2 se muestra gráficamente las líneas de conexión entre los núcleos antes mencionados. Aunque se trata de núcleos propiedad del fabricante, su uso requiere una configuración para ser útiles, se deben interconectar a algún bus para su comunicación, otras conexiones como SDA y SCL del núcleo I2C, las cuáles se explicará su función más adelante, y se de deben agregar otros valores para el comportamiento deseado.

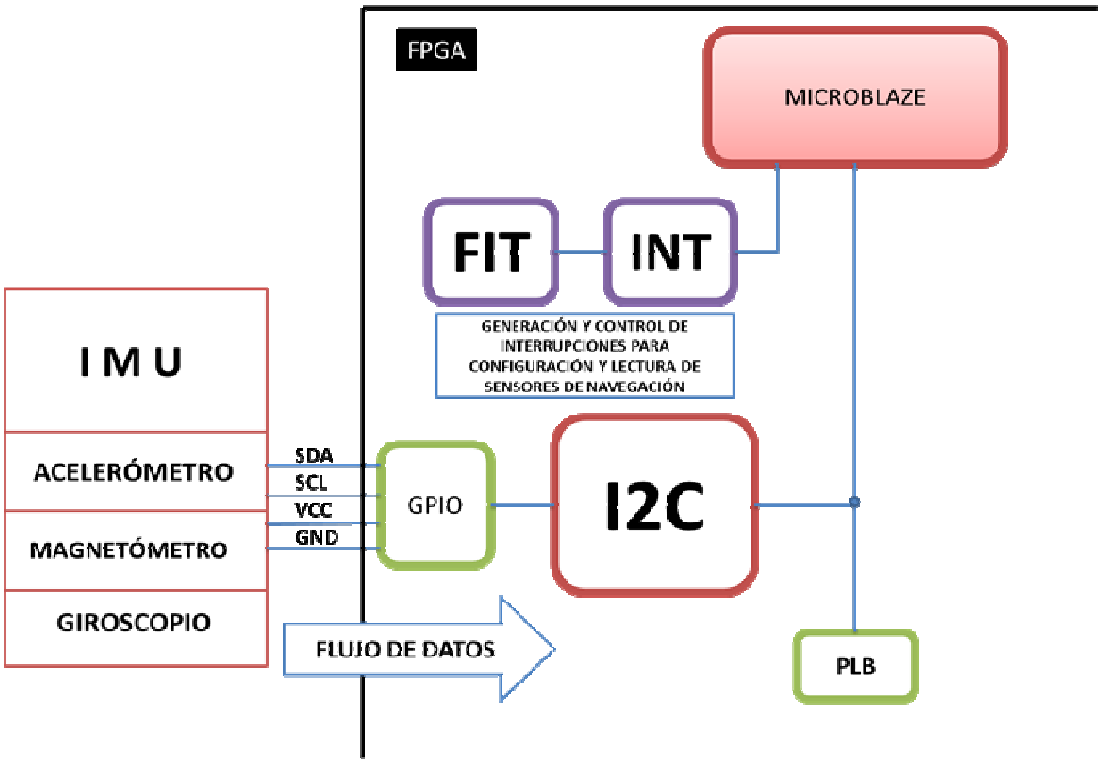


Figura 5.2. Vista detallada de conexiones de componentes del bloque AD.

Una vez obtenidos los datos de los sensores de navegación, estos son enviados a la plataforma FPGA, quien se encarga de almacenar las lecturas y enviarlas nuevamente, por medio de otra interfaz, hacia la PC externa, donde serán procesadas. La parte del empaquetado y comunicaciones se explica más detalle en secciones posteriores. A continuación se exponen las características y función de cada uno de los núcleos que integran el bloque de AD.

5.2.1 Bloque I2C

Este núcleo permite la comunicación para la escritura y lectura con cada uno de los sensores de navegación a bordo de la MSA con MicroBlaze. Es un núcleo que emula el estándar de comunicaciones seriales de alta velocidad *Inter-Integrated Circuit* (I^2C). Se encuentra dentro del catálogo de núcleos disponibles del software EDK, donde se realiza la integración y el desarrollo del sistema embebido para el FPGA.

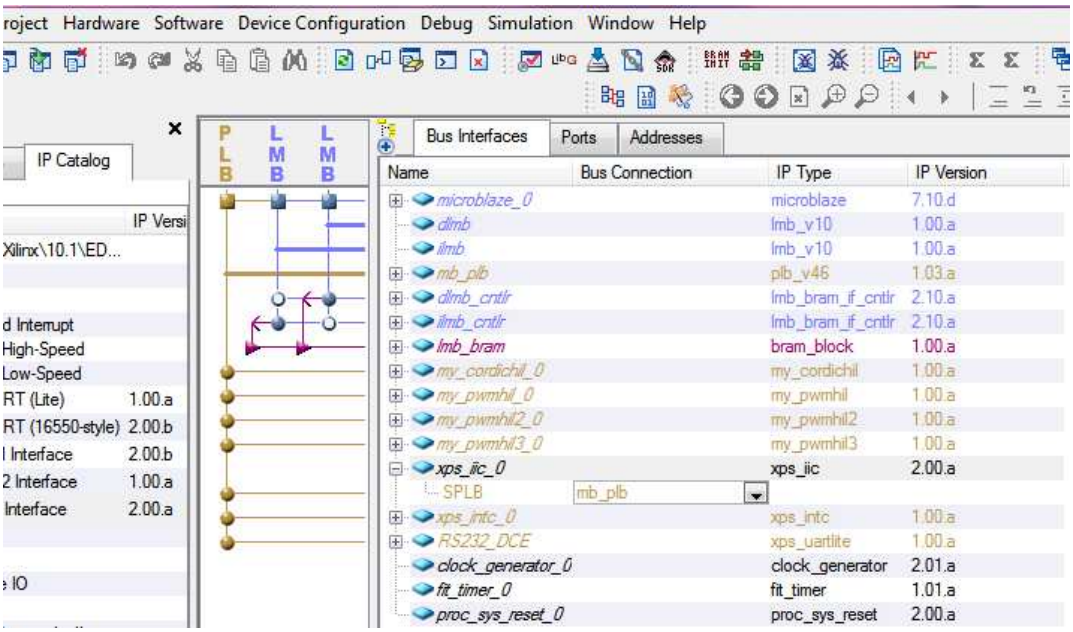


Figura 5.2 Interfaz XPS Platform Studio para la configuración del núcleo I2C, XPS IIC Interface.

Los sensores de navegación magnetómetro, acelerómetro y giróscopo, que se utilizaron en el desarrollo del sistema vienen integrados en una sola tarjeta de circuito impreso denominada IMU (*Inertial Mesuares Unit*), como se observa en la figura 5.3. El protocolo de comunicaciones utilizado por los sensores, I^2C, utiliza dos señales: SDA y SCL, la primera es para transferencia serial de datos y la segunda para la señal de reloj, mediante estas señales se puede gestionar el dispositivo para lectura y escritura de datos. La IMU tiene 9 grados de libertad (*9-DOF*, *9 degrees of freedom*), lo cual quiere decir que cada sensor tienes tres grados de libertad: **X**, **Y** y **Z;** con una resolución en las mediciones de 10, 12 y 16 bits, respectivamente. Adicionalmente, la IMU cuenta con una línea para voltaje y una de tierra, para el suministro de energía de todos los sensores.

El objetivo del bloque I2C es permitir las comunicaciones con los sensores mediante un canal bidireccional que permita al microprocesador embebido MicroBlaze enviar los comandos para la configuración y recibir los datos de medición de los tres sensores integrados en la IMU. La conexión física entre la IMU y la plataforma FPGA, se realiza a través de cuatro líneas provenientes de la IMU y conectadas a las terminales de propósito general de la plataforma desarrollo SPARTAN 3E, configuradas desde el software EDK, se muestra en la figura 5.3.

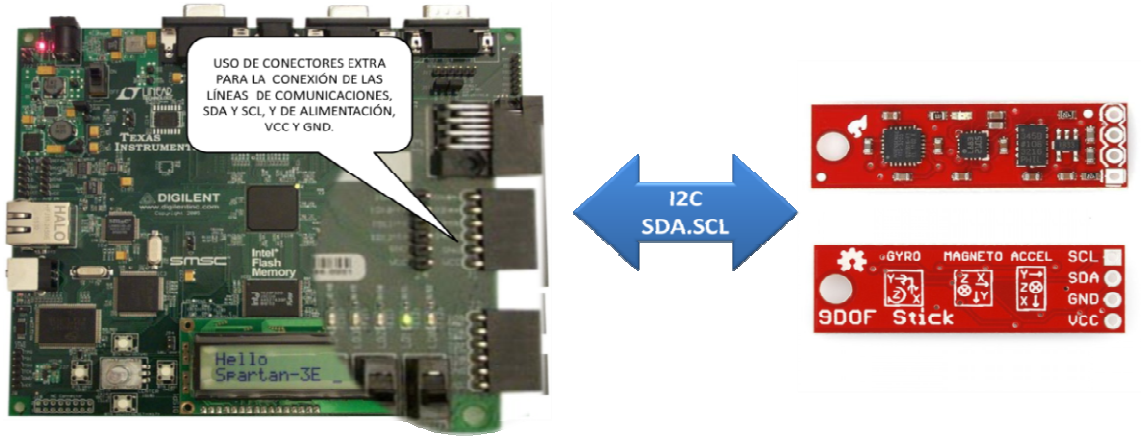


Figura 5.3. Conexión entre la IMU y la SPARTAN 3E.

Además de la integración del bloque para el manejo del protocolo de comunicaciones I2C entre el microprocesador y la IMU, fue necesario realizar la configuración de algunos parámetros de los sensores de navegación para el funcionamiento correcto de estos: en el caso del giróscopo fue necesario configurar la frecuencia de corte de un filtro digital (*pasa bajas*) en 100 kHz [45], lo cual permite atenuar componentes de alta frecuencia del sensor, de igual forma se define la alineación de datos para definir la posición del MSB y LSB dentro del grupo de bits; en cuanto a los dos sensores restantes fue necesario definir el modo de funcionamiento continuo o *single*, en el que se seleccionó el segundo.

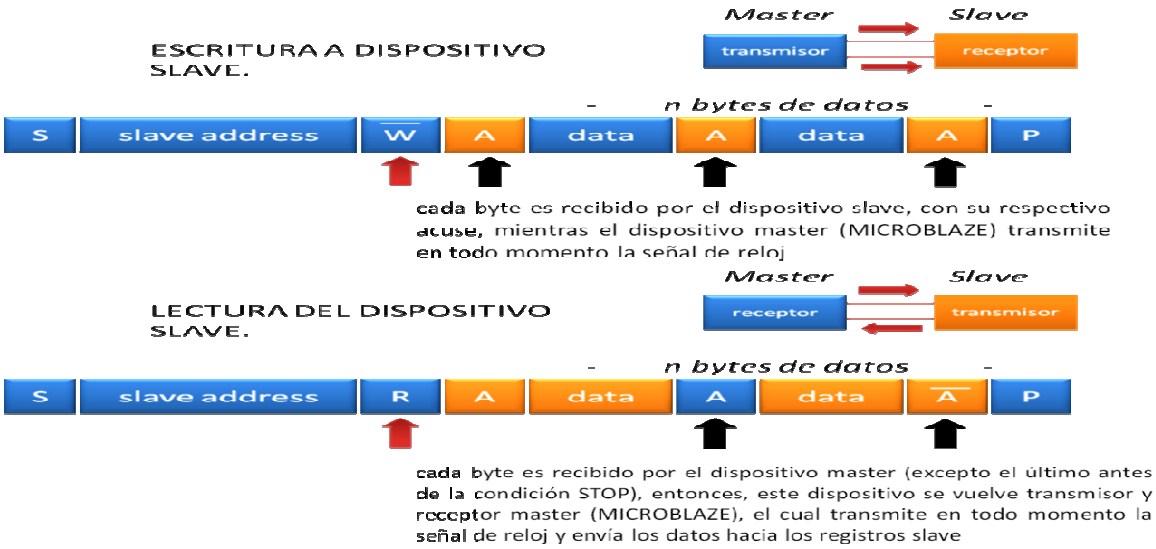


Figura 5.4 Formato de las tramas para el protocolo I2C en los sensores de navegación.

Una vez que ha sido configurado y agregado el núcleo de I2C al proyecto principal por medio del software de desarrollo EDK, es necesario programar en el cuerpo del *firmware* de MicroBlaze la lógica para enviar los comandos de escritura para la configuración inicial de los sensores y obtener las lecturas de datos. Por otra parte, aunque el envío de las tramas de datos de los sensores hacia la plataforma FPGA es transparente en la práctica, a continuación se muestra el formato de lectura y escritura de datos de los sensores de navegación en la Figura 5.4, para ilustrar mejor la forma en que deben sincronizarse los dispositivos por medio del protocolo I2C.

5.2.2 Controlador de interrupciones

El controlador de interrupciones (INT CONTROLLER) es un núcleo que tiene como función principal gestionar las interrupciones generadas a partir de eventos de diversas fuentes, siendo de las más comunes la recepción de datos por la unidad UART o bien, señales producidas por temporizadores que, luego de alcanzar una cuenta máxima definida, generan una señal, la cual puede ser utilizada como un evento fuente de interrupción del núcleo INT CONTROLLER. En el caso de este trabajo se tomó justamente la señal de un temporizador como señal de control del núcleo INT CONTROLLER.

La lógica que obedece la interrupción está basada en el funcionamiento de un temporizador de intervalo fijo (FIT), el cual será abordado en el apartados siguiente. Luego de que el FIT alcanza su cuenta máxima, la cual está definida en términos del tiempo de adquisición de datos del sistema (para este caso 100 ms) [46], se genera una señal, la cual es transferida al INT CONTROLLER y de ahí a las entradas disponibles en el puerto de interrupciones de MicroBlaze.

El núcleo controlador de interrupciones se sincroniza automáticamente para administrar las interrupciones solicitadas al microprocesador. Este núcleo está conectado al bus de periféricos PLB y no requiere de un temporizador adicional ya que funciona de acuerdo a la configuración de la interrupción; esto quiere decir que se inicia con la identificación de flancos de funcionamiento de la interrupción, lo que se denomina detección de interrupciones, ya sea en flanco de subida o bajada. Atenderá la siguiente petición hasta que el bloque haya sido liberado (Figura 5.5). Estricto sentido, este dispositivo es muy complejo, su uso en este trabajo de tesis se remite a su configuración por medio del software EDK, agregado como núcleo periférico de MicroBlaze.

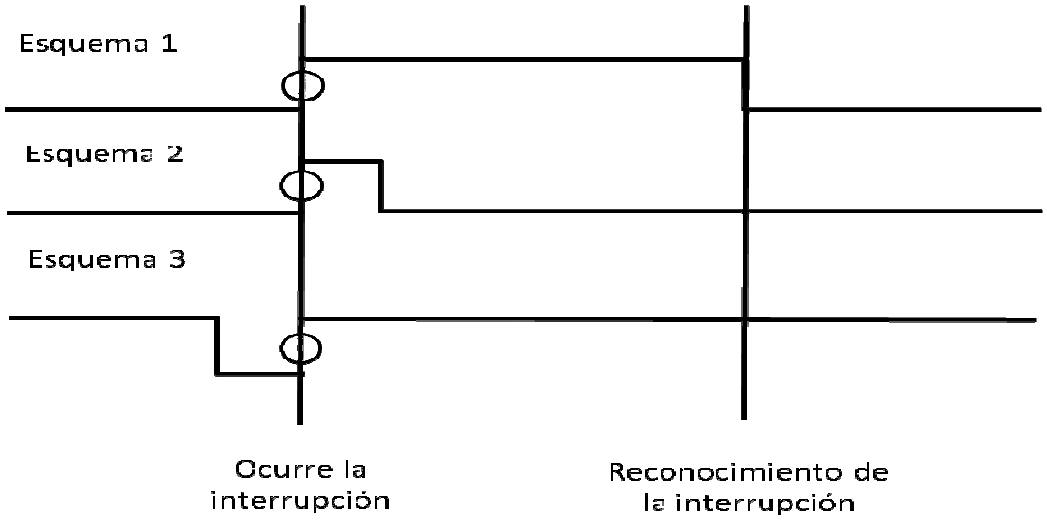


Figura 5.5 Funcionamiento de la detección de interrupciones del INT CONTROLLER.

5.2.3 Fixed Interval Timer (FIT)

El temporizador de intervalo fijo (FIT) es un núcleo clave en la adquisición de datos, ya que es quien marca los intervalos de lectura de los sensores de navegación. Como su nombre lo indica funciona por medio de un temporizador (Figura 5.5)., el cual puede ser configurado para funcionar a intervalos fijos de tiempo de acuerdo a la frecuencia del reloj principal de la plataforma FPGA SPARTAN 3E (50 MHz). Como cualquiera de los otros núcleos correspondientes al bloque AD, este periférico debe ser agregado al proyecto para su configuración y conexiones necesarias. Los principales parámetros del núcleo son el flanco en que se generará luego de cumplida la cuenta máxima y que servirá como señal de interrupción. La señal de reloj y el *reset* del núcleo son opcionales.

Figura 5.6 Diagrama de bloques de generador de interrupciones FIT.

FIT no requiere conectarse a ningún bus de datos ya que interacciona directamente con el microprocesador, para el caso de esta tesis se realizan las conexiones por medio del controlador de interrupciones [47]. También se ha configurado con un *reset* externo, el cual gestiona el microprocesador. El cálculo para determinar el tiempo de la interrupción en ciclos de reloj fue l siguiente:

$F_{tarjeta\ FPGA}$ = 50 MHz

$T_{tarjeta\ FPGA}$ = 20 ns

Si consideramos 1,000,000 de ciclos de reloj (valor estándar en EDK),

1,000,000 x $20x10^{-9}$ = 0.02 s = duración por defecto de FIT

Tomando como base este dato, si requerimos un tiempo de adquisición de 100 ms, tenemos:

20×10^{-3} [s] → 1,000,000 ciclos de reloj

100×10^{-3} [s] → ¿? Ciclos de reloj

$$número\ de\ ciclos\ de\ reloj = \frac{(100x10^{-3})(1000000)}{20x10^{-3}} = 5000000$$

El resultado es 5, 000,000 de ciclos de reloj equivalentes a interrupciones en intervalos de 100 ms. Este intervalo debe ser suficiente para llevar a cabo el proceso completo, desde el envío de las lecturas de datos adquiridos de los sensores de navegación hasta el procesamiento de esos datos y la recepción de comandos de control para los actuadores.

Ahora sólo falta describir el IP que hace posible la comunicación entre la plataforma de desarrollo y el radio módem, para el envío y recepción de datos de forma inalámbrica entre la PC con MATLAB y la plataforma FPGA SPARTAN 3E, se trata del núcleo UART para el manejo de la interfaz serie RS232.

5.3 Bloque de comunicaciones entre la MSA y PC remota

Las comunicaciones son una parte imprescindible para el sistema de simulación satelital para la validación de esquemas de control con base en la técnica HIL, las cuales requieren de los núcleos y hardware adicional para lograr comunicar la MSA con la PC remota. Este bloque representa el vínculo para la transferencia de datos entre los bloques AD y de control de actuadores, sobre la MSA y la PC remota, respectivamente, de esta forma se consigue que el sistema funcione como si estuviese a bordo en su totalidad sobre la MSA. Las comunicaciones comprenden un núcleo UART conectado al microprocesador embebido MicroBlaze y un radio módem (RF) sobre la MSA, mientras que en la PC solamente se requiere de otro radio módem. Además, se deben configurar los puertos e interfaces necesarios, tanto en las plataformas de cómputo, FPGA y PC, como en los radios. Los detalles al respecto de los componentes que se han mencionado se describirán en los apartados siguientes.

5.3.1 Universal Asynchronous Receiver/Transmitter – Data Communication Equipment (UART-DCE)

El controlador Receptor/Transmisor Asincrónico Universal (UART, *Universal Asynchronous Receiver/Transmitter*) es un componente fundamental en los subsistemas de comunicaciones seriales de una computadora, su función es tomar los *bytes* de datos y transmitirlos como bits individuales de forma secuencial, es decir, serialmente (Figura 5.5). A la llegada a su destino, otro bloque UART vuelve a ensamblar los bits en bytes completos. La transmisión serial es utilizada comúnmente con dispositivos módem, comunicaciones entre computadores sin conexiones a una red, terminales y otros dispositivos.

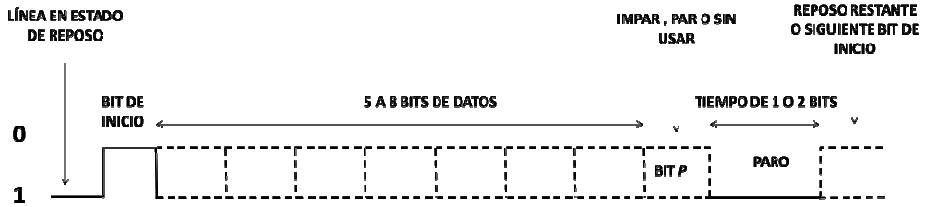


Figura 5.7 Formato de la trama serie RS-232.

El periférico UART es utilizado en el sistema que se describe en esta tesis, para el establecimiento de un vínculo al exterior de la tarjeta de desarrollo FPGA mediante el protocolo de comunicaciones serie RS232, a través de este protocolo de comunicaciones es posible comunicarse con el radio módem a bordo de la MSA e intercambiar datos con la PC remota [48]. Un esquema general de la arquitectura del núcleo UART se muestra en la figura 5.8.

El núcleo UART-DCE funciona con base en el protocolo de comunicaciones RS-232**,** el cual resuelve entre otras cosas las características eléctricas y funcionales de los dispositivos que lo utilizan para la transferencia de datos. En la plataforma de desarrollo FPGA SPARTAN 3E que se utiliza para la implementación del sistema, existe además el hardware basado en un transceptor, lo cual permite el manejo de dos puertos físicos DB9 para la conexión de equipos que cuenten con este protocolo de comunicaciones RS232 denominados DTE y DCE.

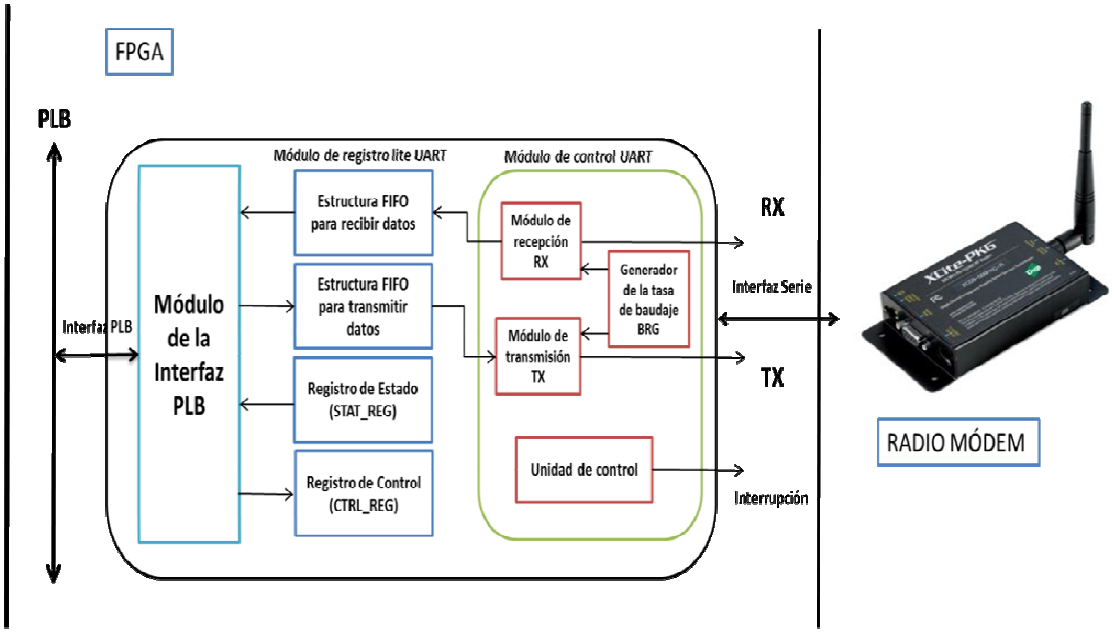


Figura 5.6 Detalle del núcleo IP UART, sus bloques y conexiones más importantes.

El periférico UART-DCE se conecta al bus de periféricos PLB, cuyos eventos de recepción o transmisión de datos permiten también considerarse como una fuente de interrupción para el microprocesador. Sus principales parámetros de configuración son el baudaje, la longitud de la trama enviada, la paridad y la habilitación de esta paridad.

De acuerdo a esos parámetros, para este sistema sólo se han configurado dos de ellos, correspondientes a la velocidad de baudaje, 9600 bps, y la longitud de las tramas, las cuales son de 8 bits.

5.3.2 Descripción del empaquetado de datos en envío a la PC

El bloque AD ha sido descrito en su totalidad en los apartados anteriores de este capítulo con los núcleos que la conforman: I2C, controlador de interrupciones y generador de interrupciones. Adicionalmente se requiere que esos datos adquiridos de los sensores de navegación sean enviados hacia el siguiente segmento del sistema, el cual está montado en una PC remota. Para ello, se requiere de una lógica adicional que opere desde el microprocesador embebido.

Para poder realizar el envío de estos datos, siendo que el problema de comunicaciones a nivel de software se ha resuelto por medio de la adopción del protocolo de comunicaciones serie, se hará por medio de dos radios módem conectados por cables RS-232 en la FPGA y en la PC, respectivamente, para realizar la comunicación de forma inalámbrica entre ellos. Para empaquetar estos datos obtenidos de los sensores y enviarlos a las PC externa, el microprocesador embebido MicroBlaze cuenta con un conjunto de bibliotecas y funciones de entrada y salida estándar para poder enviar o recibir información de otros dispositivos, el programa principal, como se ha dicho, puede contener rutinas programadas en C, C++ o incluso ensamblador, por lo tanto, para realizar el envío de datos desde la plataforma FPGA hacia las PC con MATLAB se ocupará una función ligera nativa del compilador de Xilinx, basado en las bibliotecas estándares de C y la función llamda *printf*. Esta función es *xil_printf*, útil para sistemas embebidos, aunque no tiene soporte para números flotantes ni tampoco números de 64 bits [49].

Para los propósitos de esta tesis, es una función de mucha utilidad para la transferencia de los datos obtenidos por los sensores de navegación hacia la PC remota. El prototipo de la función está basado en la siguiente macro:

void xil_printf (const *char ctrl1, ...)

La función se encuentra dentro del archivo de cabecera *stdio.h* y a su vez ésta se incluye en el fichero de la lista estándar de bibliotecas de Microblaze. Teniendo en cuenta estos detalles, los datos de los sensores de navegación, es decir, el tamaño de las tramas se basa en el número de bytes, los cuales son 10 (9 son correspondientes a los sensores de navegación, x, y, z, por cada sensor y un byte adicional para la estadística de errores de comunicación, manejado directamente en la lógica del microprocesador).

En el siguiente apartado se continúa abordando la parte de recepción de datos en la plataforma de desarrollo FPGA SPARTAN 3E, la cual incluye el diseño e integración del núcleo personalizado PWM para el control de actuadores.

5.4 Control de actuadores

El control de actuadores es la parte encargada de recibir los comandos mediante el núcleo UART, descrito anteriormente, e interpretar los comandos de control, es decir, el ciclo de trabajo y sentido de giro de las ruedas inerciales impulsadas por motores; esto se logra mediante las instrucciones en el programa principal del microprocesador MicroBlaze, donde se indica que se recibirán bytes de datos, uno para el ciclo de trabajo y un bit para el sentido de giro, y después serán escritos en los registros de datos asociados al núcleo de Modulación de Ancho de Pulso o PWM, el cual fue diseñado, desarrollado e implementado para integrarlo en la arquitectura embebida en la plataforma SPARTAN 3E. La función principal del núcleo PWM es modular el ciclo de trabajo de una señal periódica para controlar la velocidad de los motores DC de los actuadores y la dirección en que giran. Los comandos que requiere el núcleo PWM son generados por los algoritmos de procesamiento que se ejecutan en la PC remota con MATLAB y enviados hacia la plataforma FPGA a través de los radios módem. Por otra parte, los actuadores como motores de corriente directa hacen que las ruedas inerciales logren el movimiento para de maniobras de control sobre del sistema de simulación. Estos motores tienen características eléctricas de las que depende su desempeño y velocidad de respuesta, de lo que se hizo un pequeño experimento para determinarlas y los resultados son presentados más adelante; aunque, una característica fundamental es el par mecánico que desarrollan los motores debido a la corriente inducida, el cual se calcula en el algoritmo de CONTROL multiplicando el torque obtenido por la ley de control y una constante obtenida de las especificaciones eléctricas del motor, obteniendo los dos principales parámetros del núcleo PWM, ciclo de trabajo y sentido de giro.

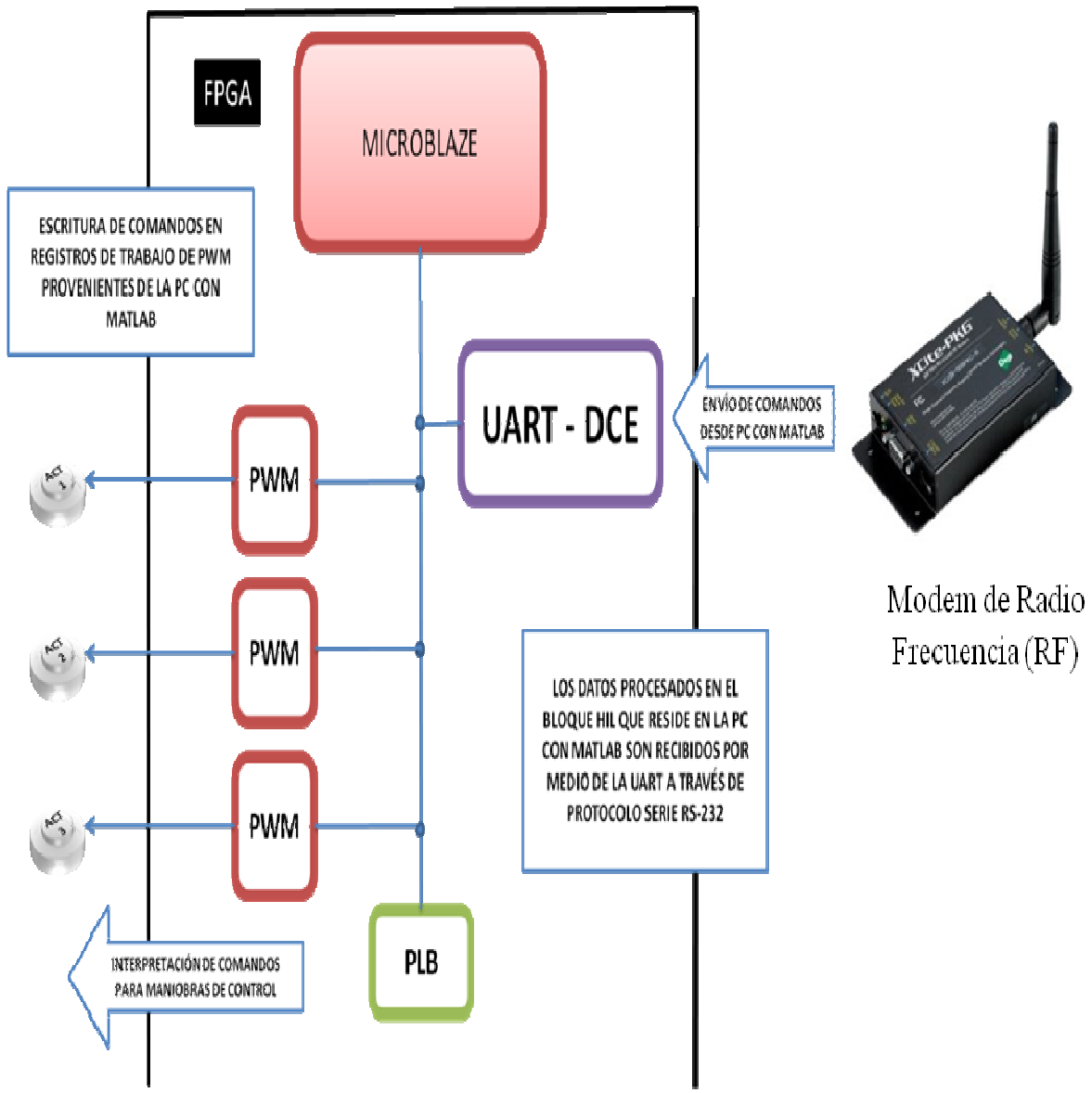


Figura 5.9 Diagrama de bloques de la composición del subsistema control de actuadores.

En seguida se explicará el diseño del núcleo PWM, el cual es un núcleo que ha sido desarrollado, personalizado y agregado a la arquitectura del sistema de simulación para la validación de esquemas de control, y se plantea la parte de diseño y la realización del núcleo, enfatizando cómo se llevo a cabo su construcción, pruebas e integración con el sistema, y finalmente se dará un resumen breve sobre el esquema completo implementado en la plataforma SPARTAN 3E.

5.4.1 Diseño del núcleo PWM

El diseño del núcleo PWM tiene el objetivo de crear el módulo de control de velocidad de los motores de cada una de las ruedas inerciales. La técnica de Modulación por Ancho de Pulso (PWM) tradicionalmente es usada para controlar la velocidad de un motor mediante la modulación de un par inducido. No obstante, en este trabajo de tesis, además de controlar la velocidad de los motores se debe controlar la asignación en el sentido de giro de dichos motores. Para ello se diseñó un núcleo que considerara estas características en su funcionamiento. En la Figura 5.10 se expone la arquitectura del núcleo PWM en la que se muestran las entradas y salidas del mismo, que además, realiza las tareas siguientes: calcular el ciclo de trabajo, tiene un contador periodo, implementa un comparador, hace un cambio de nivel de la modulación y cambio de dirección; funciones que fueron programadas en lenguaje VHDL en un archivo que fue sintetizado en la herramienta Project Navigator, y probado mediante un banco de pruebas con un simulador *waveform*.

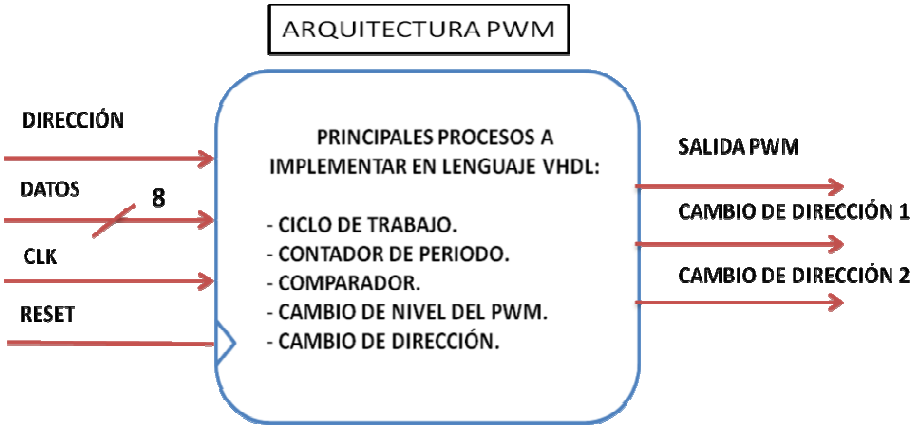


Figura 5.10 Diseño de la entidad PWM y sus principales componentes a desarrollar.

Durante su fase de diseño e integración, se crearon bancos de pruebas para validar el funcionamiento del núcleo PWM, de tal forma que se pudiera simular las señales de entrada para el núcleo, y las salidas correspondientes se visualizaron con ayuda de un osciloscopio. Como se ha mostrado en la Figura 5.10, el núcleo incluye la implementación de diversos procesos que deben realizarse como parte del comportamiento de la arquitectura creada. Las entradas son: Los datos de 8 bits o un byte, corresponden al ciclo de trabajo a modular; el sentido de giro o dirección sólo es de 1 bit; la señal de reloj, la cual es la señal generada por la plataforma de desarrollo FPGA, y una señal de *reset* de 1 bit la cual podría habilitarse o no, según se requiera.

Cabe señalar que para obtener la versión final de este núcleo fue necesario evaluarlo por etapas; mientras que se logró implementar primeramente el esquema para obtener la modulación de la señal y evaluarla mediante un banco de pruebas, posteriormente se agregó la funcionalidad para el cambio de sentido de giro de los motores. Finalmente el proyecto se sintetizó y se dejó listo para su instanciación dentro del sistema embebido en la plataforma de desarrollo SPARTAN 3E, donde se incluye tres núcleos iguales, uno para cada uno de los actuadores. Esta integración al sistema se explica a continuación.

5.4.2 Integración del núcleo PWM al sistema embebido

Para que el núcleo PWM desarrollado pueda integrarse al microprocesador MicroBlaze, es necesario realizar tareas y configuraciones para su conexión e integración con el sistema a bordo de la MSA. Como se ha visto a lo largo del capítulo el microprocesador MicroBlaze es el sistema base y es quien gestiona los núcleos periféricos personalizados e IP, y no sólo, sino que gestiona también las conexiones y realiza muchas tareas de forma transparente. Pero las que son de interés en este trabajo, son las programadas en código C, donde se indica qué se debe hacer con los datos recibidos, y en el caso del control de actuadores, se deben manipular para escribirlos en los registros del núcleo PWM.

El procedimiento para agregar un núcleo personalizado es diferente al de agregar un núcleo IP, pues una vez que el programa desarrollado en lenguaje VHDL ha sido sintetizado y verificado en su funcionamiento, la siguiente etapa es utilizar la aplicación XPS Platform Studio para agregar el núcleo PWM al proyecto en el FPGA. En esta etapa deben considerarse elementos necesarios para la integración:

- El código fuente VHDL sintetizado del núcleo PWM desarrollado en Project Navigator.
- La creación de la arquitectura base para importar un núcleo al proyecto.
- La configuración y conexiones necesarias para su interacción con el microprocesador embebido.

Las conexiones o mapeo se refieren a un conjunto de instrucciones dentro del archivo de lógica de usuario creado en la importación del núcleo PWM, son señales que permiten la interacción y compartición de la información entre el núcleo personalizado y el microprocesador embebido a través del bus de datos PLB, como se muestra en la Figura 5.11.

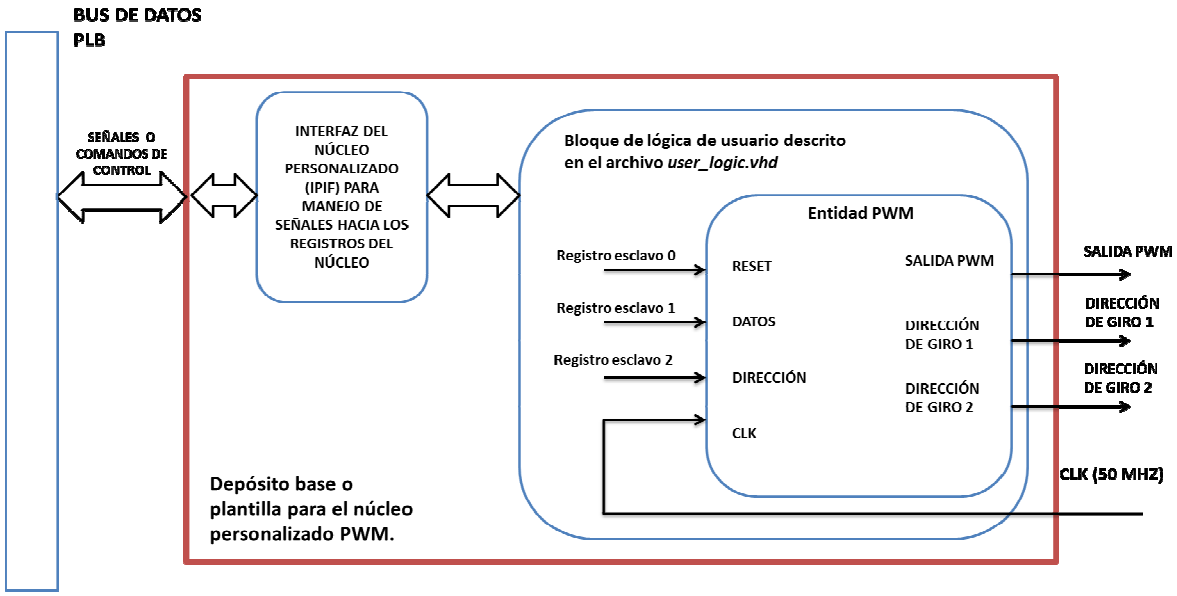


Figura 5.11 Arquitectura del núcleo personalizado PWM integrado en la PFGA.

Una vez agregado, el núcleo aparecerá dentro del catálogo de núcleos y configurarse sus salidas hacia algún puerto. El PWM ha sido entonces configurado exitosamente y sus salidas están conectadas a las terminales de entrada/salida de la tarjeta SPARTAN 3E, a las que se conectan las líneas de los motores; la asignación de terminales se realiza en un archivo denominado *restricciones de usuario* (UCF) para vincular las salidas de los núcleos con los puertos físicos en la plataforma FPGA SPARTAN 3E, este archivo puede consultarse directamente en la interfaz de la aplicación XPS. Realizado esto, ha quedado integrado el núcleo al subsistema a bordo de la MSA. Lo siguiente, a partir de todo lo antes expuesto en este capítulo, es la generación del flujo de bits que será descargado en la plataforma FPGA. Por ello a continuación se expone una explicación general de las partes que componen el subsistema.

5.5 Integración del subsistema a bordo de la MSA

De acuerdo al análisis que se hizo de las diferentes alternativas para implementar el sistema de simulación satelital para la validación de esquemas de control, se eligió trabajar bajo el esquema de la técnica HIL, con la que se puede acelerar la implementación, validación de algoritmos y la obtención de resultados.

Por ello, el sistema estará funcionando en dos bloques diferentes pero comportándose como si fuera uno solo, a lo que se denomina simulación en tiempo real, independientemente de los retrasos comunes en los diferentes dispositivos electrónicos.

En este capítulo se expone la descripción de los módulos que irán embebidos en la plataforma de desarrollo SPARTAN 3E a bordo de la MSA, estos componentes son: la adquisición de datos de los sensores de navegación, las comunicaciones sobre la plataforma FPGA para envío y recepción de datos con la PC remota y el control de los actuadores (Figura 5.12).

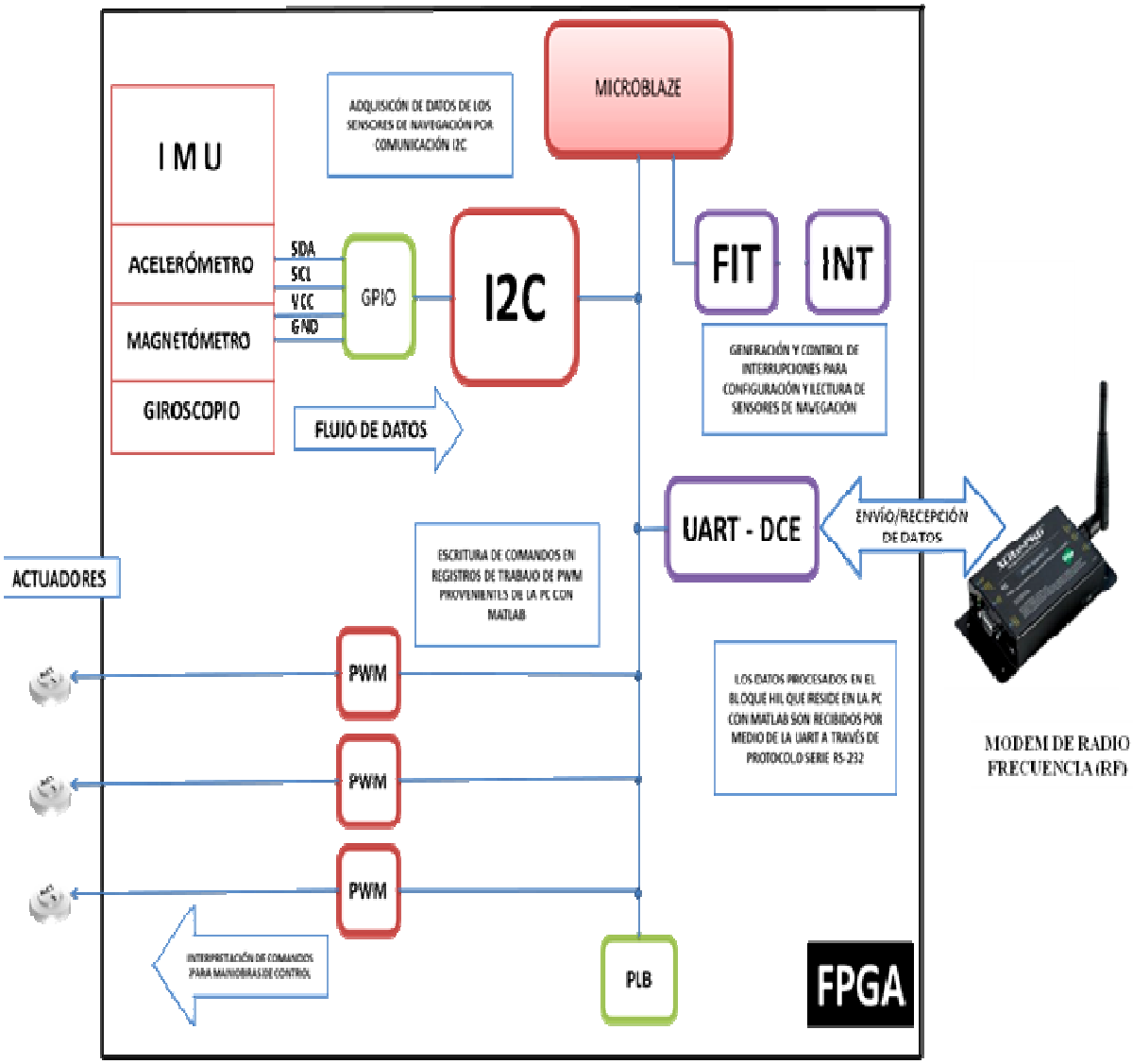


Figura 5.12 Subsistema integrado a bordo de la MSA.

La experiencia adquirida para reducir y solucionar los problemas que se presentaron durante el proceso de desarrollo e implementación de este primer bloque permitirán describir mejores recomendaciones con el fin de que dichos problemas no se presenten o se reduzcan en gran parte en el desarrollo continuo de este sistema de simulación, y que es otra aportación en del presente trabajo. en resumen, con base en la forma de atacar el problema, es necesario realizar un análisis respecto a la optimización de los recursos de hardware y software con los que cuenta en la plataforma de desarrollo FPGA, organizando la carga de tareas y procesos desde el planteamiento inicial del esquema de desarrollo para un mejor aprovechamiento de los recursos, y considerar el uso de componentes que estén listos para ser agregados y que sean requeridos para resolver tareas de procesamiento o interfaz externa o interna con diversos dispositivos según lo demanden las aplicaciones, o la creación de nuevos elementos para implementar nuevas funcionalidades que requieran resolver necesidades específicas para el sistema, como el caso de los núcleos PWM; todo, sin perder de vista los objetivos que se persiguen para lograr el funcionamiento adecuado de todo el conjunto.

El siguiente capítulo abordará la implementación de los algoritmos de coprocesamiento de algoritmos, los cuales son TRIAD, EKF y CONTROL, y que se ejecutan en una PC externa con MATLAB, y de lo que se dará detalles en el siguiente capítulo.

Capítulo 6

Bloque de coprocesamiento en la PC utilizando MATLAB

Si bien es cierto que el uso de una plataforma con recursos mínimos, cuya estructura se basa en el uso de un microcontrolador (*soft* o *hard*) para la implementación de procesos complejos ya sean de control o de cualquier otra aplicación en ingeniería, representa una gran oportunidad para el desarrollo de sistemas portátiles y a la medida de las necesidades de la aplicación, lo cual, en muchas ocasiones implica una serie de retos que no son salvables fácilmente o bien, que en su defecto, implica la inversión de una gran cantidad de tiempo para el desarrollo de ellos. Por ese motivo, es importante el uso de técnicas alternativas que faciliten la rápida implementación de esquemas y algoritmos que validen preliminarmente el desempeño del sistema y faciliten su posterior implementación en una plataforma o dispositivo definitivo. Algunas de estas técnicas, como la que se describirá en este capítulo, denominada *hardware in the loop (*HIL*),* tiene el objetivo principal de acelerar la obtención de resultados experimentales, permitiendo además evaluar algunos de los componentes físicos del sistema. El uso de esquemas de coprocesamiento externo, generalmente ubicados en computadoras personales, permite la optimización de los algoritmos y su preparación para una transferencia rápida relativamente a alguna otra plataforma de procesamiento. En este capítulo se abordará la descripción del software de procesamiento que reside en una PC remota dentro del esquema de validación HIL para el sistema de control de orientación propuesto.

6.1 Introducción

El sistema de simulación satelital para la validación de esquemas de control consta de dos bloques, uno de ellos ha sido implementado mediante MATLAB con el objetivo de que parte del procesamiento de datos se realice fuera de la MSA, de tal forma que el sistema funcione como si estuviese integrado totalmente a bordo de la MSA. Inicialmente, se pretendía que este bloque de coprocesamiento fuese implementado sobre el entorno gráfico SIMULINK, no obstante, durante la etapa de pruebas de comunicación de datos entre las plataformas FPGA y PC/SIMULINK, una vez configurados los bloques predefinidos con los parámetros que se requerían, hubo diversos problemas como la pérdida de información, interpretación incorrecta del formato de los datos, lentitud y fallas en las comunicaciones inalámbricas, y se detectó en los visores de datos (bloques SCOPE) que mostraban datos que no se esperaban, ya sea en el formato de envío o magnitudes incoherentes con lo esperado. Esto provocaba la pérdida de datos, mal funcionamiento e interrupción de la ejecución del programa. Por ello, se decidió implementar este bloque del sistema en un proyecto de *scripts* (programas con instrucciones secuenciales) de MATLAB, el cual está integrado por un conjunto de archivos tipo **.m** que contienen funciones cuya estructura ofrece mayor flexibilidad y manipulación de parámetros de las interfaces de una PC, así como control total sobre el código para realizar los cálculos, pasar parámetros, formatear datos y definir funciones a la medida, lo cual en entornos gráficos como SIMULINK o LABVIEW no podría realizarse con tanta facilidad.

Entre las prioridades del desarrollo se encontraban el manejo y configuración del puerto serie de comunicaciones, así como optimizar los recursos de procesamiento para los algoritmos. El *script* o programa principal que se desarrolló contiene las llamadas a las funciones de los algoritmos, así como la configuración y gestión de las comunicaciones: adquisición de datos, recepción de datos de los sensores de navegación y envío de comandos de control.

Cabe señalar que dos de los algoritmos utilizados e implementados en esta tesis, fueron desarrollados previamente en un trabajo de maestría [referencia córdova], los cuales son EKF y TRIAD, por ello se comenzará describiendo las generalidades de su funcionamiento, detallando solamente los aspectos de implementación más importantes en MATLAB o las modificaciones que se realizaron para su ajuste con el sistema de simulación satelital y sus características. En virtud de ello, se comenzará con el algoritmo encargado de determinar la orientación del MSA, tomando como parámetro las mediciones de los sensores de navegación, tales como magnetómetro y acelerómetro para lograrlo.

La estructura o diagrama de bloques de cómo quedo distribuido este bloque implementado en MATLAB ejecutándose sobre una PC con Windows XP (no obstante su uso se extrapola a Windows 7 o posterior, que cuente con la versión 10 u 11 de MATLAB) se muestra en la Figura 6.1.

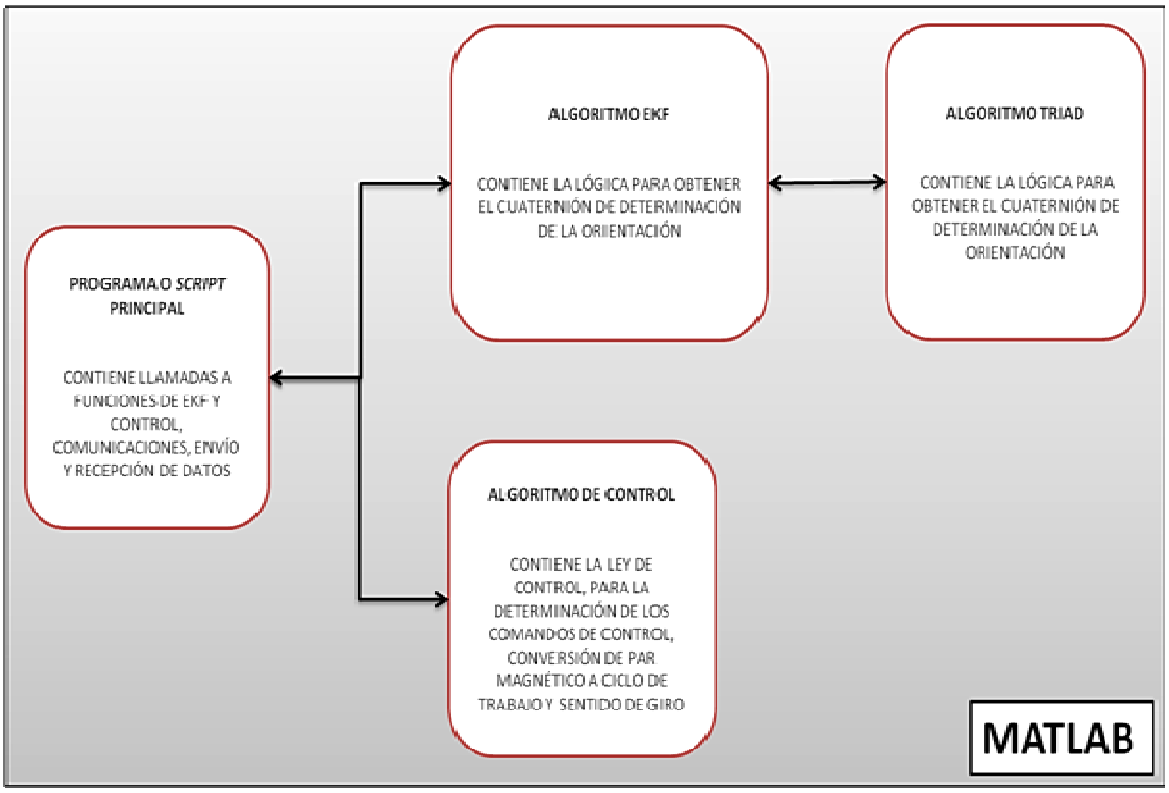


Figura 6.1 Bloques que incluye el subsistema de coprocesamiento en MATLAB.

6.2 Descripción del algoritmo TRIAD

El sistema desarrollado en este trabajo de tesis presenta dos partes fundamentales para lograr su objetivo y desarrollo, las cuales son la determinación de la orientación y el control de apuntamiento de la MSA. En este apartado se abordarán los aspectos del funcionamiento de algoritmo TRIAD (*Three-Axis Attitude Determination*), el cual es utilizado para determinar la orientación de la MSA en tres ejes. A continuación se hará una breve descripción del algoritmo y cómo funciona, cuáles son las operaciones que lleva a cabo, sus parámetros de entrada, así como las salidas que se obtiene de él.

El algoritmo TRIAD es un método sencillo para determinar para la orientación de un sistema, en comparación con otros algoritmos como los llamados recursivos, v.gr. el método de Gauss-Newton. Su principal ventaja es que requiere de un menor número de mediciones (en este caso de los sensores de navegación) para su operación y contempla un menor número de recursos de procesamiento para su ejecución, lo cual resulta un factor a considerar en la implementación. El algoritmo consiste en encontrar una matriz de transformación, la cual relaciona dos sistemas coordenados usando dos vectores de medición (provistos por dos sensores de medición inerciales, magnetómetro y acelerómetro) y dos vectores de referencia, expresados en el sistema coordenado en el que se efectuó la medición y en el de referencia respectivamente [50]. Estas tríadas de vectores expresadas como $\boldsymbol{r_1}$,$\boldsymbol{r_2}$,$\boldsymbol{r_3}$ y $\boldsymbol{s_1}$,$\boldsymbol{s_2}$,$\boldsymbol{s_3}$ corresponden a las mediciones de los sensores de navegación y a los vectores de los marcos de referencia, respectivamente, para obtener la matriz de rotación $\boldsymbol{A_{rot}}$ de acuerdo con las siguientes operaciones:

$$\boldsymbol{r_1} = \frac{\boldsymbol{uM}}{||\boldsymbol{uM}||} \qquad \textbf{(6.1)}$$

$$\boldsymbol{r_2} = \frac{\boldsymbol{r_1} \times \boldsymbol{vM}}{||\boldsymbol{r_1} \times \boldsymbol{vM}||} \qquad \textbf{(6.2)}$$

$$\boldsymbol{r_3} = \boldsymbol{r_1} \times \boldsymbol{r_2} \qquad \textbf{(6.3)}$$

$$\boldsymbol{s_1} = \frac{\boldsymbol{uR}}{||\boldsymbol{uR}||} \qquad \textbf{(6.4)}$$

$$\boldsymbol{S_2} = \frac{\boldsymbol{S_1} \times \boldsymbol{vR}}{||\boldsymbol{S_1} \times \boldsymbol{vR}||} \qquad \textbf{(6.5)}$$

$$\boldsymbol{s_3} = \boldsymbol{s_1} \times \boldsymbol{s_2} \qquad \textbf{(6.6)}$$

$$\boldsymbol{A_{rot}} = \boldsymbol{r_1} * \boldsymbol{s_1}^{\boldsymbol{T}} + \boldsymbol{r_2} * \boldsymbol{s_2}^{\boldsymbol{T}} + \boldsymbol{r_3} * \boldsymbol{s_3}^{\boldsymbol{T}} \qquad \textbf{(6.7)}$$

Una vez calculada la matriz de rotación $\boldsymbol{A_{rot}}$, se debe someter a un procedimiento adicional para expresarla como un cuaternión de orientación. Entonces, los resultados de $\boldsymbol{A_{rot}}$ se transforman a una representación en parámetros simétricos de Euler, donde se obtiene el cuaternión, siendo los primeros tres elementos los que definen la parte vectorial, correspondientes a los ejes **X**, **Y**, **Z** y el cuarto elemento corresponde a la parte escalar. Las relaciones para la obtención de los elementos a partir de los resultados de a $\boldsymbol{A_{rot}}$ se muestran en las ecuaciones **(6.8)**, **(6.9)** y **(6.10)**.

$$y_1 = e_1 sin\,(\varphi/2) \qquad (6.8)$$

$$y_2 = e_2 sin\,(\varphi/2) \qquad (6.9)$$

$$y_3 = e_3 sin\,(\varphi/2) \qquad (6.10)$$

$$y_4 = cos\,(\varphi/2) \qquad (6.11)$$

Tal que e_i corresponde al eje de Euler, y φ es el ángulo principal de rotación. Estas cuatro componentes forman el cuaternión de orientación$\boldsymbol{y}$, el cual será obtenido con base en las operaciones descritas e implementadas en MATLAB, quien ofrece soporte para realizar operaciones matriciales, permitiendo reducir los tiempos de desarrollo y acelerar la obtención de resultados.

6.3 Implementación del TRIAD

La implementación del algoritmo TRIAD en MATLAB se realizó considerando su estructura como una función secuencial invocada desde otra función, la del algoritmo EKF (descrita más adelante), la cual llamará a la función TRIAD y le pasará los parámetros de entrada que son las componentes **X**, **Y**, **Z** de cada uno de los sensores de navegación acelerómetro y magnetómetro, y tendrá como salida un vector de cuatro elementos, denominado cuaternión de orientación. El procedimiento mostrado en el apartado anterior puede implementarse casi directamente en el lenguaje de programación de MATLAB, lo que muestra el potencial que tiene esta aplicación en cuanto a la biblioteca de funciones listas para utilizarse que incluye una gran cantidad de documentación tanto en el propio software como en foros de desarrolladores en internet, la cual permite acelerar el aprendizaje para utilizar esas funciones.

Las operaciones consideradas en el algoritmo se basan en vectores y sus componentes, y también están asociadas con matrices, y siendo que el código en MATLAB es un lenguaje basado en matrices, como se ha dicho, las expresiones y cálculos son descritos en una forma muy aproximada a la representación matemática expuesta. La estructura funcional y el flujo de datos del algoritmo implementado en MATLAB se describen en el diagrama de la Figura 6.2:

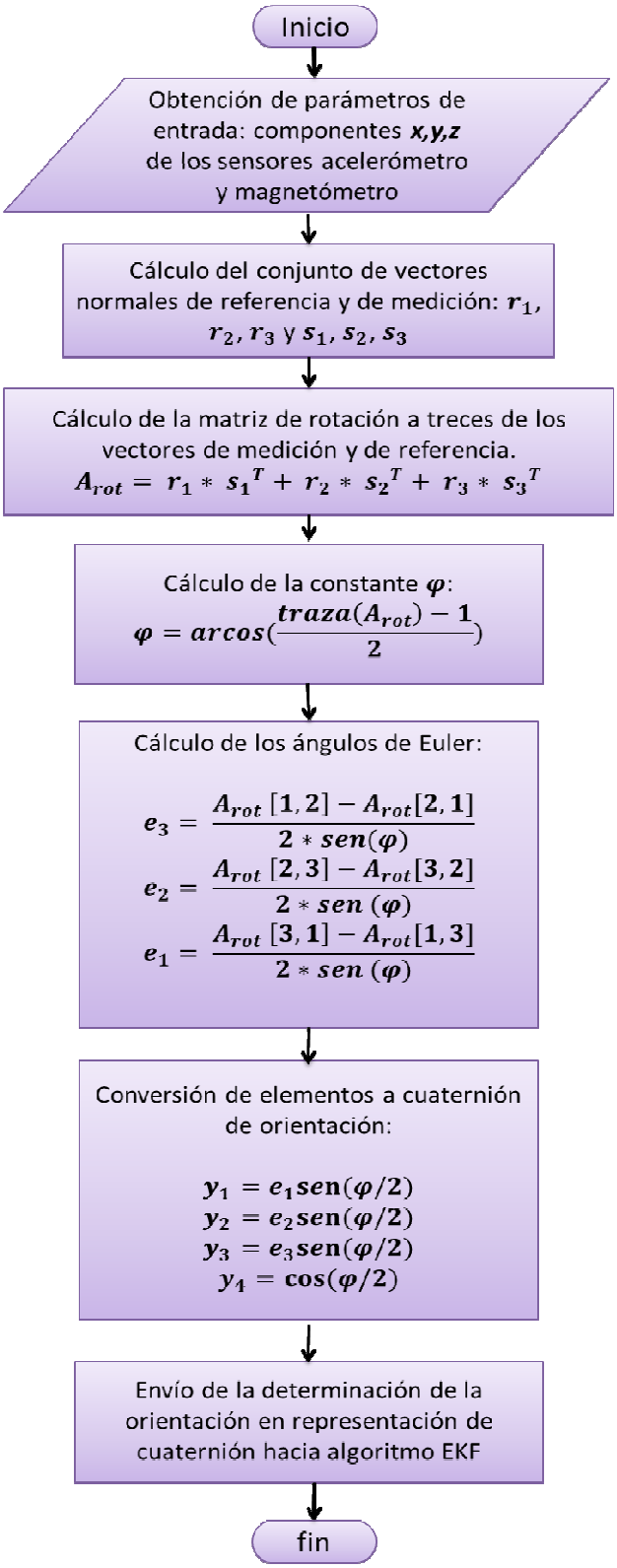


Figura 6.2 Bloques del algoritmo TRIAD que incluye el subsistema de coprocesamiento en MATLAB.

Debido a que las mediciones de los sensores de navegación utilizados para el cálculo de la determinación de la orientación son afectadas por el ruido producido por la acción de los motores eléctricos (actuadores) de la MSA y otros dispositivos a bordo de ésta, se ha aprovechado las características del algoritmo EKF como un filtro para mitigar este problema. A continuación se explicará más sobre el algoritmo EKF y su fin en este trabajo de tesis.

6.4 Descripción del algoritmo EKF

El filtro de Kalman en su forma general tiene una variante conocida como *Extended Filter Kalman* (EKF) discreto, es uno de los algoritmos que forma parte del bloque de coprocesamiento del esquema de trabajo HIL descrito en esta tesis, implementado y ejecutado en MATLAB sobre una PC externa. Se trata de un algoritmo estimador recursivo que fue desarrollado por Rudolph Emil Kalman a finales de la década de 50's, con la finalidad de filtrar y predecir sistemas lineales. Estos algoritmos actualmente se utilizan por su conveniencia en cuanto que no requieren almacenamiento de datos pasados y permiten el procesamiento en tiempo real de nuevas observaciones entrantes.

El filtro de Kalman es un conjunto de instrucciones que aborda el problema general de la estimación, es una herramienta utilísima que soporta estados presente, pasado y futuro de un sistema perturbado por ruido estocástico [51].

La implementación de este algoritmo tuvo como base el desarrollo realizado en [52], cuyos resultados exitosos por medio de simulaciones numéricas y utilizando modelos de realidad virtual permitieron con gran nivel de confianza y con algunos ajustes de por medio tomarlo como base para este trabajo de tesis. Los procesos que componen el conjunto de ecuaciones del Filtro de Kalman Extendido Discreto se describen en seguida.

De forma general y matemática, se considera un sistema no lineal, de la forma [53]:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{t}) + \boldsymbol{\omega}(\boldsymbol{t}) \qquad \textbf{(6.12)}$$

$$\boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{t}) + \boldsymbol{v}(\boldsymbol{t}) \qquad \textbf{(6.13)}$$

Donde:

$\boldsymbol{v(t)}$ se asume que es una función de ruido blanco de las mediciones.

$\boldsymbol{\omega(t)}$es una función de ruido blanco del proceso del sistema.

$\boldsymbol{x}$ es el vector que contiene las variables del sistema de espacio de estados.

$\boldsymbol{u}$ es el vector de comandos de control.

$\boldsymbol{y}$ es el proceso de medición.

$\boldsymbol{f(x,u,t)}$ y $\boldsymbol{h(x,t)}$ son funciones no lineales.

En el Filtro de Kalman Extendido Continuo el proceso de estimación está dado por:

$$\dot{\hat{\boldsymbol{x}}} = \boldsymbol{F}(\hat{\boldsymbol{x}}, \boldsymbol{u}, \boldsymbol{t}) + \boldsymbol{K}\boldsymbol{v} \qquad \textbf{(6.14)}$$

Donde $\boldsymbol{v}$ es el proceso de innovación definido por:

$$\boldsymbol{v} = \boldsymbol{y} - \boldsymbol{H}(\hat{\boldsymbol{x}}) \qquad \textbf{(6.15)}$$

Y $\boldsymbol{K}$es la ganancia de Kalman dada por:

$$\boldsymbol{K} = \boldsymbol{P}\boldsymbol{H}^{\boldsymbol{T}}\boldsymbol{R}^{-\boldsymbol{1}} \qquad \textbf{(6.16)}$$

Siendo $\boldsymbol{R}$ la matriz de covarianza de las mediciones, **Q** es la matriz de covarianzas del proceso,$\boldsymbol{P}$ la ecuación de error de covarianza:

$$\dot{\boldsymbol{P}} = \boldsymbol{F}\boldsymbol{P} + \boldsymbol{P}\boldsymbol{F}^{\boldsymbol{T}} - \boldsymbol{P}\boldsymbol{H}^{\boldsymbol{T}}\boldsymbol{R}^{-\boldsymbol{1}}\boldsymbol{H}\boldsymbol{P} + \boldsymbol{Q} \qquad \textbf{(6.17)}$$

Para la primera iteración, se debe de establecer una condición inicial para el vector de estados $\hat{\boldsymbol{x}} = \hat{\boldsymbol{x}}_{\boldsymbol{0}}$ y su matriz de error de covarianza $\boldsymbol{P} = \boldsymbol{P}_{\boldsymbol{0}}$.

Finalmente, $\boldsymbol{F}(\hat{\boldsymbol{x}}, \boldsymbol{u}, \boldsymbol{t})$ y $\boldsymbol{H}(\hat{\boldsymbol{x}}, \boldsymbol{t})$ vienen del proceso de *linealizar* a lo largo de las trayectorias de las variables estimadas $\hat{\boldsymbol{x}}$:

$$\boldsymbol{F} = \frac{\partial \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{t})}{\partial \boldsymbol{x}}\Big|_{\boldsymbol{x}=\hat{\boldsymbol{x}}} \qquad \boldsymbol{H} = \frac{\partial \boldsymbol{h}(\boldsymbol{x},\boldsymbol{t})}{\partial \boldsymbol{x}}\Big|_{\boldsymbol{x}=\hat{\boldsymbol{x}}} \qquad \textbf{(6.18)}$$

En [54] se puede encontrar una descripción matemática más detallada y completa de este procedimiento. En la Figura 6.3 se muestra el diagrama con los procesos que componen el EKF discreto y el flujo de los datos.

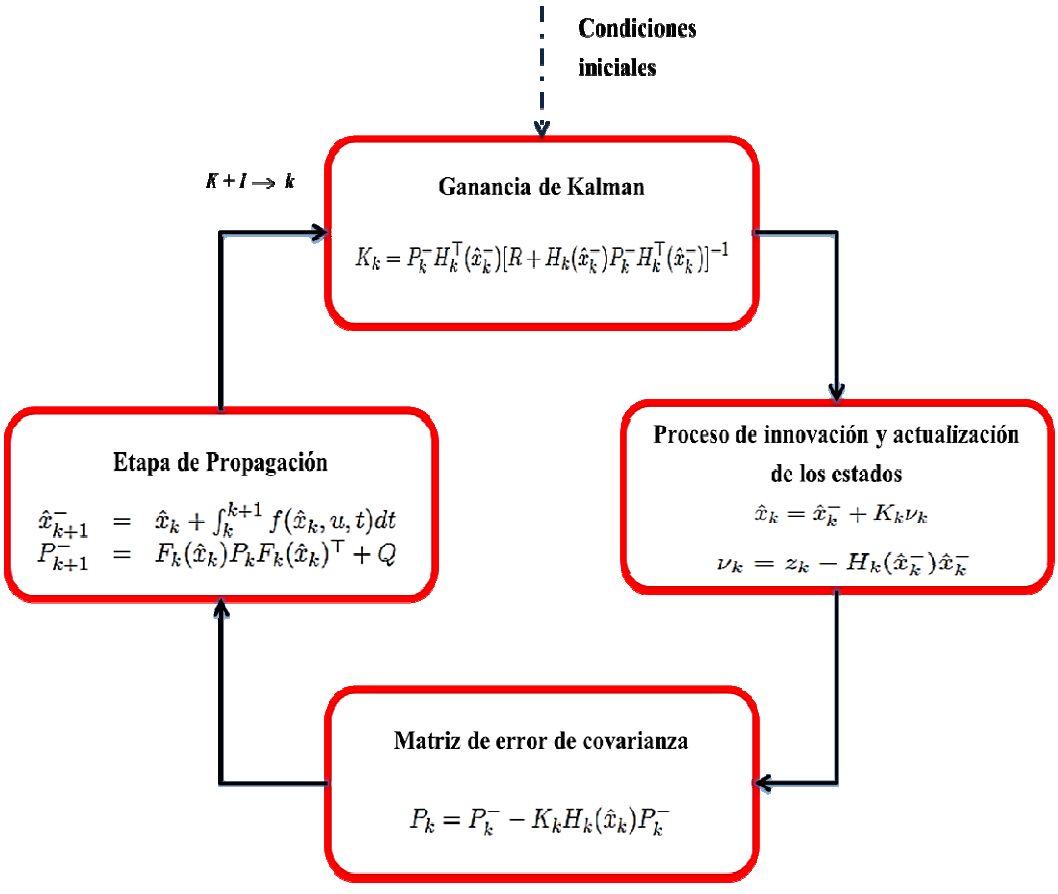


Figura 6.3 Diagrama de procesos del algoritmo EKF discreto.

La implementación de EKF se expone en la siguiente sección, donde se explican aquellas modificaciones y ajustes que se llevaron a cabo para la integración con el sistema de determinación de la orientación y control.

6.5 Implementación del algoritmo EKF

Con base en los requerimientos y características del sistema que se ha integrado en este trabajo de tesis, el algoritmo EKF ha sido modificado de tal forma que pueda ajustarse a las condiciones, componentes y características de la plataforma MSA. El algoritmo está comprendido por un conjunto de ecuaciones complejas, las cuales tienen asociadas un gran número de operaciones aritméticas y cálculos de funciones trigonométricas, así como el manejo de elementos matriciales que son parte de la lógica del funcionamiento, por ello, el objetivo de la implementación en MATLAB es que todo el subsistema de procesamiento y control quede listo y optimizado para ser integrado en una etapa de desarrollo posterior a una plataforma FPGA.

Cabe señalar anticipadamente, que esta transferencia entre el sistema desarrollado en MATLAB y en el FPGA, con llevará una fase previa de análisis de codiseño, en la cual se tendrá que analizar la conveniencia de desarrollar algunas partes del algoritmo en hardware, como elementos de lenguaje HDL o bien, como parte del firmware del procesador embebido que se utilizaría en el sistema en el FPGA.

El funcionamiento de cada uno de los procesos que lleva a cabo el algoritmo EKF será descrito en seguida, los cuáles guardan una relación intrínseca pues no son independientes, si no que al ser iterativo, su naturaleza es secuencial y dependiente de los resultados obtenidos en cada proceso.

Finalmente en la Figura 6.3, se muestra cómo quedó comprendido en bloques descritos en instrucciones de MATLAB.

Inicializar variables: En este proceso se establece un estado inicial del sistema, por ello se pasan como parámetros variables, y posteriormente deberán ser estimadas y corregidas por la realimentación en los resultados de los parámetros durante la ejecución del programa, incluyendo el cuaternión de orientación y la velocidad angular. En este proceso viene implícito el establecimiento de constantes, como el caso de covarianzas de error, auxiliares en la corrección de la desviación de error implícita en el giróscopo.

Proceso de innovación y actualización de los estados: Este proceso parte del concepto de la estimación de valores, donde no hay algo nuevo dentro del proceso, sino que se hace referencia a los estados obtenidos anteriormente. Se incluye la determinación de la orientación por TRIAD, y con base en ello se complementa la parte de estimación y corrección de los datos.

Matriz de error: Este proceso considera las matrices de error expresadas en términos de las covarianzas de error correspondientes a los procesos de estimación y la medición, es decir, que incluye valores estimados realimentados en las iteraciones y los obtenidos con base en las mediciones de los sensores de navegación.

Etapa de propagación: Al ser descrito el sistema como un modelo no lineal, este proceso realiza una *linealización* del dicho modelo para ajustarlo a las características del sistema, es decir, se utiliza un método para realizar correcciones y reducción de errores en el modelo propuesto, y se aplica sobre la matriz de covarianza de error así como en las variables obtenidas del proceso de innovación, donde finalmente se obtiene el cuaternión estimado y la velocidad angular estimada del sistema.

Ganancia de Kalman: Es una matriz que establece la magnitud del error entre la estimación calculada y la medida de las matrices de error de covarianza, principalmente reduce la covarianza de error medida o a posteriori.

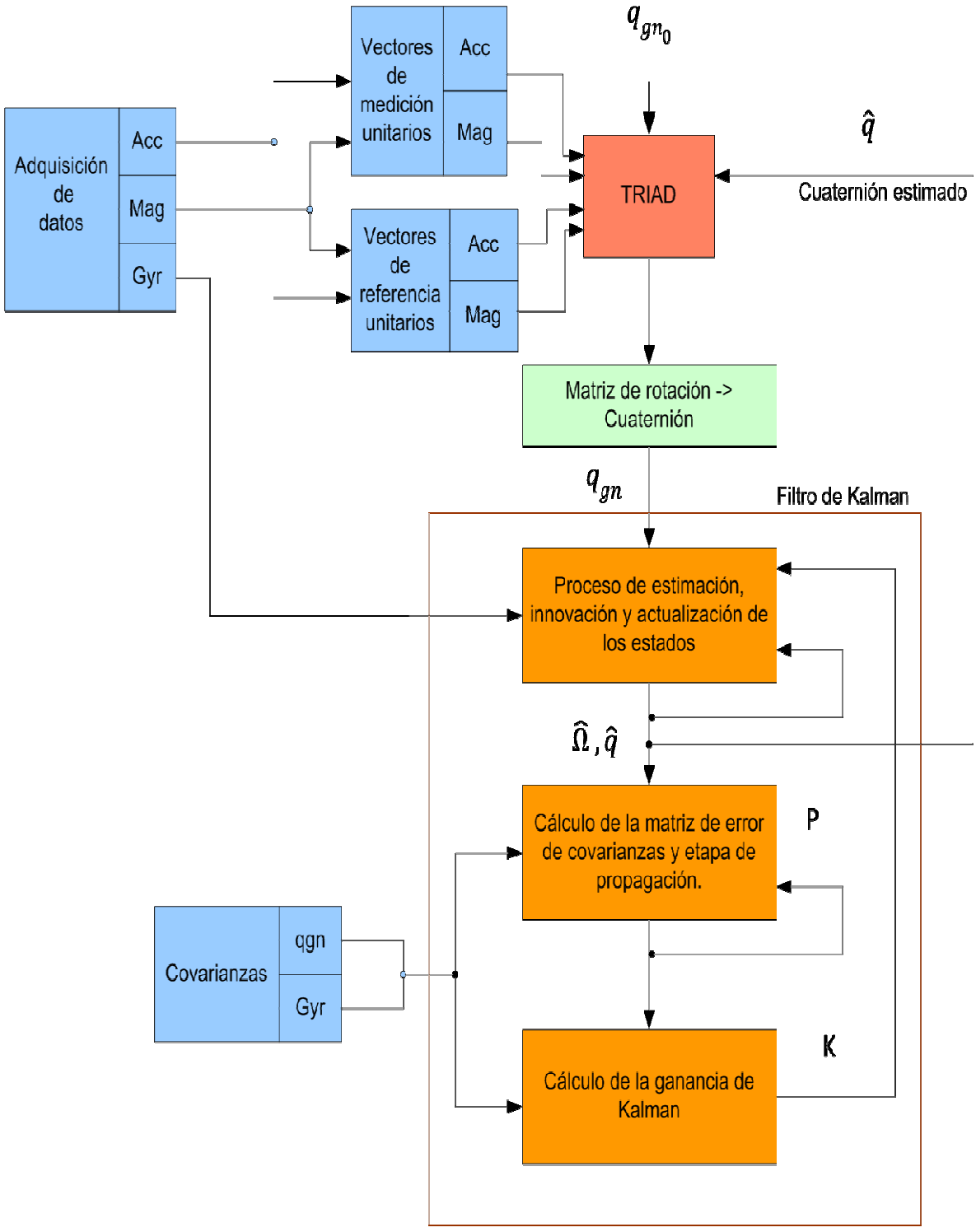


Figura 6.4 Distribución de los bloques que conforman la implementación de EKF en MATLAB (recuadros naranjas) y bloques externos con los que interacciona.

Este algoritmo es el centro de interacción de todo el subsistema que reside que en MATLAB, pues a través de él pasan los resultados de algoritmo TRIAD, y tiene como salidas finales los parámetros que han de pasarse al algoritmo de control.

6.6 Bloque de control

Dentro del bloque de coprocesamiento externo que considera el uso de la técnica HIL, ha sido desarrollado el módulo de control, con base en los recursos de programación que ofrece MATLAB para poder desarrollar una solución integral que requiere el sistema de control de orientación basado en el esquema HIL propuesto en esta tesis. Este bloque está conformado por un sistema de funciones que cumplen tareas específicas como la recepción de datos provenientes de la plataforma FPGA, la aplicación de los algoritmos de determinación utilizando métodos determinísticos como TRIAD , filtrado y estimación de la orientación mediante el uso del algoritmo EKF, así mismo considera un bloque FINAL que procese los datos de orientación dentro de una ley de control, por medio de la cual se generarán los comandos de control para los actuadores, mismos que son el ciclo de trabajo y el sentido de giro, los cuales serán transferidos a los circuitos integrados de potencia que controlan cada una de las tres ruedas inerciales instaladas sobre la plataforma de simulación.

La ley de control está basada en un modelo lineal de orientación en tres ejes [59], cuyos parámetros de entrada están contenidos dentro de las partes real e imaginaria de componentes del cuaternión de orientación estimado, obtenido del algoritmo EKF. El tipo de controlador utilizado es del tipo proporcional- derivativo (PD), cuya expresión se muestra en la ecuación **(6.19)**.

$$\boldsymbol{\tau_c} = -\boldsymbol{k_p}[\mathbf{3}\times\mathbf{3}]\bar{\boldsymbol{\varepsilon}}_{\boldsymbol{estimada}}[\mathbf{3}\times\mathbf{1}] - \boldsymbol{k_d}[\mathbf{3}\times\mathbf{3}]\bar{\boldsymbol{\omega}}_{\boldsymbol{estimada}}[\mathbf{3}\times\mathbf{1}] \qquad \mathbf{(6.19)}$$

Donde: $\boldsymbol{k_p}$ y $\boldsymbol{k_d}$ son las matrices de ganancias en cada uno de los tres ejes que considera una geometría cúbica circunscrita en la plataforma de simulación; $\bar{\boldsymbol{\varepsilon}}_{\boldsymbol{estimada}}$ es la parte vectorial del cuaternión estimado resultado del EKF, del que sólo se utilizan los tres primeros elementos, dejando fuera la parte escalar; $\bar{\boldsymbol{\omega}}_{\boldsymbol{estimada}}$ es la velocidad angular estimada del sistema.

Una vez obtenido el torque de control de la ley expresada en la ecuación **(6.19)**, este debe ser acondicionado de tal forma que pueda ser interpretado digitalmente y sea transmitido de regreso desde la PC externa hacia el sistema embebido en la plataforma FPGA, para, posteriormente, ser escrito en los registros de datos del núcleo correspondiente PWM para el control de actuadores. Por tal motivo los comandos de control deberán tener una longitud de 8 bits sin signo, adicionalmente se agregará un bit de signo para indicar el cambio de sentido de giro de las ruedas inerciales, en función de la posición relativa de los sensores de navegación inercial.

Cabe señalar que el proceso de acondicionamiento que realiza la conversión de torque a ciclo de trabajo y sentido de giro para PWM considera un factor máximo para el torque de los motores, el cual se considera a partir del dato de torque máximo consignado en las hojas de especificaciones del fabricante, que es de 0.7 [Nm], además de un factor de escalamiento respecto del porcentaje máximo para el ciclo de trabajo, en este caso se ha decidido manejar entre un 60 y 70 por ciento del ciclo de trabajo máximo. En este mismo proceso, se determina el sentido de giro para lo actuadores basado en el signo obtenido del par de control, donde (-) representa un giro en sentido horario y (+) el sentido antihorario.

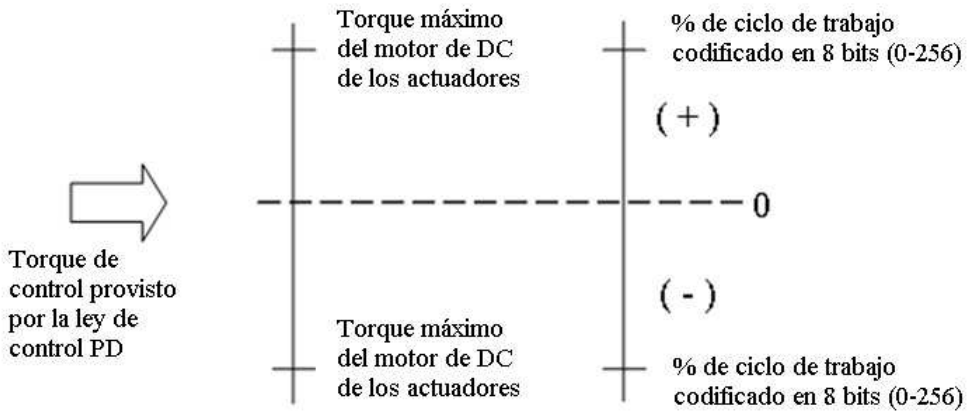


Figura 6.5 Relación de los torques máximos respecto al ciclo de trabajo de los actuadores.

Más adelante se presentan algunas pruebas experimentales realizadas con los actuadores que permitieron caracterizar su comportamiento evaluando su desempeño y comprobar cuál puede ser el torque máximo inducido sin que haya saturación en los motores, permitiendo que así sea especificado en el programa principal y evitar comportamiento inesperado o innecesario cuando esté la simulación ejecutándose.

La Figura 6.6 muestra la descripción de la estructura del bloque de control:

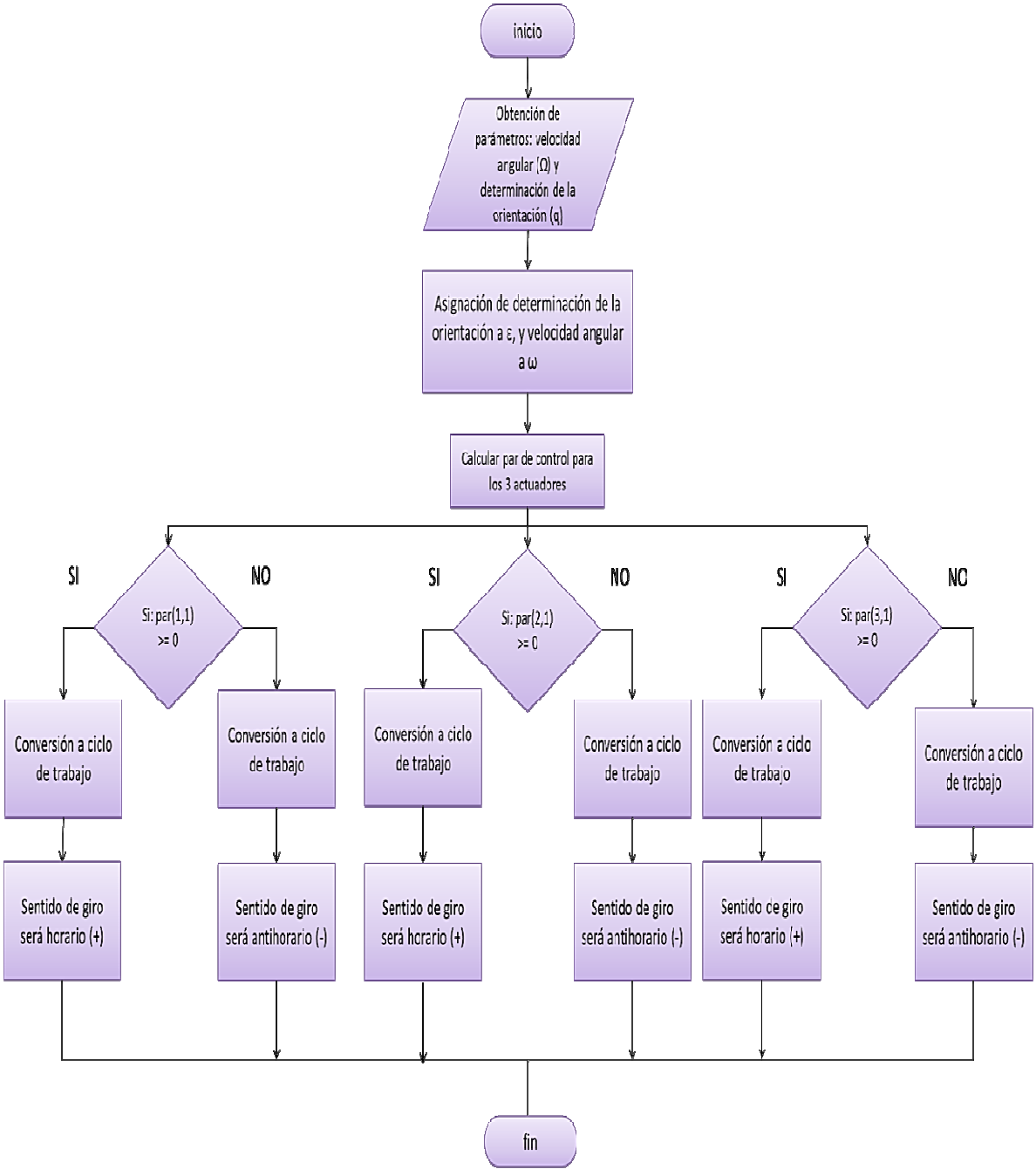


Figura 6.6 Diagrama de flujo de cómo se implementó el algoritmo de control en MATLAB.

6.7 Implementación en MATLAB

El subsistema de control implementado en MATLAB está formado por archivos .m o *scripts* de funciones que incluyen los algoritmos TRIAD, EKF y CONTROL, todos ellos interaccionan mediante las salidas obtenidas de cada bloque de función. Realizan diversas tareas como obtener la matriz de rotación y calcular el cuaternión de orientación de la MSA a partir de las mediciones provenientes de los sensores de navegación, también se lleva a cabo el filtrado y la corrección del cuaternión de orientación del sistema y la medición de la velocidad angular del sistema.

Finalmente, como se muestra en la Figura 6.6, mediante la ley de control se obtienen los comandos de control expresados como ciclos de trabajo y sentidos de giro para cada uno los actuadores instalados en los tres ejes de la plataforma del simulador. Este bloque o segmento del sistema cuenta con un programa principal, que lleva a cabo la configuración del puerto serie para la adquisición de datos, ejecuta un ciclo de iteraciones que puede ser infinito o definir un valor específico de iteraciones o muestras para la recepción de los datos provenientes de la plataforma SPARTAN 3E, a los cuales da formato y coloca en arreglos unidimensionales para posteriormente hacer las llamadas a las funciones y transferir los datos necesarios a cada uno de los algoritmos implementados.

Mediante las salidas de cada función, al final se obtienen los comandos de control que serán enviados por el mismo puerto vía módem de RF hacia la tarjeta SPARTAN 3E sobre la MSA (Figura 6.7**)**. Adicionalmente se cuenta con la opción de almacenar los resultados de alguna prueba operativa en archivos de texto planos, para su posterior análisis y generación de gráficas, por medio de las cuales es posible evaluar el desempeño de las pruebas desarrolladas.

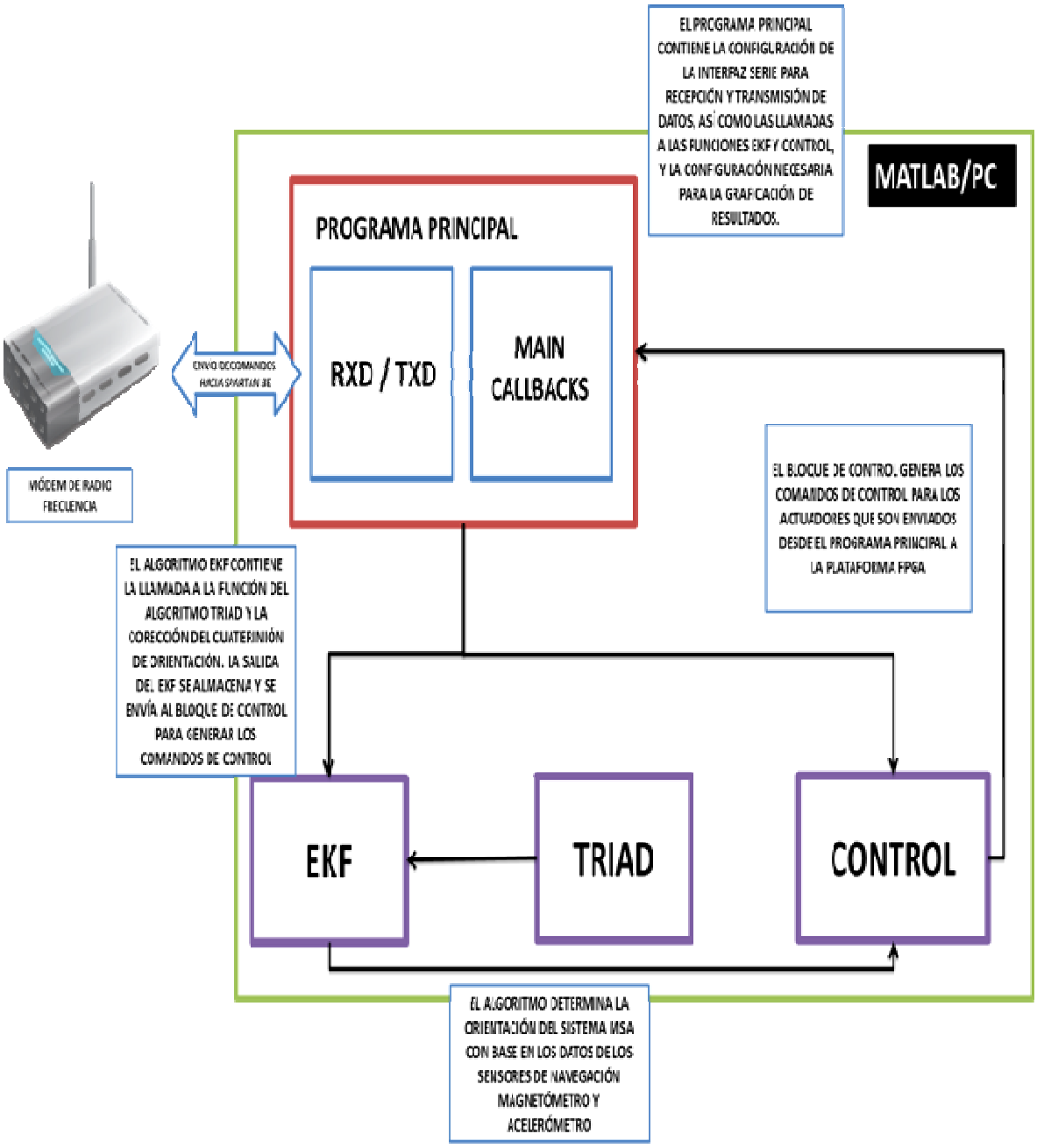


Figura 6.7 Diagrama del subsistema de procesamiento implementado en MATLAB.

Como se comentó anteriormente, se ha realizado un experimento con el que se pretende caracterizar los actuadores, motores y ruedas inerciales para obtener algunas características importantes, como la velocidad angular de los actuadores, y el torque máximo inducido a los mismos sin que haya saturación.

Este experimento se realizó con base en un estroboscopio modelo SF-911 Pasco Scientific para determinar la velocidad angular de las ruedas inerciales en revoluciones por minuto (RPM), proponiéndose tres relaciones para mostrar los resultados: la primera, de ciclo de trabajo (*duty cicle*) inducido a los motores contra la velocidad angular generada; la segunda, de voltaje inducido a los motores contra la velocidad angular generada; y finalmente, se obtuvo una tabla que expone la velocidad angular obtenida experimentalmente, su equivalente en [rad/s] y el momento angular generado por dicha velocidad.

Siendo los valores de voltaje y porcentaje de ciclo de trabajo propuestos como una serie de valores constantes, para determinar la velocidad angular se procedió así: Se tiene una línea que parte del centro hacia la periferia de la rueda inercial, dibujada sobre su superficie. Se sabe que el estroboscopio funciona con base en la persistencia de imágenes sobre la retina del ojo, para ello, utiliza una lámpara de gas cual *flash* de alta intensidad; con un dispositivo como este se puede ajustar la frecuencia de flash por minuto de forma manual hasta que la posición de la rueda inercial permanece estacionaria a la vista, entonces puede verificarse directamente en el dispositivo la velocidad equivalente en revoluciones por minuto (Figura 6.8.).

Figura 6.8 Prueba realizada en el laboratorio para determinar de la velocidad angular con ayuda de un estroboscopio.

La tabla de voltaje contra velocidad angular (Figura 7.3) se obtuvo a partir del voltaje suministrado con una fuente variable, mientras el intervalo dinámico de valores se obtuvo en función de las especificaciones técnicas del motor las cuales especifican máximo un máximo de 18 [V].

CICLO DE TRABAJO (% REAL-% REDONDEADO)	VELOCIDAD ANGULAR [RPM]
29.41 - 30	678
39.21 - 40	1044
50.1 - 50	1346
153 - 60	1481
178.5- 70	1647
204 - 80	2363

Figura 6.9 Tabla de relación de ciclo de trabajo contra la velocidad angular de la ruedas inerciales.

VOLTAJE [V]	VELOCIDAD ANGULAR [RPM]
3	840
5	420
6	1375
7	1584
8	1794
9	2074
10	2290
11	2480
12	2708
13	2920
14	3150
15	3393

Figura 6.10 Tabla de relación de voltaje contra la velocidad angular de la ruedas inerciales.

Ahora, a partir de los resultados, haciendo uso de la definición de la cantidad de movimiento angular **L** de una partícula, la cual está dada por el producto de su cantidad de movimiento lineal (***p* = *mv***) por la distancia perpendicular que va del eje a la partícula que gira.

$$\boldsymbol{L = m\,v\,r} \qquad \textbf{(6.20)}$$

Donde:

L, es la cantidad de movimiento angular.
m es la masa de la partícula.
v es la velocidad línea de la partícula.
r es la distancia perpendicular que va del eje a la partícula que gira.

Considerando la rueda inercial como un cuerpo rígido extenso, girando en torno a un eje, cada partícula de la rueda inercial describe una cantidad de movimiento angular definida en la ecuación **(6.20)**. Si la velocidad lineal está dada por $\boldsymbol{v = \omega r}$ **y** remplazamos en **(6.20)**, se tiene que todas sus partículas tienen la misma velocidad angular **ω**, y la cantidad de movimiento angular del cuerpo está dada por:

$$\boldsymbol{L = (\sum mr^2)\omega} \qquad \textbf{(6.21)}$$

Siendo que $(\sum \boldsymbol{mr^2})$ es el momento de inercia, la cantidad de movimiento puede expresarse como:

$$\boldsymbol{L = I\omega} \qquad \textbf{(6.22)}$$

Donde: **L** es la cantidad de movimiento angular en [kgm^2/s]. **I** es el momento de inercia en kgm^2, y **ω** es la velocidad angular en [rad/s].

Finalmente, la velocidad angular en RPM debe expresarse en [rad/s], para ello, se aplica este procedimiento: Primero se calcula la velocidad angular a partir de la frecuencia, como se ilustra en este ejemplo:

$$\boldsymbol{840\frac{rev}{min} \times \frac{1min}{60s} = 14\ \frac{rev}{s} =} \qquad \textbf{(6.23)}$$

Entonces:

$$\boldsymbol{\omega = 2\pi f = (2)(\pi)\left(14\frac{rev}{s}\right) = 87.964\frac{rev}{s}} \qquad \textbf{(6.24)}$$

Si de las especificaciones de la rueda inercial se sabe que su inercia es $\boldsymbol{I = 0.0038\ kgm^2}$, por lo tanto, su cantidad de momento angular es:

$$\boldsymbol{L = I\omega = (0.0038\ kgm^2)\left(87.964\frac{rev}{s}\right) = 0.334\frac{kgm^2}{s}} \qquad \textbf{(6.25)}$$

De esta manera se obtuvo la siguiente tabla, correspondiente a las revoluciones calculadas para la Figura 6.11.

VELOCIDAD ANGULAR [RPM]	VELOCIDAD ANGULAR [RAD/S]	MOMENTO ANGULAR [KGM^2/S]
840	14.00	0.05
420	7.00	0.03
1375	22.92	0.09
1584	26.40	0.10
1794	29.90	0.11
2074	34.57	0.13
2290	38.17	0.15
2480	41.33	0.16
2708	45.13	0.17
2920	48.67	0.18
3150	52.50	0.20
3393	56.55	0.21

Figura 6.11 Tabla del momento angular generado por la velocidad angular de la rueda inercial.

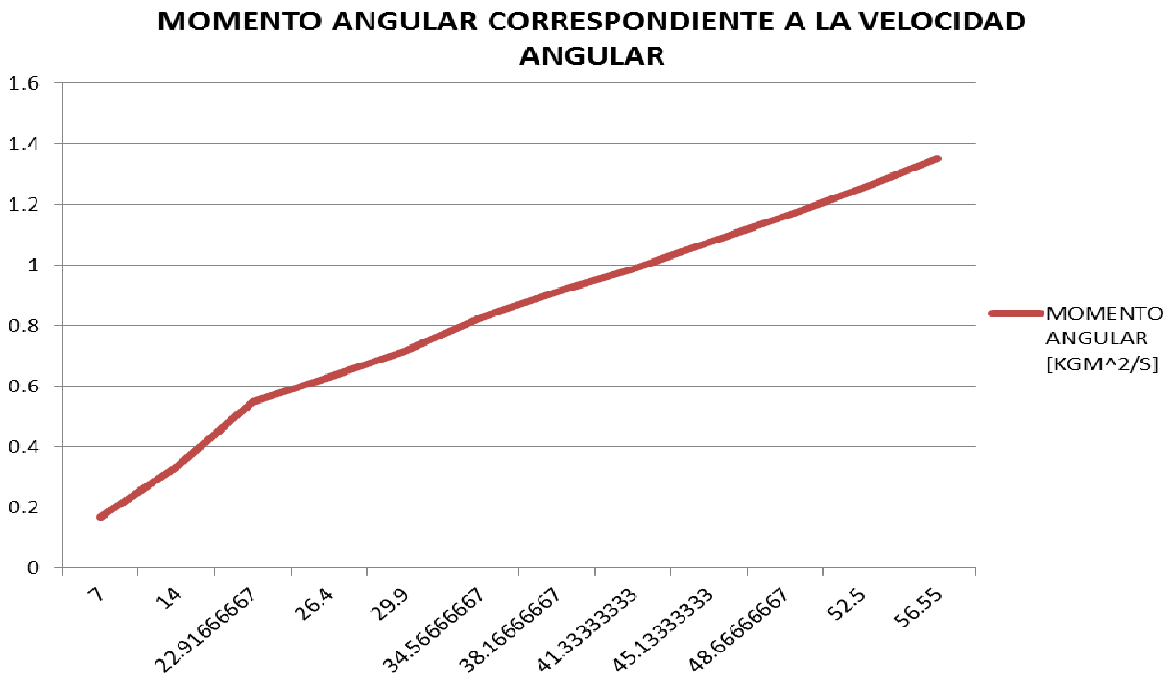


Figura 6.12 Gráfica del momento angular generado por la velocidad angular de una rueda inercial.

Los resultados obtenidos y mostrados en las tablas son una forma de observar claramente el desempeño de las características de los actuadores, desde su consumo y potencia, hasta la fuerza que genera de acuerdo a su velocidad. Los resultados describen cuantitativamente la variación que se puede presentar cuando el sistema trabaja en su totalidad. En el caso del porcentaje del ciclo de trabajo sólo se hizo la medición hasta un 80% del desempeño máximo, y corresponde al que se maneja en todas las pruebas del sistema, con el fin de evitar la saturación magnética provocada por el consumo de corriente eléctrica. En la gráfica de la Figura 6.12 se muestra el comportamiento del momento angular que se puede obtener para cada rueda, y que se considera para evaluar la capacidad del sistema para realizar las maniobras con los actuadores actuales.

6.2 Sistema totalmente integrado en dos bloques

El sistema de simulación satelital para la validación de esquemas de control consta de dos sistemas que interaccionan entre sí en tiempo real y su funcionamiento es coherente como si sólo se tratara de un solo sistema totalmente integrado a bordo de la plataforma, esto se ha logrado con el uso de la técnica de cosimulación denominada HIL. Para ello, las comunicaciones se han llevado a cabo por medio de dos radios módem (RF) que comunican a la MSA con la PC/MATLAB que realiza el coprocesamiento de datos; dichas comunicaciones se han configurado a una tasa de transmisión de 9600 bps en ambas partes, la cual fue fijada con base en pruebas experimentales realizadas con el sistema, do se comprobó el impacto de realizar transmisiones a una mayor tasa, siendo ésta la más óptima para realizar el procesamiento y envío y recepción de datos entre la MSA y la PC sin tener pérdida de información e interferencias por el ruido del funcionamiento del sistema, e incluso que el proceso se interrumpa, porque la velocidad de transferencia es más rápida de lo que se requiere entre una interrupción y otra en la plataforma FPGA, por lo que el buen desempeño se vería afectado. Por otro lado, el sistema se compone de bloques de programa y diversos dispositivos, y los dos subsistemas ha sido implementados en diferentes plataformas: en una FPGA SPARTAN 3E, que contiene la parte de generación y administración de interrupciones para la adquisición de datos de los sensores de navegación, el manejo del generador PWM para el control de actuadores, la interfaz de comunicaciones con los sensores de navegación mediante el uso del protocolo I2C, así como la parte de la transmisión y recepción de datos seriales entre la plataforma MSA y la PC externa que ejecuta el sistema de funciones que componen el esquema de control de orientación y control en MATLAB. Una imagen que muestra la integración del sistema de control de orientación se muestra en la Figura6.13.

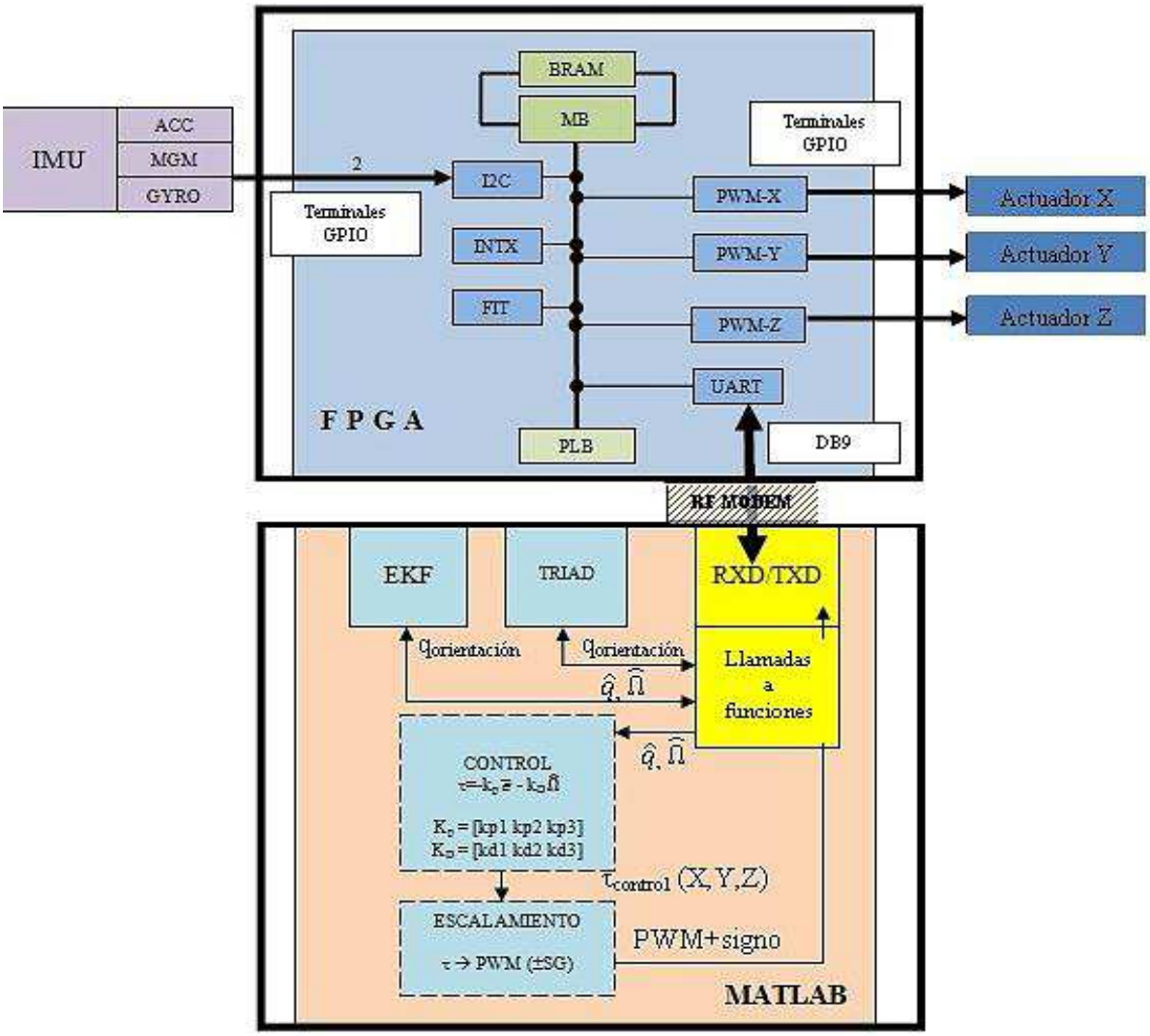


Figura 6.13 Módulos de la plataforma de simulación satelital MSA

Un sistema con estas características, es decir, basado en la técnica HIL, tiene como principal objetivo acelerar el desarrollo y la implementación para la obtención de resultados, pero además tiene otras ventajas importantes como reducir los costos de desarrollo, pues otro de los objetivos que se persiguen son los desarrollos de bajo costo, en este caso al simular el funcionamiento del sistema con sensores reales y coprocesamiento fuera de la plataforma, pueden analizarse la cantidad de recursos que requeriría un sistema real de este tipo, aportando a la decisión en la adquisición de hardware que soporte los requerimientos necesarios o elegir un producto más sofisticado pensando en un desarrollo que integre más módulos, considerando un escalamiento del sistema, además de la corrección y validación de software antes de desarrollar un modelo real.

No obstante, a pesar de la flexibilidad que tiene el sistema y el mismo desarrollo en la plataforma FPGA a bordo de la MSA, se deben analizar otros aspectos que se originan por la naturaleza propia de este particular, donde se tuvieron que resolver una serie de problemas que se dieron durante el desarrollo e implementación de todo el sistema, como fueron: las comunicaciones inalámbricas; evaluación de las plataformas de implementación de coprocesamiento, donde inicialmente su utilizaría SIMULINK para este fin, pero por causas anteriormente explicadas, sobre todo en la pérdida de información, se decidió implementar los algoritmos sobre MATLAB; documentación y utilización de herramientas de software y hardware para el uso y manejo de los diversos dispositivos que componen la plataforma MSA; entre otras..

También, el sistema cuenta otros componentes primordiales para su funcionamiento, se tienen los actuadores e instrumentación, ruedas inerciales, sensores de navegación, batería a bordo de la MSA para suministro de dispositivos y actuadores, así como un dispositivo con los *drivers* para el control de corriente de los motores de las ruedas inerciales. Finalmente, atendiendo al trabajo futuro que seguirá desarrollando y mejorando el sistema, deben considerarse dos aspectos importantes, el proceso de desarrollo y la complejidad, la cuales seguirán creciendo en cuanto a requerimientos y recursos. Debe evaluarse, por tanto, el hardware, pues uno de los objetivos es que todo el sistema lógico y de procesamiento vaya a bordo de la MSA, entonces, hay que considerar si es suficiente una plataforma FPGA, incluso si lo es con la que se desarrollo esta tesis; en cuyo caso, considerando la complejidad del sistema, debe ser necesario adquirir un dispositivo FPGA, que cuente con mayores recursos.

El siguiente apartado es una presentación de los resultados obtenidos en este trabajo de tesis, donde se expone una parte de resultados basados en el comportamiento de la plataforma de simulación satelital MSA, y se presenta un breve apartado de los productos obtenidos del desarrollo de dicha plataforma.

Capítulo 7

Resultados experimentales

El sistema de simulación para la validación de esquemas de control de orientación satelital (MSA), permite validar estrategias de control de orientación en tres ejes en tierra. La primera versión de implementación física del sistema está basada en la técnica *hardware in the loop* (HIL), la cual permite acelerar el desarrollo e implementación en un modelo físico para obtener resultados en un menor lapso de tiempo, además de facilitar la validación de diversos componentes y equipos que la integran. El sistema MSA se compone de dos bloques principales, los cuales interaccionan mutuamente comportándose como si se tratara de un sistema completo, lo cual además de acelerar la obtención de resultados experimentales, permite la depuración de hardware y software de un sistema tan importante en vehículos espaciales como el subsistema de control de orientación. En este capítulo se presentan los resultados obtenidos en el desarrollo de la primera aproximación de la integración del sistema MSA. En esta descripción de resultados se presenta una serie de gráficas significativas de todas las pruebas realizadas en las que se fueron realizando algunos ajustes para la sintonización de los parámetros en los algoritmos que influyen directamente en el comportamiento del sistema. También se realiza una descripción breve de la metodología empleada para la realización de pruebas, así como una discusión de los resultados obtenidos y esperados.

7.1 Introducción

La integración del sistema de simulación satelital para la validación de esquemas de control ha sido realizada con base en la implementación de una serie de módulos con tareas específicas que se encuentran distribuidos en dos segmentos principales: PLATAFORMA MÓVIL (MSA) y PC externa. Los detalles de estos módulos ya han sido descritos en capítulos anteriores de este trabajo, los cuales han sido desarrollados con una metodología propia (Figura 7.1), la cual integra los siguientes puntos fundamentales: planteamiento inicial y requerimientos del sistema que se ha desarrollado; análisis y diseño de los módulos que integran al sistema (equipo y material) así como las pruebas realizadas para su validación parcial; la integración de todos los módulos desarrollados y finalmente, las pruebas de validación del sistema integrado y la discusión de los resultados obtenidos.

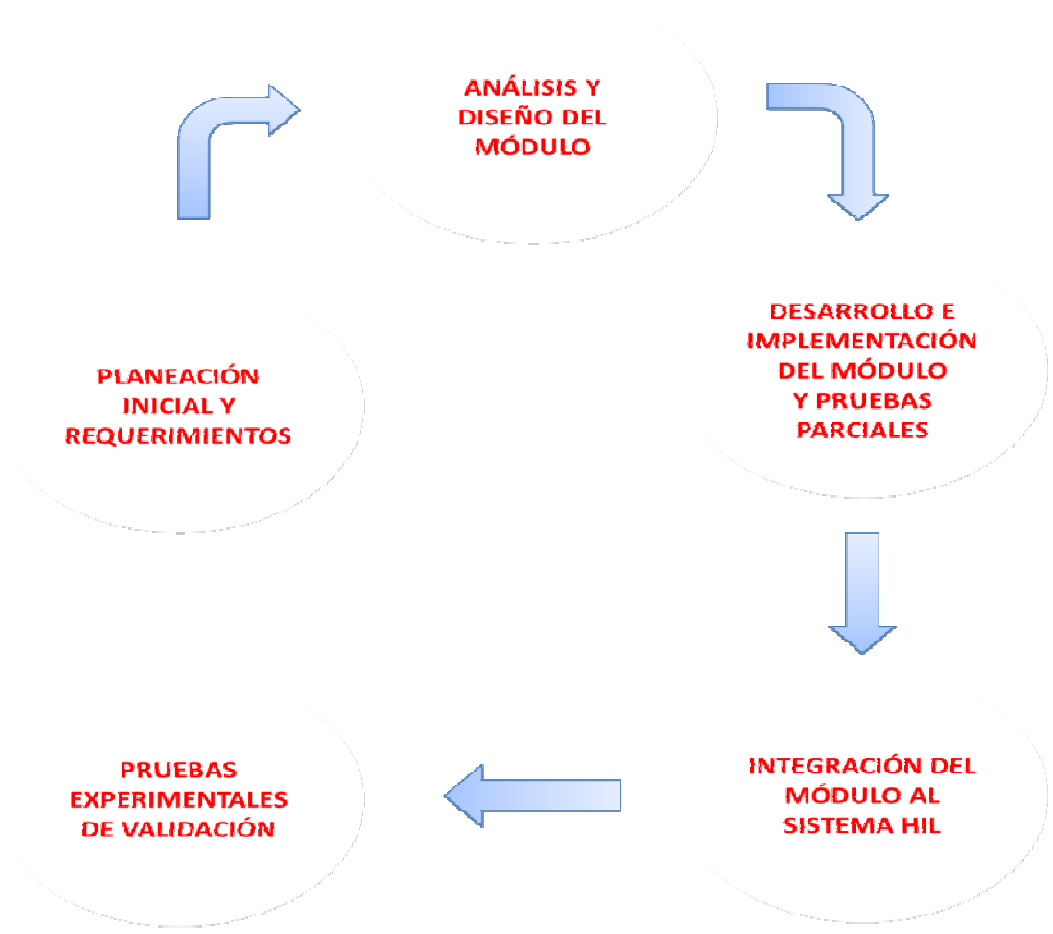


Figura 7.1 Flujos de tareas para la integración de módulo al sistema HIL.

Esta metodología ha arrojado el producto final que es la integración del sistema el cual tiene por objetivo validar el esquema de control basados en la MSA. La validación en tierra es fundamental pues una vez que el satélite o vehículo aeroespacial ha sido puesto en órbita no puede ser reparado o modificado, por lo que validar su funcionamiento en tierra, de forma parcial durante el desarrollo y completamente al final del mismo, resulta ser una gran ventaja pues se está logrando una optimización y corrección continuas del sistema con la finalidad de reducir el riesgo de alguna falla ya sea de hardware o software. Esta validación se ha llevado a cabo mediante pruebas experimentales en laboratorio, en las que se realizan las configuraciones y conexiones previas de todo el arreglo experimental del sistema.

7.2 Arreglo experimental

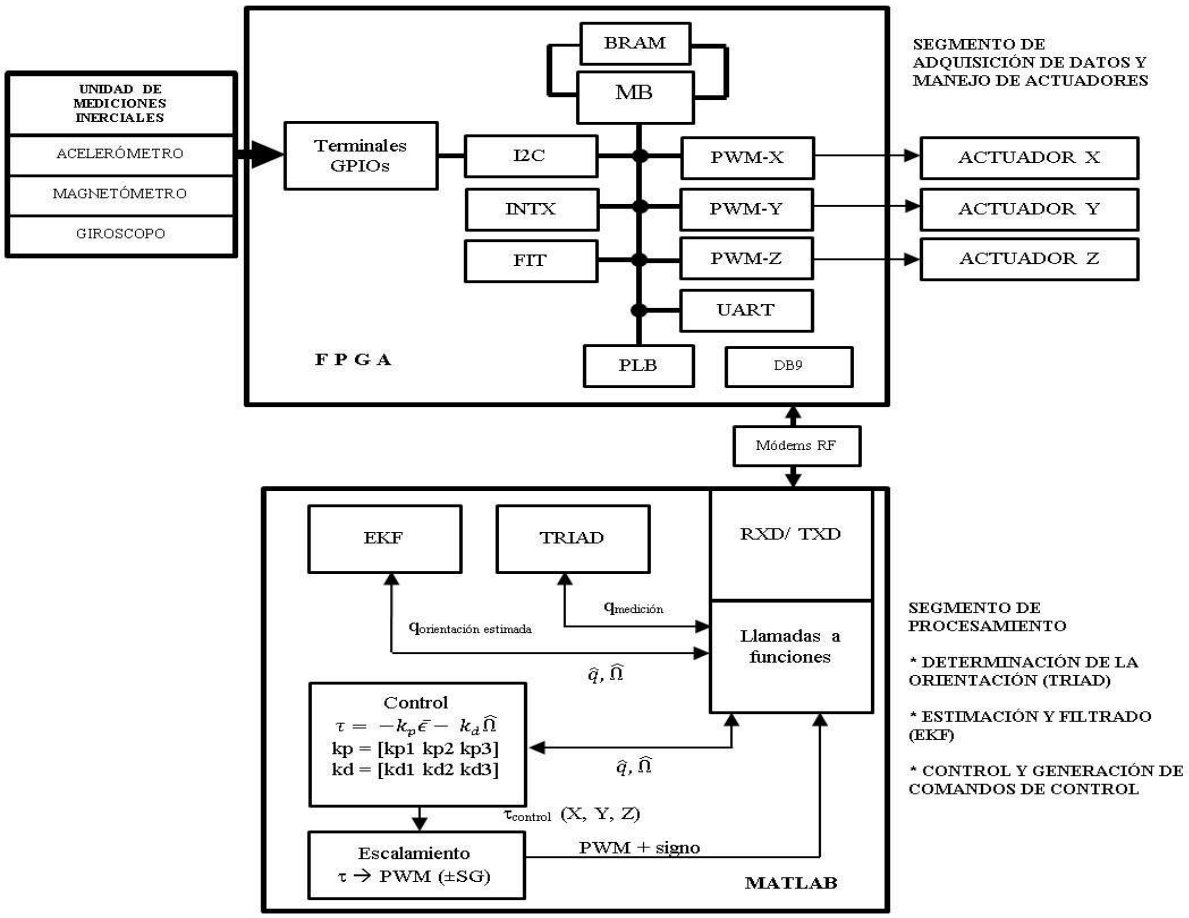


Figura 7.2 Diagrama lógico de los componentes del sistema de simulación satelital para la validación de esquemas de control.

El arreglo experimental es todo el conjunto de componentes del sistema de simulación satelital para la validación de esquema de control basado en una plataforma de desarrollo FPGA, esto quiere decir que son diversos dispositivos y componentes a bordo de la MSA y externos a ella, como se describe a continuación. Como se ha dicho, el sistema se integra por dos segmentos principales, los cuales están comunicados e interaccionan de forma inalámbrica, por medio de dos radios módem, uno en cada bloque. Como puede verse en la Figura 7.2, el segmento de adquisición de datos y manejo de actuadores lleva a bordo una plataforma de desarrollo FPGA que mediante la gestión de interrupciones realiza la lectura de los datos de los sensores de navegación, les da formato ASCII a esos datos y los envía hacia el segmento de coprocesamiento, además, recibe los comandos de control y los escribe en los respectivos registros de cada núcleo PWM para el manejo de los actuadores. Mientras que el segundo segmento, el de coprocesamiento, implementado en MATLAB y ejecutándose sobre una PC, realiza las tareas gestión de puerto serie para las comunicaciones con el primer segmento; determinación de la orientación del sistema mediante el algoritmo TRIAD, así como la estimación y filtrado de la misma; la medición de la velocidad angular del sistema; así como el cálculo del torque en cada uno de los ejes, de los cuales se realiza una conversión para determinar el signo y el ciclo de trabajo y ser enviados como comandos de control hacia la plataforma FPGA; todo esto es gestionado por un programa principal (Figura 7.3).

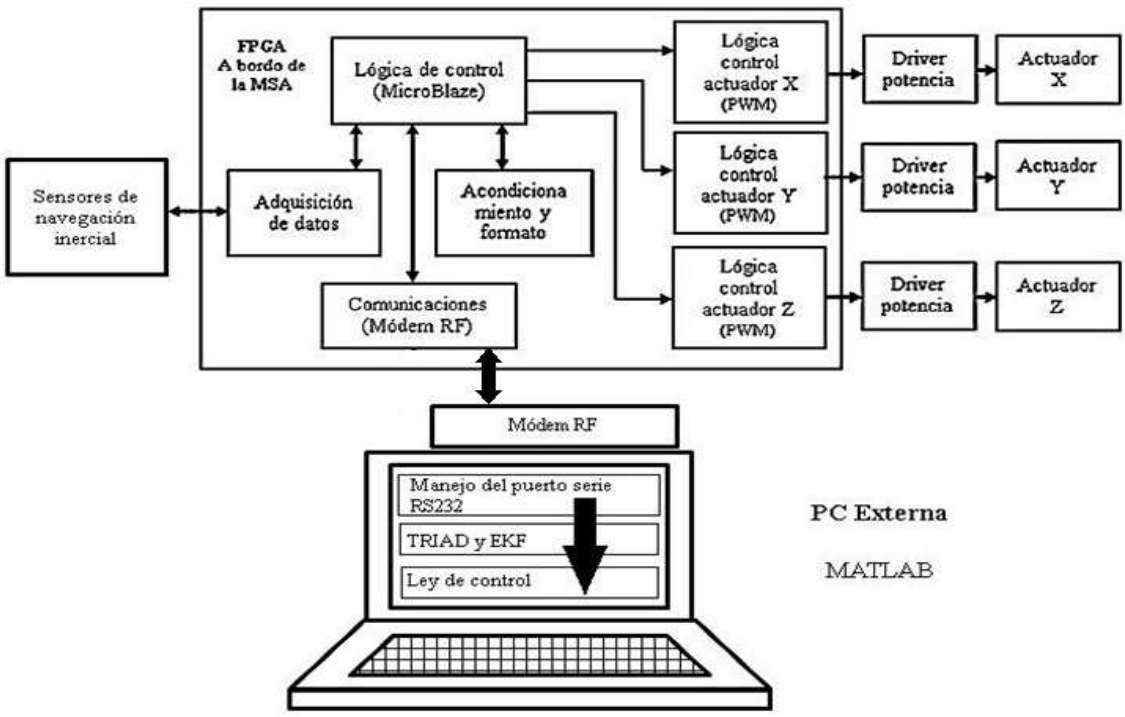


Figura 7.3 Diagrama de bloques de la interacción entre los segmentos que componen al sistema de simulación satelital para la validación de esquemas de control.

Todos estos elementos operan bajo un complejo esquema de interacción y su funcionamiento se puede considerar como transparente durante las pruebas, no obstante, antes de comenzar éstas se requiere llevar a cabo diversas tareas como: calibración y revisión de la configuración vía software de los sensores de navegación inercial y las conexiones con la plataforma SPARTAN 3E de las diversas líneas de datos y alimentación; revisión de velocidad de transmisión los radios módem para las comunicaciones inalámbricas; revisión de las conexiones de alimentación para la plataforma SPARTAN 3E, radios módem y actuadores; y la revisión de las conexiones de las salidas de señales PWM hacia los *drivers* de potencia de los actuadores y la del puerto serie DCE de la plataforma SPARTAN 3E. Mientras, por otro lado se verifica y revisa la conexión del cable para la descarga del firmware desde la PC hacia la plataforma SPARTAN 3E, y se valida la conexión del puerto serie de la PC para las comunicaciones inalámbricas; por último, se definen los parámetros de operación (valor de constantes en algoritmos, ganancias de controladores) antes de comenzar las pruebas.

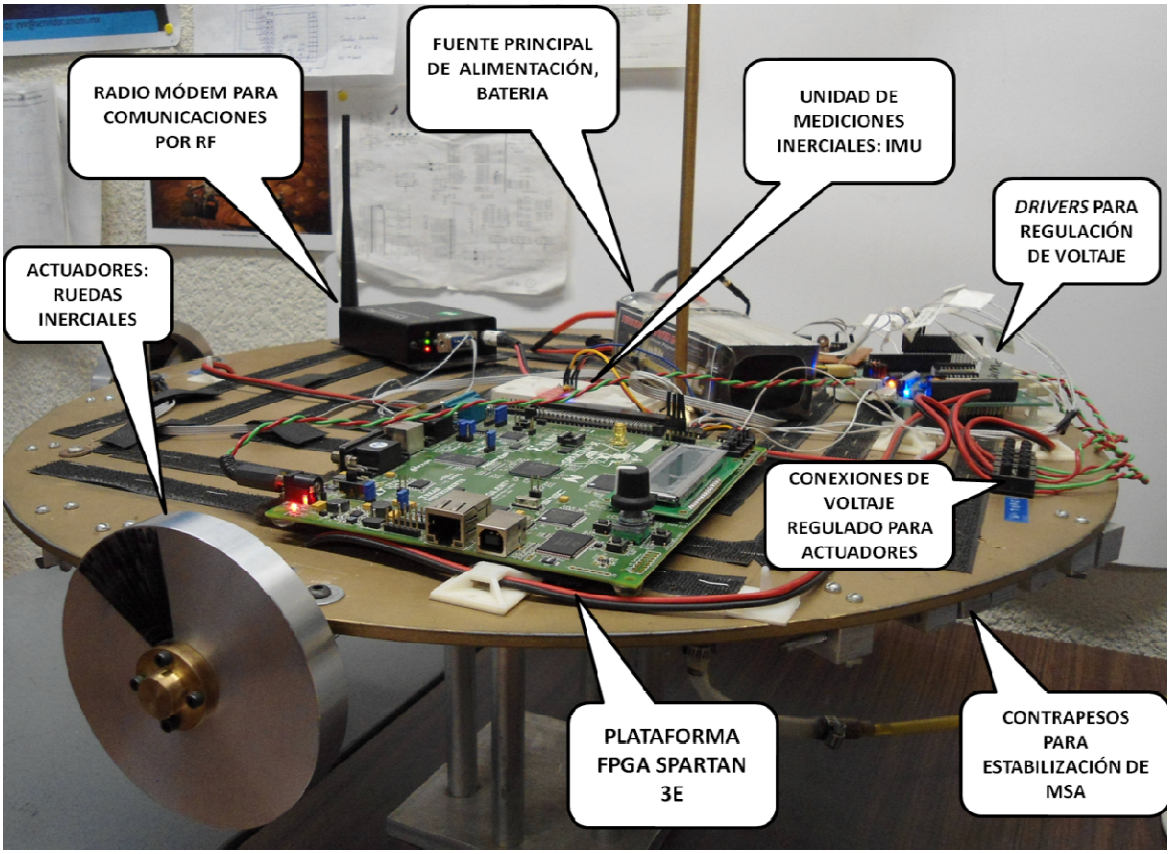


Figura 7.4 Elementos a bordo de la MSA.

En cuanto a los segmentos del sistema, Por un lado se tiene el segmento a bordo de la MSA, el cual contiene la mayor cantidad de dispositivos y equipo electrónico; y por otro lado, se tiene el segmento en la PC externa que incluye adicionalmente un radio módem que permite la transferencia de datos de forma inalámbrica con el segmento MSA. Con base en esta descripción, se dará hará una revisión de dichos elementos de forma visual. La MSA (Figura 7.4) lleva a bordo un esquema de instrumentación que incluye los siguientes componentes y equipos: plataforma de desarrollo FPGA SPARTAN 3E de Xilinx©, actuadores (motores y ruedas inerciales), sensores de navegación inercial (magnetómetro, acelerómetro y giróscopo), radio módem (dispositivo para comunicaciones por radiofrecuencia), batería de alimentación, contrapesos de estabilización y *drivers* para el control de potencia de los actuadores.

En el desarrollo e integración de este esquema de instrumentación subsistema destacan las pruebas realizadas para determinar la velocidad angular de las ruedas inerciales con objeto de caracterizar su desempeño operativo y aprovecharlo en la definición de la ley de control, las cuales fueron fabricadas en los talleres mecánicos del IINGEN-UNAM, y que fue explicado en el capítulo anterior. El otro segmento, el bloque de coprocesamiento desarrollado fuera de la MSA, se compone de una PC y un radio módem que permite las comunicaciones inalámbricas, envío y recepción de datos entre ambos segmentos, en la banda de los 900 MHz con una potencia de 100 mw (Figura 7.5).

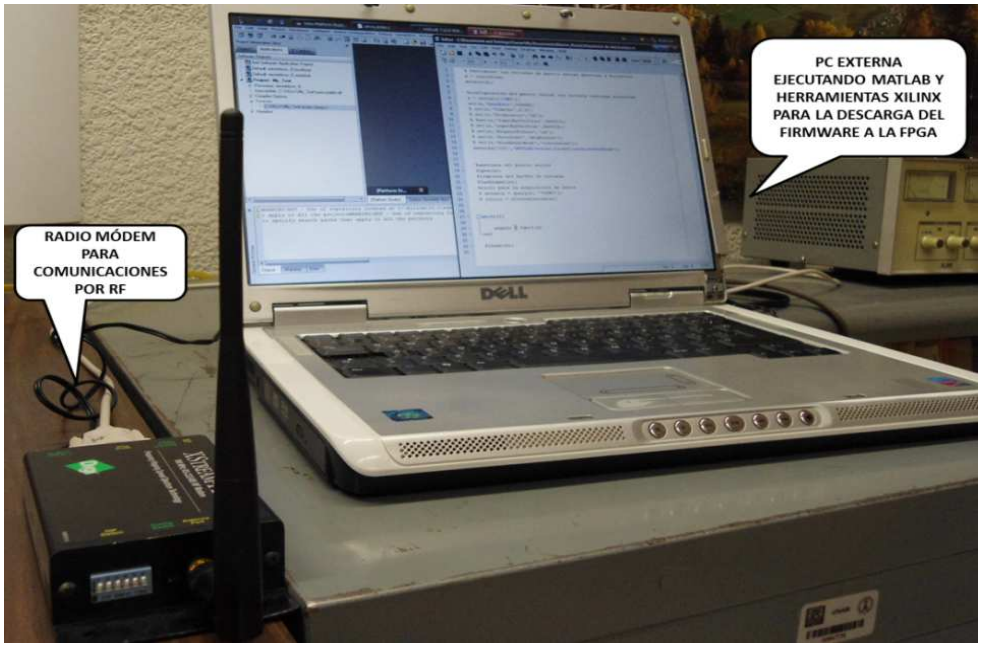


Figura 7.5 PC externa y radio módem.

Adicionalmente a este equipo y componentes electrónicos, se cuenta con una tiene una compresora de aire, cuyo flujo de aire de salida se hace pasar por un banco de filtros (coalescente y desecante) para retirar la agua y residuos de combustible (aceite principalmente) orgánicos en el mismo, que proporciona el aire para simular las condiciones de no fricción del espacio, donde un aire limpio de impurezas no simula la condición de no fricción propiamente dicho, lo que permite es mejorar el desempeño del simulador en términos de contar con un flujo de aire constante, ya que además evita el taponamiento de los capilares, donde se genera el flujo que crea el colchón sobre el que flota la plataforma móvil de la MSA. El flujo de aire se inyecta a través de una manguera conectada en la base de la MSA y mediante este mecanismo se crea un colchón de aire que simula dichas condiciones. Todo esto es necesario para poner en funcionamiento el sistema y crear las condiciones para realizar el experimento y validación del sistema. Y una vez CON que se tiene el material y equipo necesario, también existe una metodología para realizar las pruebas de validación de sistema. Esta metodología se adoptó con el fin de tener un procedimiento para supervisar las interconexiones de los dispositivos y tomas de alimentación para garantizar el funcionamiento correcto de todos los componentes y reducir la posibilidad de un fallo que pudiera ser un costo adicional no previsto.

7.3 Metodología para la realización de pruebas

La preparación del arreglo experimental para realizar las pruebas del sistema de simulación satelital para la validación de esquemas de control se llevó a cabo con el objetivo de realizar estas en las condiciones óptimas y reduciendo la posibilidad de posibles fallos en las interconexiones y alimentación del equipo y dispositivos electrónicos que componen al sistema. De esta forma, se reduce la posibilidad que si se presenta algún comportamiento anómalo o algún fallo del sistema, se deba a factores de interconexión, y con ello, enfocarse en mayor medida a la sintonización de parámetros de los algoritmos de procesamiento. Para la revisión de las conexiones entre dispositivos, se verifican inicialmente las dos tomas principales, una de 5 [V] para alimentar la plataforma FPGA, y unas más, para alimentar los actuadores con 15 [V] y el radio módem a bordo de la MSA. También se revisan las conexiones necesarias para las comunicaciones inalámbricas y para la descarga del firmware desde la PC a la plataforma SPARTAN 3E.

Con la misma importancia, hay una serie de ajustes que se realizan en el sistema que son determinantes en el comportamiento del sistema e influyen directamente en el resultado y están presentes, y son modificables, en todas las pruebas; estos son:

- El balance manual de la mesa. Este balance es establecer una posición inicial de la plataforma MSA, a partir del cual, y con base en el sistema de referencia, la plataforma debe apuntar en una posición determinada prescindiendo de esa posición.

- Ajustes en el sistema de referencia. El algoritmo TRIAD implementado, incluye dos valores de referencia de los sensores, es decir, las coordenadas, X, Y, Z de un punto. Un punto calculado a partir del promedio de las mediciones del magnetómetro, y un punto elegido por convención de las condiciones físicas de la tierra, en este caso del valor de la gravedad, para el acelerómetro. Estos valores pueden ser cambiados dependiendo del comportamiento que se espere, en este caso, el cero del sistema de referencia corresponde con el de la MSA.

- Sintonización de parámetros. Hay otros parámetros que se deben ajustar para mejorar el comportamiento y desempeño de la MSA, éstos son las covarianzas de error mencionadas en el algoritmo EKF y las matrices de ganancias del algoritmo de CONTROL. Las primeras se ajustan para reducir el desfasamiento de los resultados del TRIAD y el EKF, así como la reducción de ruido en la medición de la velocidad angular del sistema; la segundas se ajustan para mejorar la respuesta de los actuadores para controlar y estabilizar la MSA.

Establecida la metodología para realizar las pruebas, se muestran a continuación la discusión sobre los resultados obtenidos a partir del comportamiento de la MSA descrito gráficamente.

7.4 Descripción de la Pruebas experimentales

En este apartado se presentan los resultados gráficos de las pruebas experimentales, haciendo una descripción del comportamiento que se observó de acuerdo con la comparación entre los algoritmos TRIAD y EKF, el intervalo de valores de los sensores de navegación, los valores de torques obtenidos en cada uno de los ejes de la MSA, así como los valores obtenidos del giróscopo y la medición de la velocidad angular por parte del algoritmo EKF. Además se realiza una discusión breve de acuerdo a los resultados obtenidos y lo que se esperaba con base en el comportamiento gráfico.

Las condiciones para cada una de las pruebas han sido a partir del movimiento, es decir, se indujo una perturbación a la MSA generando un movimiento que debía ser amortiguado, actuando el control derivativo para realiza el frenado, y una vez que el sistema lo conseguía, intentar realizar el apuntamiento mediante la actuación del control proporcional. Aunque los resultados muestran el comportamiento del caso mencionado, se hicieron pruebas respectivas desde el reposo del sistema, donde se realizaba en mayor parte una labor de apuntamiento, independientemente de la posición inicial, por lo que se consideró presentar el caso en el que había movimiento y observará la intervención las dos partes que conforman la ley de control, derivativa y proporcional.

Los casos que son presentados realizaron el apuntamiento en una posición determinada y fija para las tres pruebas experimentales, de tal forma que para dicha posición se observará la evolución y mejora del comportamiento esperado, y este comportamiento se extrapola independientemente de la posición fijada para el apuntamiento. Como se ha mencionado se exponen los resultados de una muestra de tres pruebas, empero, en laboratorio se realizaron un conjunto mayor de ellas, de las cuales se reportan tres casos significativos que muestran la evolución para obtener el comportamiento esperado.

7.4.1 Prueba significativa I

La primera prueba significativa, y las sucesivas, muestran el comportamiento gráfico obtenido de las pruebas experimentales del sistema de simulación satelital para la validación de esquemas de control, sus diferentes algoritmos, como TRIAD y EKF, y datos de los sensores giróscopo, magnetómetro y acelerómetro; esto, de un muestreo realizado durante 25 segundos. La primera gráfica (Figura 7.9) muestra las componentes del cuaternión obtenido por los algoritmos TRIAD (azul) y EKF (rojo), a partir del comportamiento fijado para estas pruebas, empero, el comportamiento no tiene relación ni consecuencia con lo que se esperaba pues la curva muestra un desempeño por debajo del cero y posteriormente un comportamiento semiamortiguado, en esta prueba se fijó con los valores: -141.1, -207.9, 307.8. Visualmente se pudo observar que la plataforma MSA se comportó con poca acción ante la perturbación inducida en el eje Z, y posteriormente al intentar suavizar el movimiento, los actuadores indujeron un ciclo de trabajo mayor, pero el cual se especificó en valores entre 70% y 80% de ciclo de trabajo para evitar la saturación de los motores, haciendo que la MSA oscilará sin alcanzar el apuntamiento que se esperaba.

También, se observa cierta coincidencia entre las curvas del cuaternión EKF y cuaternión TRIAD, resultado no esperado, pues la curva de EKF debe ser más *suave* con respecto a la del TRIAD. Por lo tanto, se requiere un ajuste en los valores de las covarianzas de errores asociadas al proceso del algoritmo EKF, tomando por criterio que sus valores sean próximos en magnitud, en este caso, no menores a 2 y no mayores a 4, mientras que en el caso de las ganancias, se procura que sean muy pequeñas, no menores que cero, y no mayores y que 1.

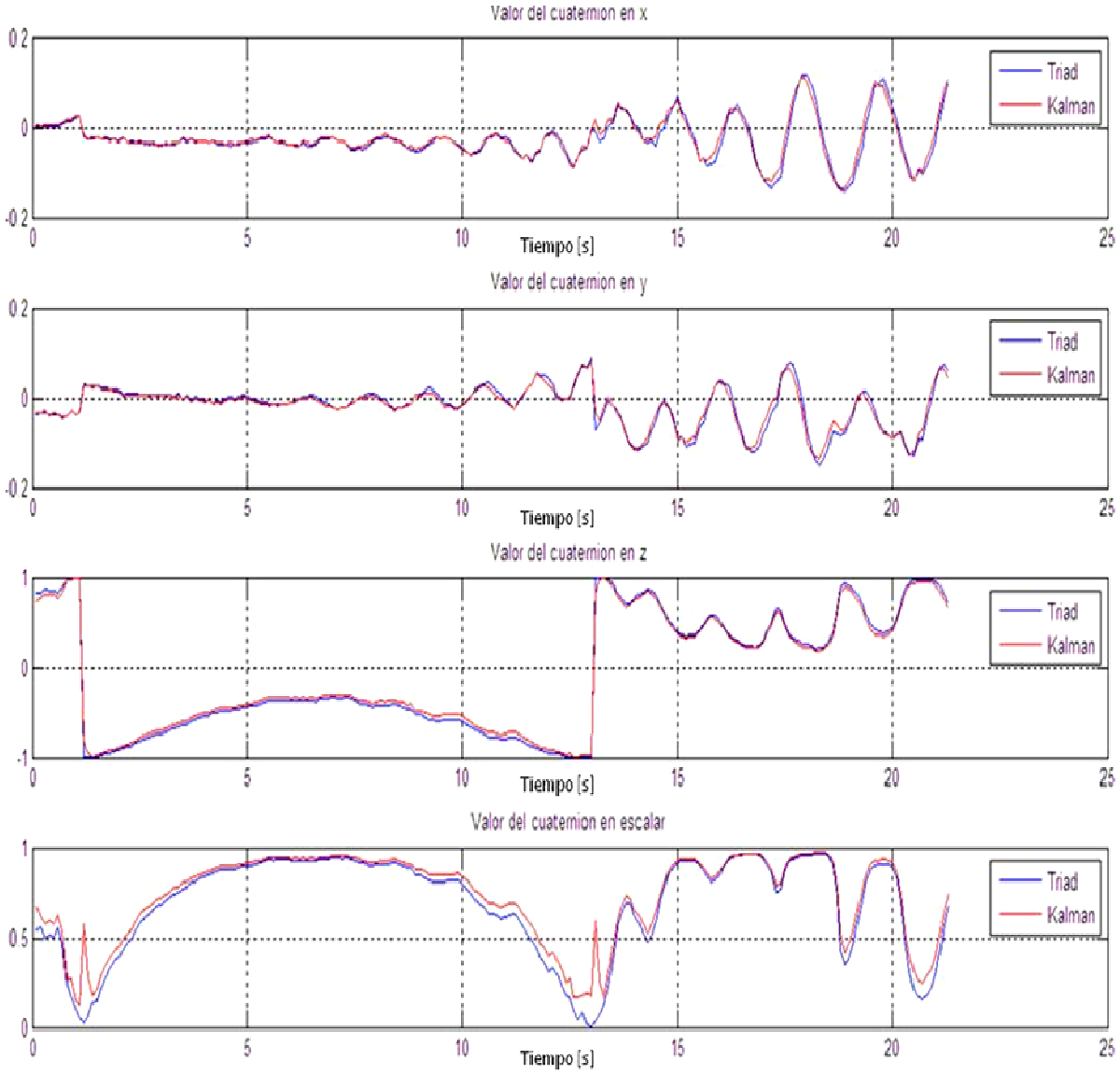


Figura 7.9 Comportamiento de las curva del algoritmo EKF en los ejes X,Y y Z contra las del algoritmo TRIAD

En el caso de los valores de los sensores (Figura 7.10) en el intervalo de 5 a 10 segundos los valores no convergen, incluso el valor para la componente en Z se mantiene sin muchas variaciones, siendo que la variación más grande se esperaba en Z, y posteriormente, debía describirse un comportamiento divergente, es decir, los valores de las componentes *X*, *Y*, *Z*, no deben entrecruzarse, pues el comportamiento esperado es que se mantengan separados dichos valores, lo cual se traduce en un movimiento uniforme. Adicionalmente se muestra en la Figura 7.10 una gráfica de la tasa de errores de recepción y transmisión, por diversos factores en las comunicaciones, y que no afectan significativamente los resultados; entre aquellos factores se identificaron: pérdida de datos en el *handshaking* de las comunicaciones inalámbricas, ruido y caída de voltaje en las señales desde los sensores.

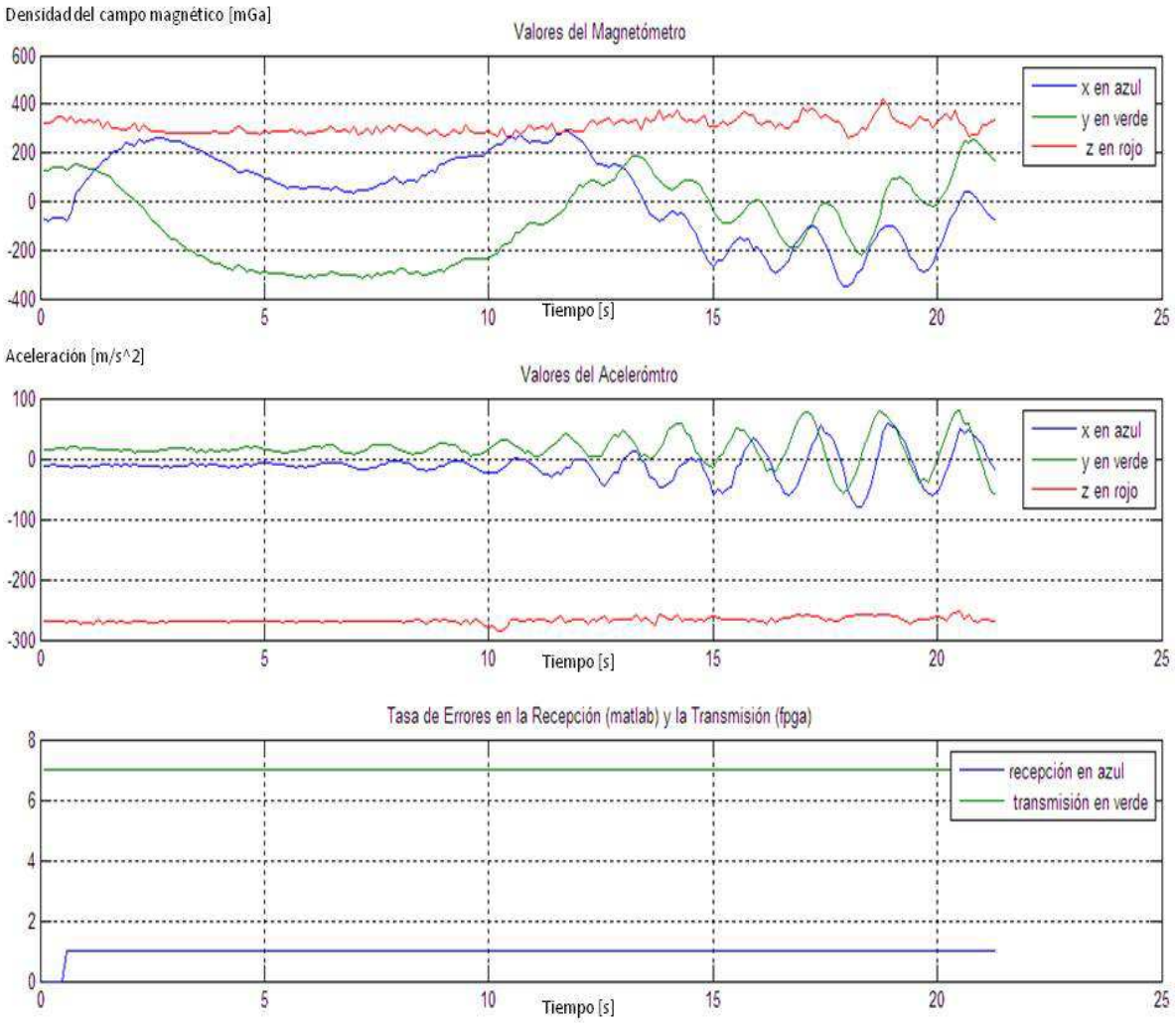


Figura 7.10 Gráfica de comportamiento del magnetómetro y acelerómetro, así como la tasa de errores en las comunicaciones.

La velocidad angular del sistema (Figura 7.11), por su parte, muestra un valor máximo en Z de 0.5 rad/s durante la prueba que corresponde a la perturbación inducida en ese eje, describiendo un comportamiento ascendente y con una caída en el valor mencionado. Es una situación similar como sucede con la gráfica de los valores del cuaternión en EKF, se describen oscilaciones en los ejes que no se esperaba. Debe tomarse en cuenta que la gráfica de la velocidad angular del sistema se espera que sea una curva más suave que la obtenida de los valores del giróscopo, y para ellos deben ajustarse las covarianzas de error en el algoritmo EKF, con base en los criterios mencionados anteriormente.

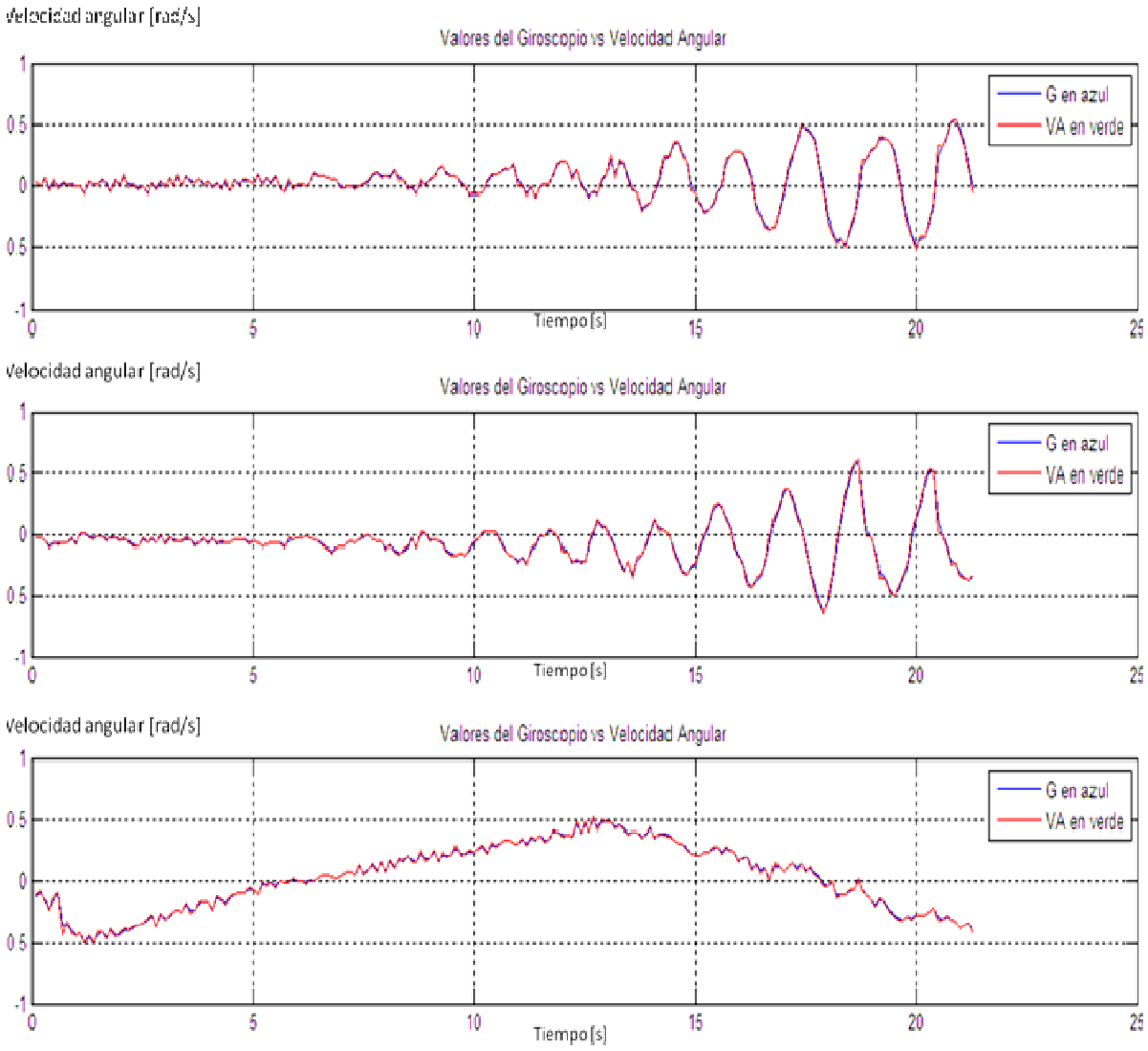


Figura 7.11 Gráfica de comportamiento de los valores del giróscopo contra la velocidad angular.

Finalmente, el valor del torque en *Z* (Figura 7.12) sí muestra un valor máximo, entre 0.5 y 1, durante la parte inicial de la prueba, y va decreciendo conforme avanza el tiempo, aunque los valores en los ejes *Y* y *Z* se mantienen oscilatorios. No obstante, el torque puede ser adecuado, pero no es coherente con el comportamiento en las demás gráficas, lo que equivale a decir que los valores de las ganancias no son lo suficientemente grandes, pues para esta prueba se tienen los valores siguientes: 0.3, 0.2, 0.51 y 0.11, 0.13, 0.33 en la diagonales principales, de las matrices de ganancias para cada una de las partes de la ley de control, proporciona y derivativa, respectivamente.

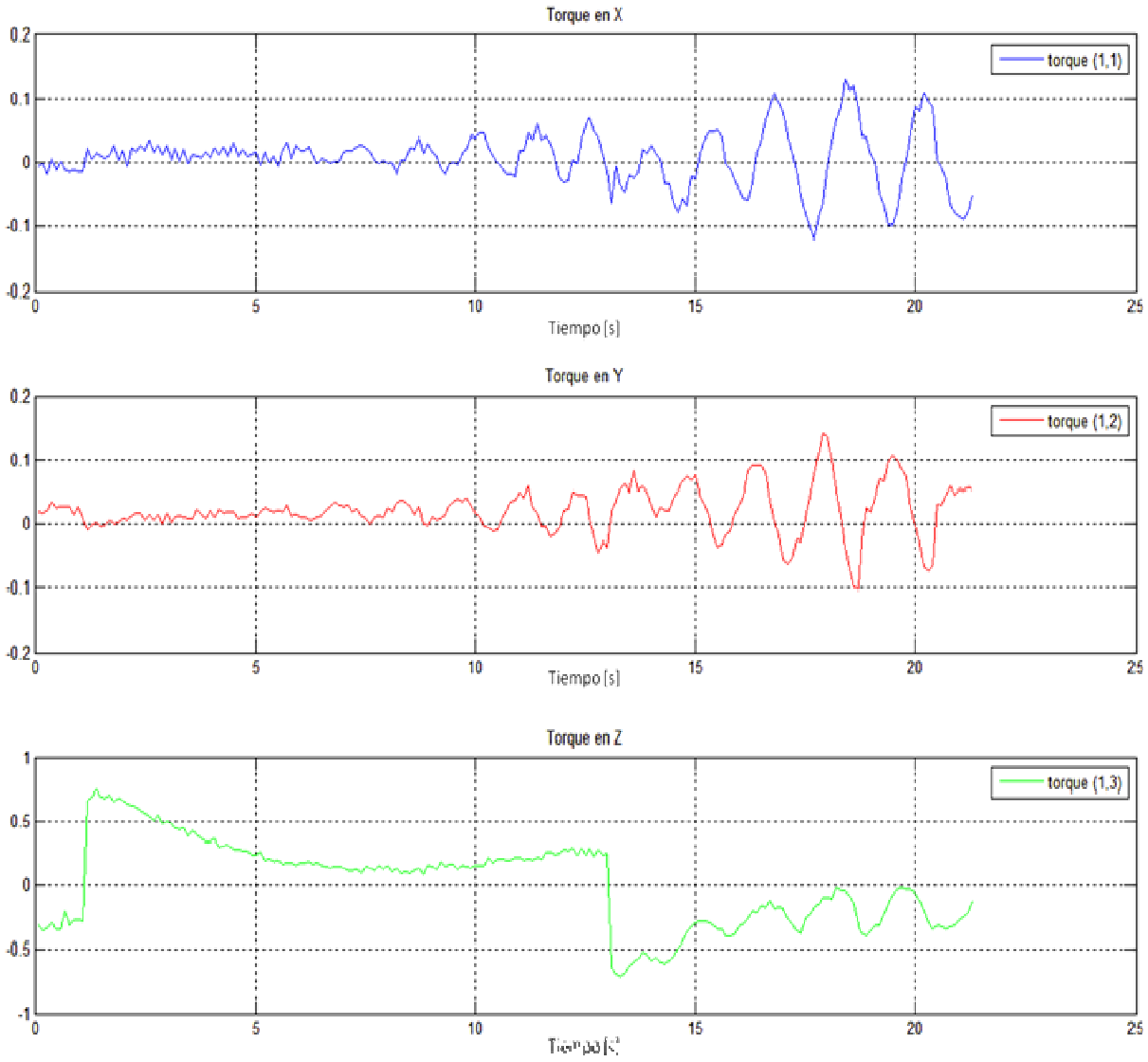


Figura 7.12 Gráfica de comportamiento de los valores torque generado en los ejes de la MSA.

Se concluye entonces, que los valores de las matrices de ganancias no son los adecuados, por un lado, para el caso del control derivativo, no son suficientemente grandes para producir un valor que consiga frenar el movimiento de la MSA en el eje *Z*. Por otro lado, en el control proporcional, son valores que exceden lo requerido, aunque, en el eje Z parece ir disminuyendo la oscilación a partir de un valor máximo, la tendencia en los ejes *X* y *Y* van aumentando, lo cual no debe suceder en una situación como la planteada.

De igual forma, debieron corregirse los valores de la covarianzas de error, y como se dijo, los de las ganancias, considerando los criterios mencionados para mejorar el desempeño y respuesta del sistema.

7.4.2 Prueba significativa II

En la prueba significativa II se ajustaron los elementos de acuerdo a la conclusión de la primera prueba, con los siguientes valores para las ganancias de la ley de control: 0.3, 0.2, 0.43 y 0.21, 0.23, 0.4. Siendo este un trabajo mayormente experimental, los ajustes que se necesitaban para mejorar el desempeño y el comportamiento gráfico se obtuvieron con base en criterios experimentales y buscando conseguir la descripción de movimiento amortiguado del movimiento del sistema y reflejado en las gráficas del cuaternión de los algoritmos TRIAD y EKF, mayor divergencia en los valores del magnetómetro y acelerómetro y mejor corrección en la gráfica de la medición de la velocidad angular de la MSA.

En la Figura 7.13 puede verse el cambio respecto a la primera prueba significativa, los valores obtenidos para los ejes *Y* y *Z* son muy pequeños y no presentan mayor oscilación, puede verse un cambio muy grande al inicio en el eje *Z*, que representa la perturbación del sistema. Este comportamiento que se aproxima mucho a lo esperado aún presenta detalles, pues en algunos intervalos su valor es muy grande, llegando a 1 en magnitud, y se ven picos que no han sido corregidos en la parte de las covarianzas.

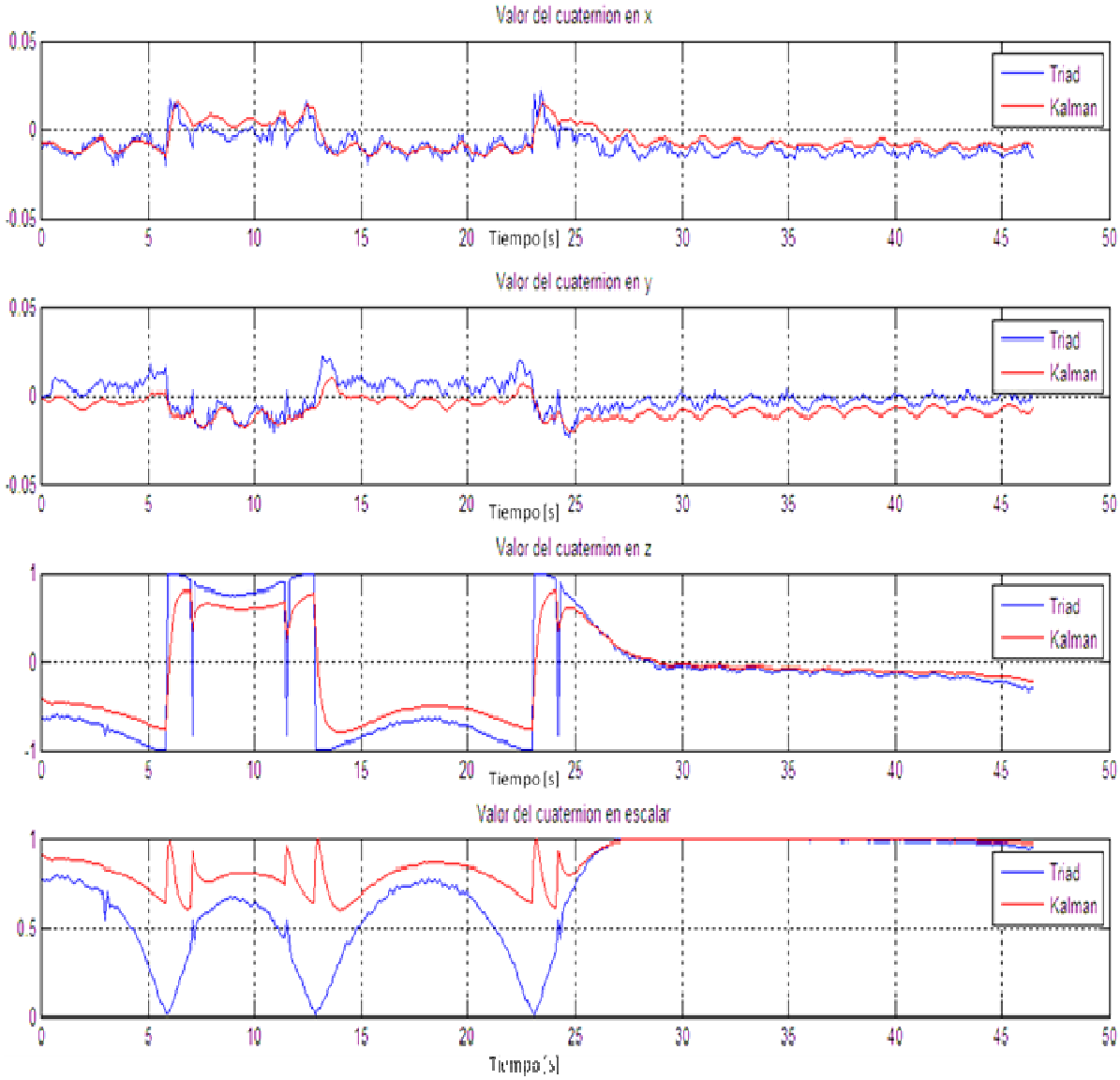


Figura 7.13 Comportamiento de las curva del algoritmo EKF en los ejes X,Y y Z contra las del algoritmo TRIAD

En la gráfica de la Figura 7.14 se aprecia una mayor divergencia en las componentes del magnetómetro, durante los intervalos de 0 a 5 y de 15 a 20, y de 25 hasta el final de la prueba, aunque con picos en las gráficas. Por otra parte, en el caso del acelerómetro no hay un comportamiento muy descriptivo, aunque se ve uniforme durante toda la prueba, y es el comportamiento esperado en este caso, sin oscilaciones en los ejes *X* y *Y* y con ligeras variaciones en el eje *Z*. La tasa de errores, por otra parte, no muestra mayor crecimiento y no representan pérdidas de datos significativas.

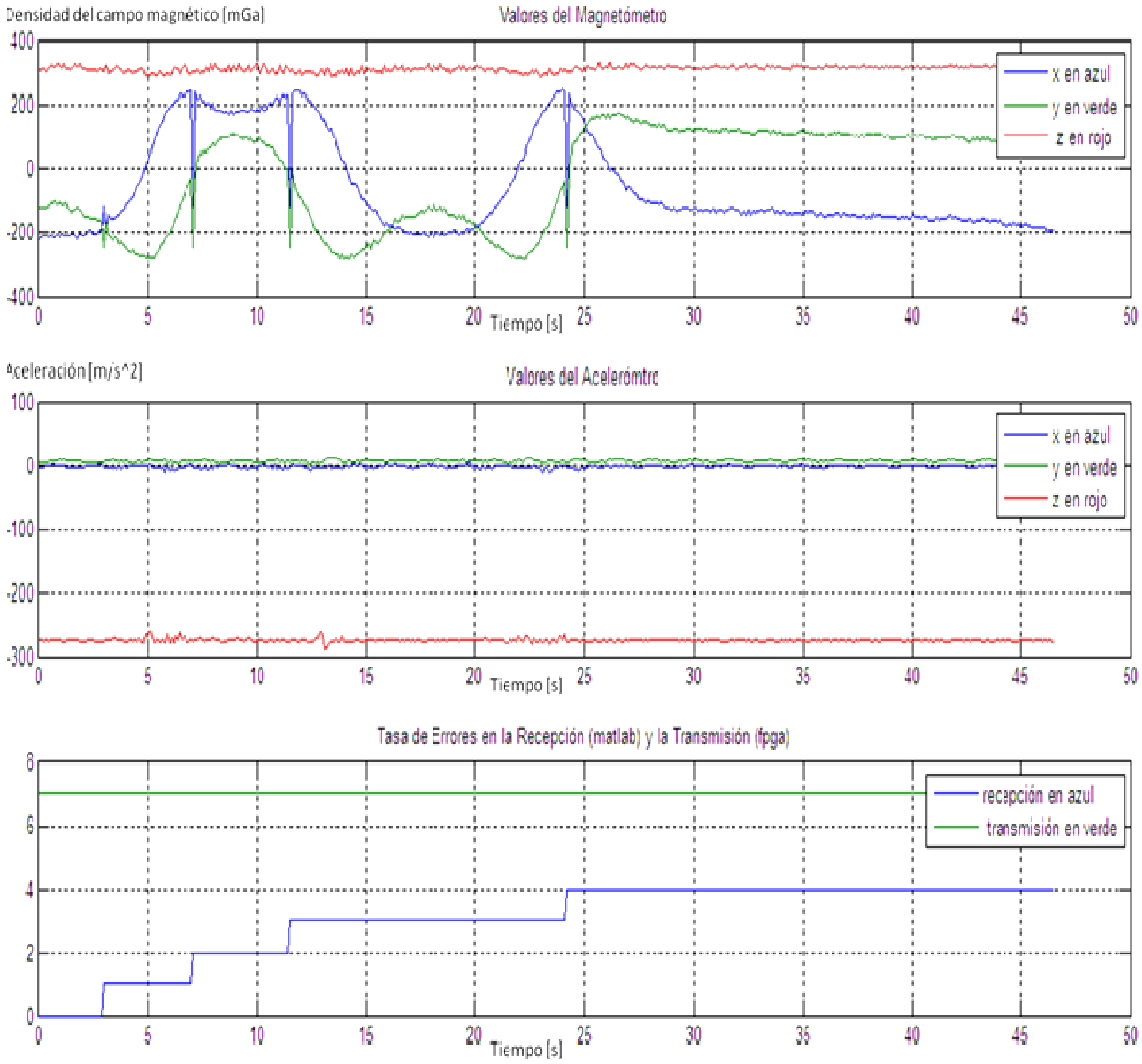


Figura 7.14 Gráfica de comportamiento del magnetómetro y acelerómetro, así como la tasa de errores en las comunicaciones.

En el caso de los torques generados, también se obtuvo un mejor desempeño y comportamiento gráfico, claramente puede verse en el eje Z la progresiva disminución de la magnitud del torque, siendo 0.2 el valor máximo en el intervalo de 10 a 15, y visualmente esa magnitud es suficiente para verse reflejada en la acción de los actuadores, quienes tienen la velocidad suficiente sin llegar a saturase, mientras en los ejes X y Y puede apreciarse que se mantienen en un intervalo de valores muy pequeños en el orden hasta de 10^{-3}, mismos que deben corregirse un poco más en la medida de lo posible de acuerdo con la experimentación.

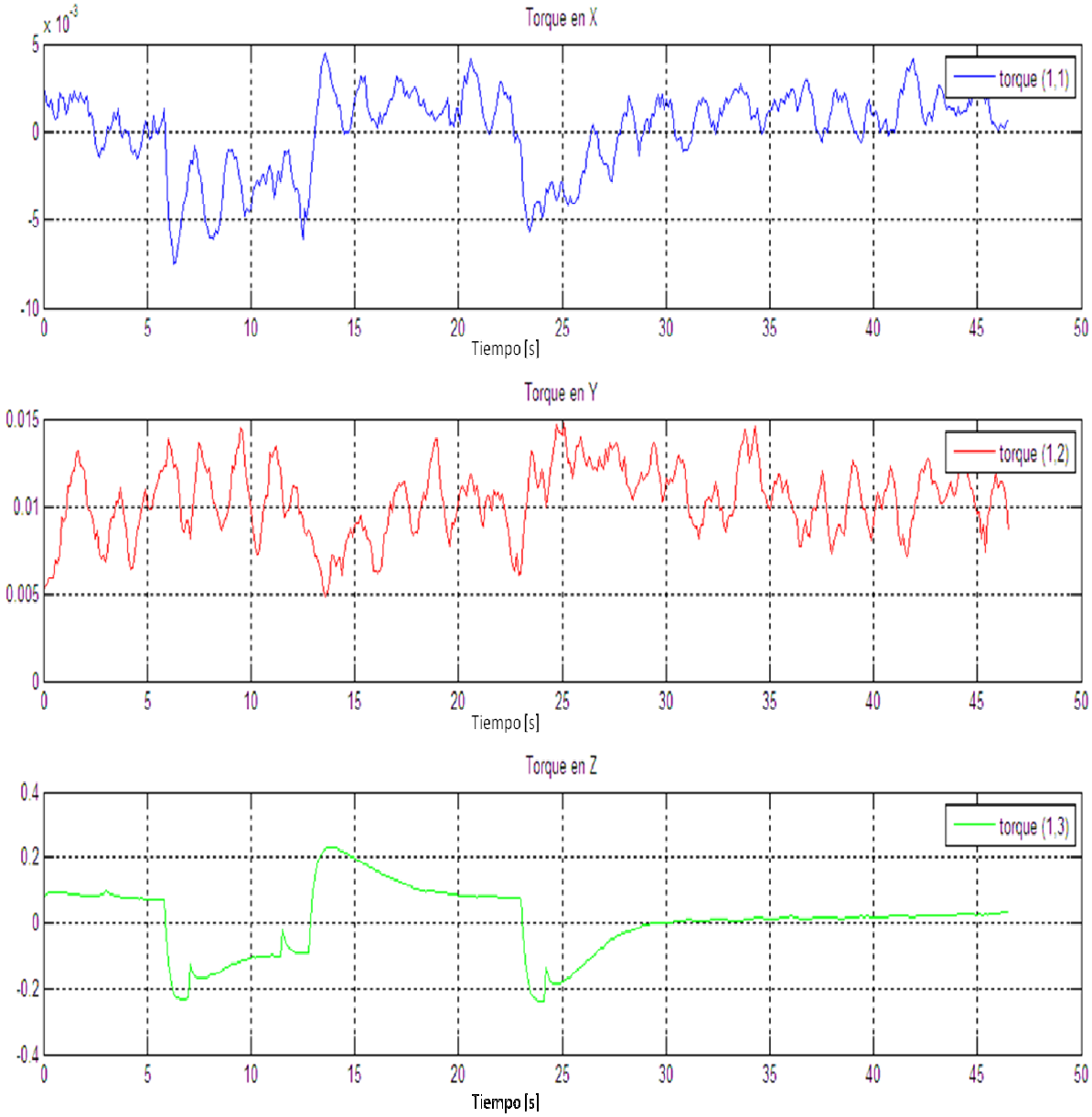


Figura 7.15 Gráfica de comportamiento de los valores torque generado en los ejes de la MSA.

En cuanto a la gráfica de la velocidad angular, en esta prueba se ajustaron los valores de las covarianzas: la asociada al giróscopo de 0.2; asociadas a los procesos de corrección y determinación del cuaternión de 0.3, 0.21 y 0.3. Por lo que el comportamiento de los valores del sensor giróscopo han sido corregidos durante la obtención de la velocidad angular del sistema y puede verse el cambio significativo, además de que el comportamiento indica que la velocidad, efectivamente va reduciéndose hasta que el sistema logra mantenerse en la posición de apuntamiento.

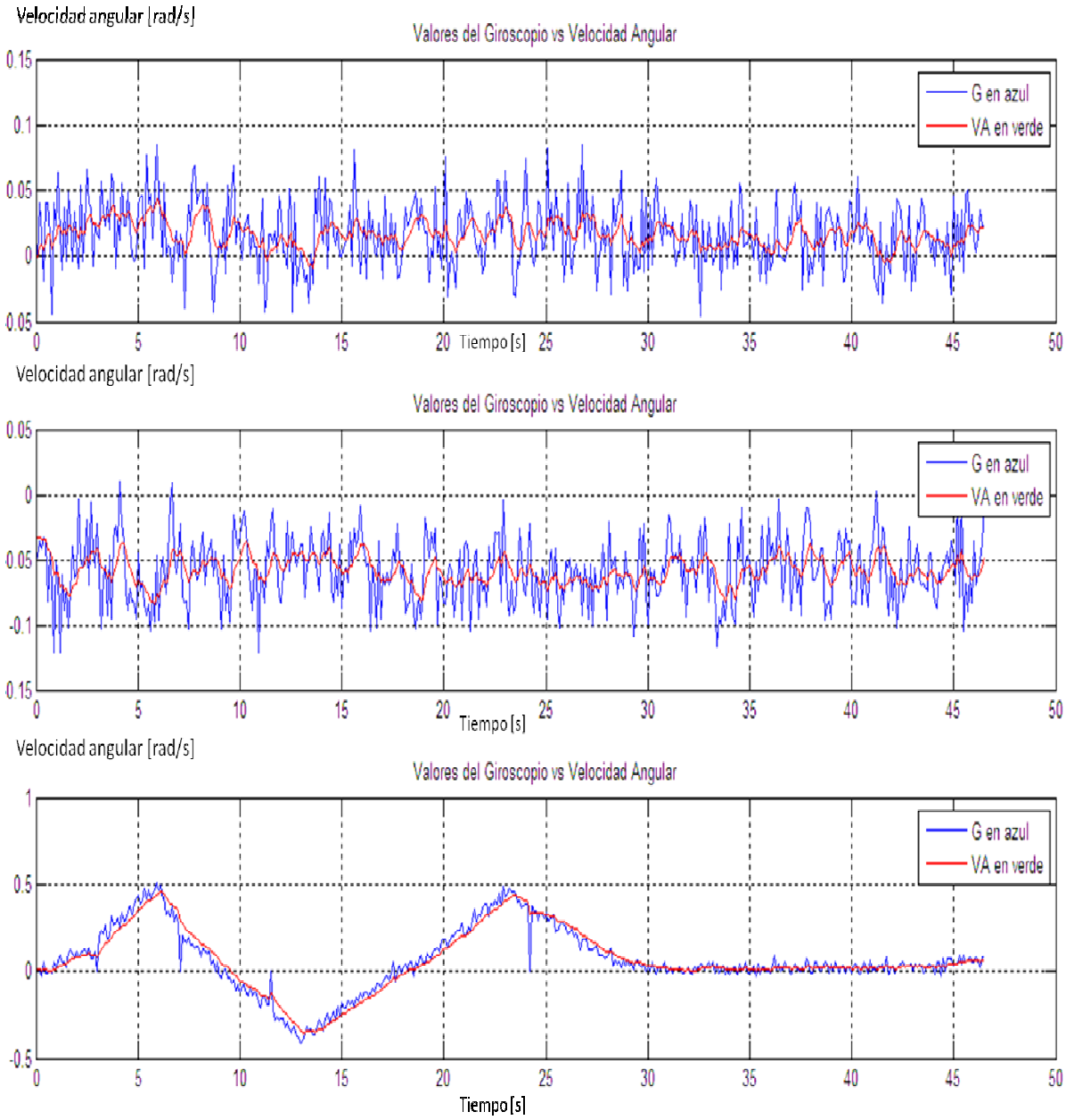


Figura 7.16 Gráfica de comportamiento de los valores del giróscopo contra la velocidad angular.

Realizadas las pruebas para lograr que el sistema se comporte y realice las funciones principales de estabilización y control, se debe mejorar esta aproximación obtenida, con lo cual podrá concluirse el trabajo realizado y el alcance de esta primera aproximación de la integración del sistema de simulación para la validación de esquemas de control.

7.4.3 Prueba significativa III

La prueba significativa III es la que se obtuvo con el mejor rendimiento y el comportamiento más adecuado de todas las pruebas realizadas del sistema de simulación para la validación de esquemas de control, donde pudo validarse el esquema de control propuesto de apuntamiento y estabilización, dada una perturbación en la MSA o realizara esta tarea desde en un posición inicial en reposo, pues en ambos casos realizó el apuntamiento que se indicó con base en el sistema de referencia utilizado en el algoritmo TRIAD, con valores -138, -215 y 303, en función de las mediciones del magnetómetro.

Esta tesis finaliza con los resultados obtenidos de las pruebas realizadas en laboratorio, más no así el trabajo con el sistema, y con base en esta primera aproximación obtenida se harán una serie de conclusiones y recomendaciones para el trabajo futuro.

Para realizar esta prueba se ajustaron por última vez los valores de las ganancias, quedando del siguiente modo: 0.28, 0.25, 0.336 y 0.15, 0.1, 0.3, en la parte proporcional y derivativa, respectivamente, puede observarse que la componente proporcional y derivativa correspondiente al eje Z, son mayores que las correspondientes a las componentes de los ejes *X* y *Y*.

En las gráficas de la Figura 7.17 se puede observar el comportamiento deseado de la MSA, el amortiguamiento inicial, y posteriormente la disminución de movimiento hasta el frenado y apuntamiento que debe realizar el sistema.

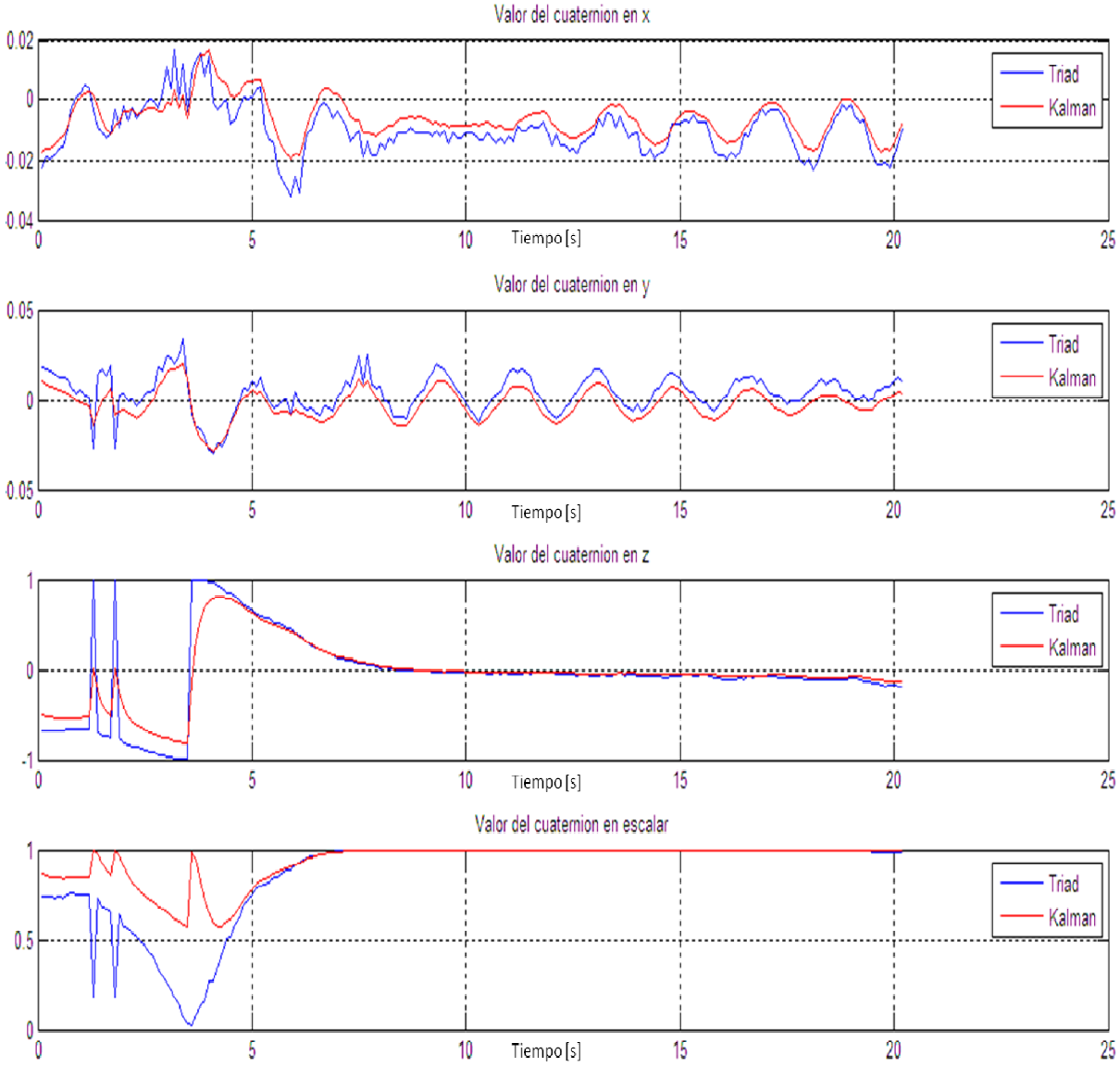


Figura 7.17 Comportamiento de las curva del algoritmo EKF en los ejes X,Y y Z contra las del algoritmo TRIAD

En la Figura 7.18 se muestran los valores del magnetómetro y acelerómetro, donde se aprecia claramente la divergencia de los valores del magnetómetro a partir de los 5 segundos en adelante, así como la divergencia inicial hasta el final del acelerómetro, y una menor tasa de errores en las transmisiones. Hasta el momento se han cubierto los resultados esperados en esta aproximación. Cabe mencionar que para obtener estas pruebas sólo se ajustaron los valores de las ganancias, mientras que los valores para las convergencias ya no fueron modificados, quedando los mismos valores que en la prueba anterior.

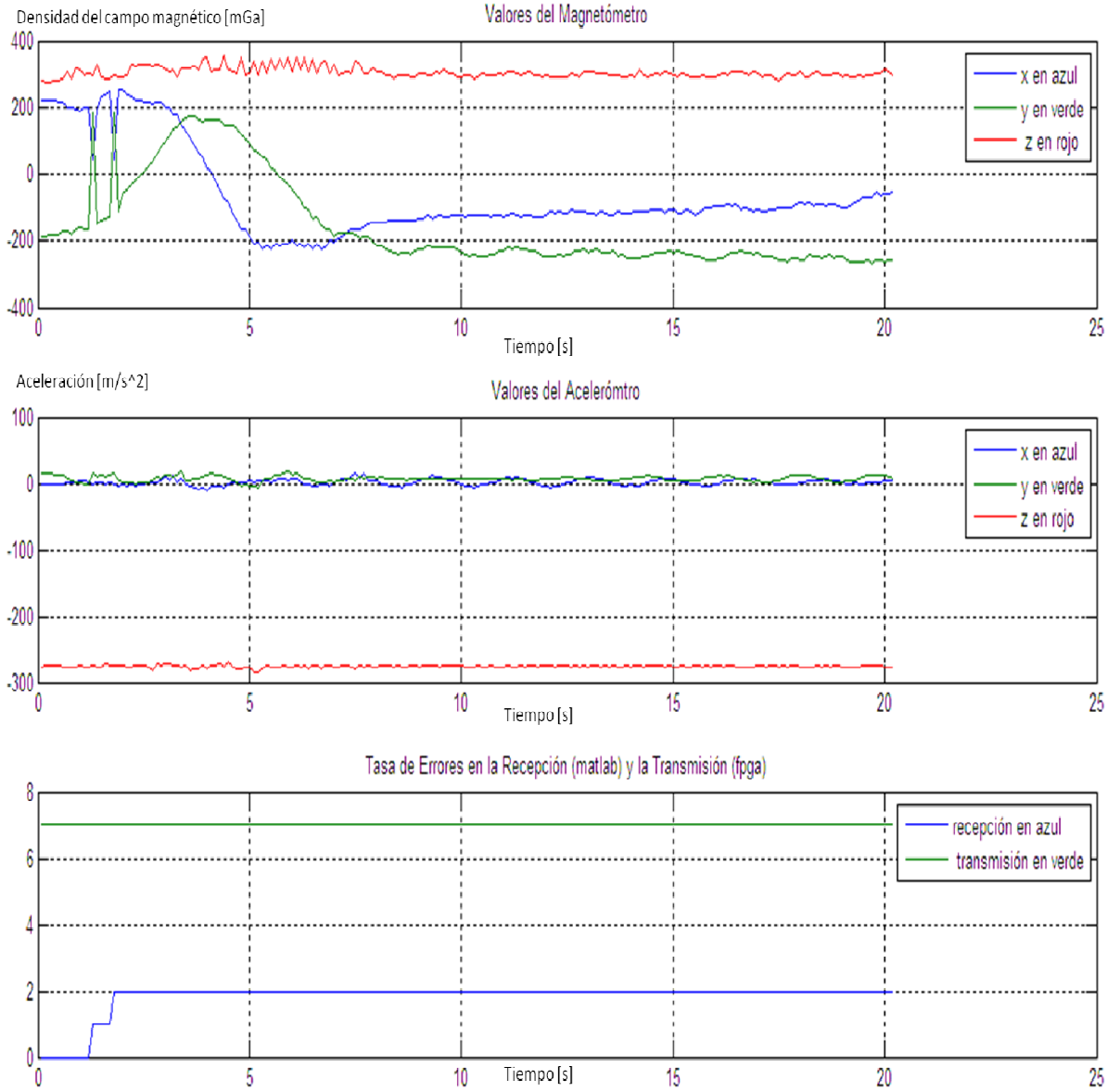


Figura 7.18 Gráfica de comportamiento del magnetómetro y acelerómetro, así como la tasa de errores en las comunicaciones.

En la siguiente gráfica también se ha verificado el funcionamiento del algoritmo EKF, obteniendo una curva más suave para la velocidad angular de la MSA, y se comprueba al compararla con la gráfica que se describe con los valores obtenidos por el giróscopo, todos los picos que se obtiene en la gráfica azul son corregidos en la gráfica roja, y se ve que el mayor valor obtenido en el eje Z es de 0.6 rad/s y posteriormente disminuye hasta que el sistema alcanza su apuntamiento.

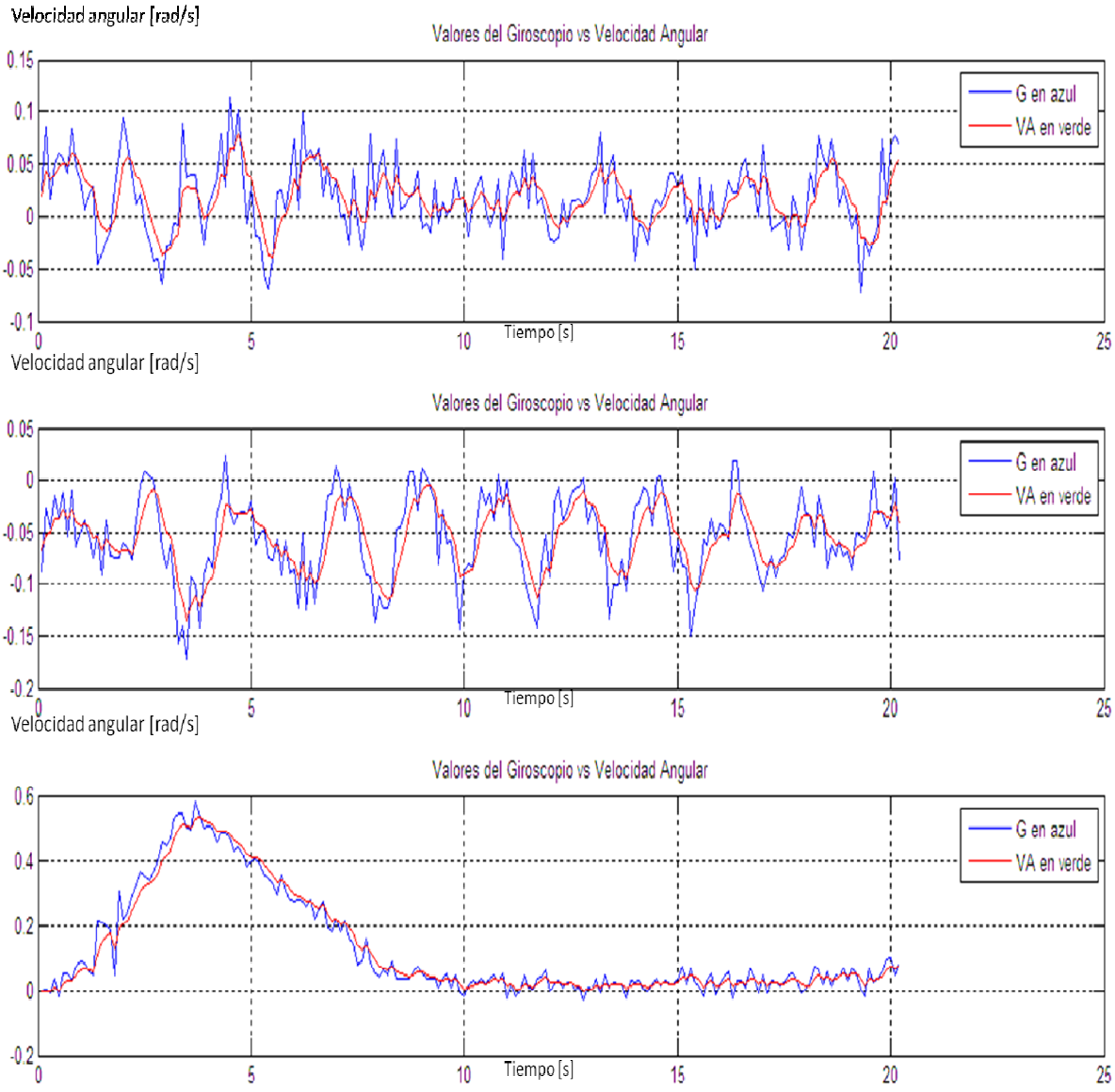


Figura 7.19 Gráfica de comportamiento de los valores del giróscopo contra la velocidad angular.

Finalmente se obtuvo la gráfica de los torques en cada uno de los ejes del sistema, los cuales muestran los valores mayores en el eje *Z*, donde se indujo una perturbación, y llega el valor máximo de 0.2, permitiendo un buen desempeño de las ruedas sin que saturen, y ese valor muestra una tendencia uniforme de disminución hasta que el sistema llega al apuntamiento y estabilización de la MSA, mientras en los ejes *X* y *Y* pueden corroborarse que los valores graficados son muy pequeños en comparación con los valores en Z.

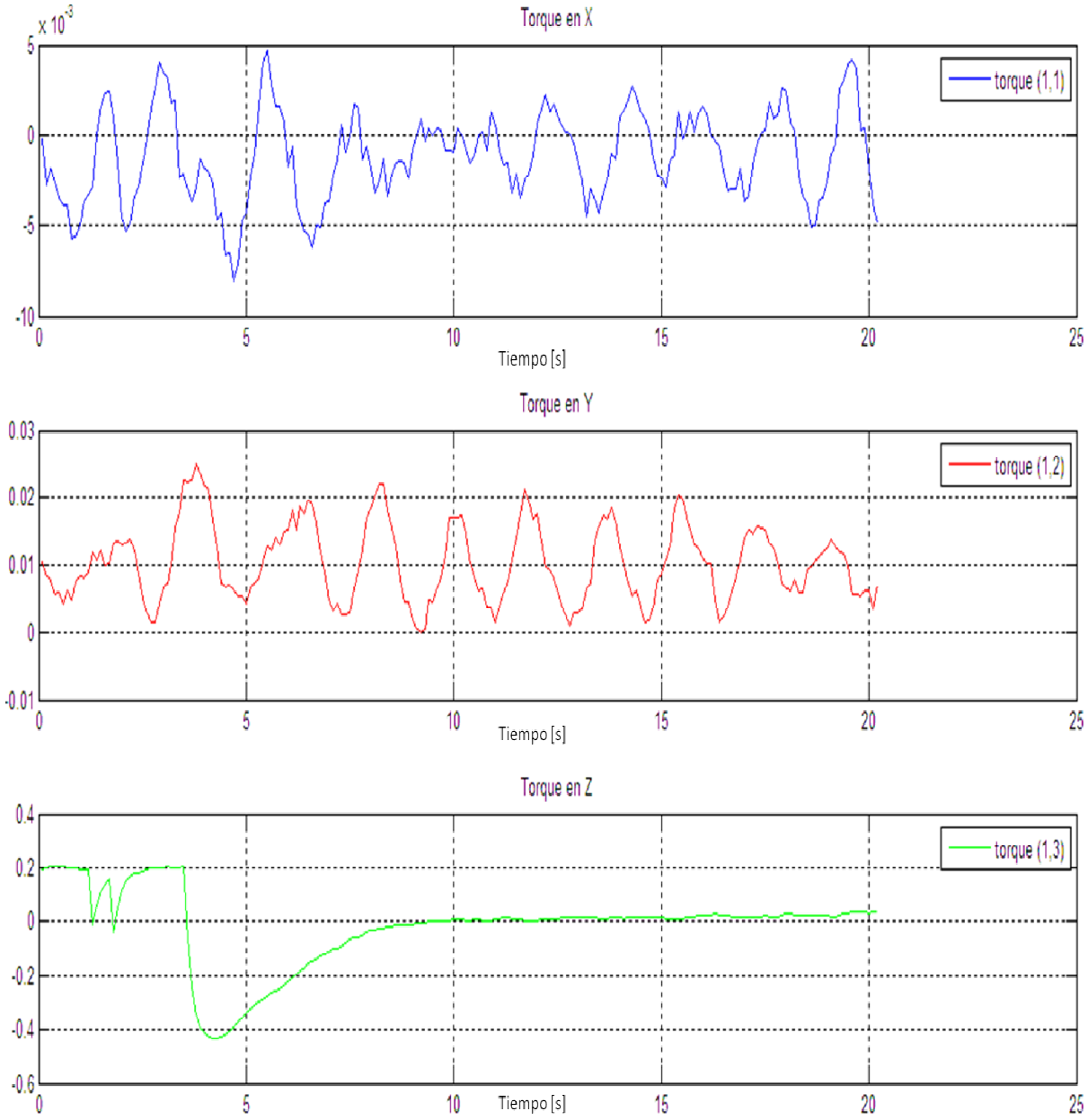


Figura 7.20 Gráfica de comportamiento de los valores torque generado en los ejes de la MSA

Como se ha dicho, los resultados de esta aproximación fueron los mejores entre las pruebas realizadas, con base en éstas se ha validado el esquema de control de la orientación y la estabilización de la MSA, y con ello se deben proponer consideraciones y recomendaciones para la continuación del trabajo.

Entre esas recomendaciones está el mejoramiento de hardware, en cuanto a los recursos de procesamiento, la implementación de toda la parte lógica en un solo dispositivo de procesamiento a bordo, así como el mejoramiento de los actuadores, reevaluando con base en estos resultados la opción de fabricar actuadores con mejores características que respondan a las maniobras que requiere un sistema como la MSA y siempre teniendo en cuenta que es un sistema que será trasladado a un satélite real, y como tal, deberá validarse el funcionamiento en su totalidad y que sólo requiera ligeros cambios para trasladarse al satélite.

El siguiente apartado se enfoca en las conclusiones del trabajo desarrollado, así como las consideraciones que se deben tener en cuenta y que están fundamentadas en la experiencia en el desarrollo así como en los resultados obtenidos.

Capítulo 8

Conclusiones y recomendaciones

8.1 Conclusiones

En la propuesta de esta tesis se plantearon e indicaron objetivos, donde el principal fue diseñar, implementar y validar operativamente un sistema de control de orientación en tres ejes sobre un simulador para pruebas de control para satélites pequeños que llamamos Mesa Suspendida en Aire (MSA), la cual es una plataforma de pruebas basada en un cojinete de aire tipo mesa. Este sistema es un sistema de validación en tierra de un ADCS de un satélite, donde éste realiza las funciones de determinación de la orientación y el control del vehículo en el espacio y debe cumplir ciertos requisitos que lo caracterizan en función de la misión del satélite; por su parte, la MSA tiene como finalidad ser una plataforma abierta y didáctica, que permita validar diversos sistemas de control sobre una plataforma instrumentada con componentes genéricos y simular condiciones de no fricción.

En este capítulo se exponen las conclusiones sobre la integración que se hizo del sistema, y también se darán recomendaciones para el trabajo futuro teniendo en cuenta la experiencia obtenida en el desarrollo de esta primera aproximación del sistema de simulación satelital para la validación de esquemas de control.

Por lo tanto:

- Se integró una plataforma de simulación satelital para validar la lógica y hardware del ADCS (sensores, actuadores, computadora de vuelo), software (algoritmos de procesamiento) y la instrumentación necesaria para realizar las maniobras de control, así como demás accesorios necesarios para el funcionamiento del sistema.

- Se utilizó la técnica *hardware in the loop* (HIL) para la aceleración del desarrollo del sistema de simulación satelital, y con ello se realizaron pruebas de validación de cada uno de los bloques que integran el sistema: por un lado el segmento implementado en la plataforma FPGA, que se encarga de la adquisición de datos del magnetómetro, acelerómetro y giróscopo, así como de recibir los comandos de control y escribirlos en los registros del núcleo PWM para las maniobras de control; y por otro, el segmento implementado en MATLAB que ejecuta sobre una PC externa a la plataforma, los algoritmos de procesamiento TRIAD, EKF y CONTROL, gestionados por un programa principal.

- Se resolvieron los problemas de comunicación inalámbrica entre los dos segmentos que componen el sistema de simulación satelital, mediante la incorporación de radios módem, ajustando la velocidad y formato de envío entre los diversos puertos de las plataformas FPGA y PC, teniendo una velocidad de transmisión de 9600 bps.

- Se realizaron pruebas parciales para la validación de la instrumentación de la MSA, realizando la configuración de los sensores para la lectura de datos, así como la activación de un filtro integrado pasa bajas para la reducción del ruido en la señal del sensor giróscopo, y también se validó el desempeño de la ruedas inerciales logrando calcular la velocidad angular de cada una de ellas en función del ciclo de trabajo inducido, así como el momento angular generado de acuerdo a esa velocidad.
- El modelo dinámico de la MSA se retomó de un trabajo previo con el fin de obtener la ecuación de movimiento del sistema, que en el caso de un ADCS es una dinámica de la orientación, donde se relaciona la velocidad angular del sistema con los momentos de las fuerzas que actúan sobre él. Esta relación viene dada por la velocidad angular mencionada y los tensores de inercia que se refieren a distribución de masa en los ejes principales del sistema, que como se expuso en el Capítulo 3, el centro de masa del sistema es coincidente con el centro del sistema de referencia, esto nos permitió estudiar el cuerpo en su movimiento rotacional y simplificar el análisis. No obstante, como se ha dicho, desarrollar completamente la ecuación de movimiento permitirá mejorar el esquema de control y estabilización, así como permitirá tomar el análisis y realizar una simulación virtual del mismo, ampliando el espectro de aplicaciones y alcances del sistema en la investigación.

- Mediante el uso de herramientas de software de Xilinx, se desarrollaron núcleos personalizados y se hizo uso de núcleos propietarios para realizar las funciones siguientes: núcleos personalizados PWM para la interpretación de comandos de ciclo de trabajo; uso de núcleos propietarios para el control y ejecución de interrupciones para la lectura de datos de los sensores magnetómetro, acelerómetro y giróscopo, y uso del núcleo I2C como protocolo de comunicación con estos sensores.

- Con base en las herramientas de software Xilinx se generó un arquitectura de cómputo embebida haciendo uso del microprocesador Microblaze, el cual se encarga de gestionar los núcleos periféricos y además incluye rutinas en lenguaje de alto nivel para la configuración de la IMU, adquisición de datos de los sensores de navegación, envío de datos de los sensores con el formato adecuado hacía la PC externa y recepción de los comandos de control desde la PC externa.

- Se implementaron en lenguaje de programación MATLAB los algoritmos siguientes: TRIAD para la determinación de la orientación del simulador satelital; EKF para la corrección y estimación de la orientación del simulador satelital, así como para la medición de la velocidad angular del sistema; y la ley de control que determina los ciclos de trabajo y el sentido de giro para cada una de las ruedas inerciales.

- Se realizaron análisis y discusiones sobre las mejores estrategias de trabajo para realizar la implementación del sistema, así como reducir el tiempo de desarrollo del mismo, con lo que justificó y se argumentó el uso de las técnicas HIL, así como las herramientas con las que se contaba para integrar el sistema, sentando las bases para la continuidad del trabajo sobre la misma línea.

- Se realizaron pruebas experimentales suficientes con las que se validó, verificó y ajustó el funcionamiento del sistema, teniendo como primera aproximación estabilización en los tres ejes ante una perturbación del mismo, y aunque las maniobras de apuntamiento aún no pueden lograr la inclinación en los tres ejes debido a las limitaciones en el hardware e instrumentación, como el tamaño de las ruedas, no se cuenta con bobinas de par magnético, entre otras, se obtuvieron resultados positivos. El comportamiento gráfico que se obtuvo de las diversas pruebas experimentales muestra una curva amortiguada y la disminución del movimiento y velocidad angular del sistema hasta su estabilización, y posteriormente el apuntamiento dentro de una ventana en la cual permanece el sistema teniendo un margen de error entre 10 y 15 grados.

8.2 Recomendaciones y trabajo futuro

Las recomendaciones en este trabajo están enfocadas a la continuidad del mismo con el objetivo de mejorar y robustecer el sistema, no sólo en el esquema de control y estabilización, sino también en cuanto a los componentes físicos y lógicos, de tal forma, que se retomé la estrategia de trabajo y se tengan en cuenta los puntos críticos que se expondrán, y que han sido obtenidos a partir del análisis y la experiencia a lo largo de la integración de este sistema.

- La integración de la arquitectura de cómputo en un solo dispositivo FPGA a bordo de la MSA, realizando la transferencia del bloque de coprocesamiento desarrollado en MATLAB hacia la plataforma de desarrollo.

- Considerando lo anterior, para llevara a cabo la transferencia se debe analizar qué bloques o funciones se realizarán en hardware (núcleos personalizados o propietarios, escritos en lenguaje VHDL) o software (programación secuencial en lenguaje de alto nivel como C o C++, o de bajo nivel como ensamblador).

- Se deberá tener en cuenta el número de operaciones que realizan cada uno de los algoritmos (TRIAD, EKF, CONTROL), y con base en ese análisis se puede determinar las funciones o bloques que serán implementados en hardware para aprovechar las características de reutilización de código.

- Se debe completar el desarrollo del análisis dinámico de la MSA para mejorar el esquema de control y estabilización con el fin de simular virtualmente el funcionamiento de la plataforma, así como incorporar más elementos en la instrumentación de la MSA y como otro medio de validación. Se debe priorizar el diseño de nuevas estrategias de control, lo cual implica selección de instrumentación y mejora de los algoritmos para ver el sistema como es, no lineal; también realimentar velocidades angulares.

- La incorporación de nuevos componentes y mejoramiento de la instrumentación que hay debe considerarse como un aspecto fundamental para el mejoramiento del funcionamiento, que en tal caso, se debe considerar la incorporación de sensores con mejor resolución en las muestras de datos, bobinas de torque magnético, ruedas inerciales de mayor diámetro, etc.

- Las nuevas características y funciones adicionales, tales como seguimiento de sol, fase de eclipse, entre otras que pueda realizar la MSA, dependerán del mejoramiento del esquema de control, así como el modelado dinámico del mismo; es por ello la importancia de seguir profundizando y mejorando lo que se ha realizado hasta ahora con la integración de este sistema.

- El uso de FPGA se debe extender en trabajos posteriores explotando sus características de procesamiento concurrente, escalabilidad, versatilidad y reducción del impacto ante la obsolescencia, así como el uso de las técnicas HIL para la validación y verificación del funcionamiento del los diversos componentes del sistema antes de su integración final.

- Se debe mejorar el código desarrollado, ya sea reduciendo las líneas de código para optimizar los procesos, o bien, para agregar nuevos módulos o segmentos de código que realicen nuevas funciones y tareas, dando prioridad al tiempo determinado para su ejecución.

- Se ha de incluir un método numérico o estadístico que permita seleccionar o determinar valores más adecuados y precisos para los parámetros relacionados directamente con el comportamiento y desempeño de este sistema, como las covarianzas de error, las matrices de ganancias, por mencionar algunos; reducir el tiempo en el tratamiento de datos de los sensores de navegación (procesarlos en formato original, el cual es octal, reduciendo el tiempo del proceso); soporte técnico en la plataforma FPGA para los componentes que se utilizaron actualmente (uso de protocolos serie, I2C, bus PLB o de mayor velocidad, soporte para MicroBlaze), etc.

Todo esto puede lograrse manteniendo la línea de trabajo que hasta ahora se ha seguido para la integración del sistema de simulación satelital para la validación de esquemas de control, y aunque todos estos puntos de alguna manera se enfocan en mejoras técnicas, lógicas y matemáticas, también debe haber énfasis en los recursos humanos, en este caso las personas involucradas en el proyecto y que en sus manos están las bases y herramientas para perfeccionar esta primera versión, que busca ser un pilar en la investigación satelital en México en la categoría académica y los satélites pequeños.

Bibliografía:

[1] Meissner, David. *A three degrees of freedom test bed for nanosatellite and cubesat attitude dynamics, determination, and control*. Monterey, California, 2009. pp. 14-15.

[2] Meissner, David. *A three degrees of freedom test bed for nanosatellite and cubesat attitude dynamics, determination, and control*. Monterey, California, 2009. pp. 12.

[3] Meissner, David. *A three degrees of freedom test bed for nanosatellite and cubesat attitude dynamics, determination, and control*. Monterey, California, 2009. pp. 12-13

[4] Schwartz, Jana L. *The distributed spacecraft attitude control system simulator: from design concept to decentralized control.* Blacksburg, Virginia. 2004. pp 42.

[5] Downs, Matthew C. *Adaptive Control Applied to the Cal Poly Spacecraft Attitude Dynamics Simulator.* San Luis Obispo, California, 2009. pp 2-3.

[6] Dae-Min Cho; Dongwon Jung; Panagiotis Tsiotras. A *5-dof experimental platform for autonomous spacecraft rendezvous and docking*. Atlanta, Georgia, pp 2-5.

[7] Walchko, Kevin J.; Nechyba, Michael C.; Schwartz. Eric; Arroyo, Antonio. *Embedded Low Cost Inertial Navigation System*. Gainesville, Florida. 2003. pp 3-4.

[8] Hoja de datos de acelerómetro. *SCA3000-E01 3-Axis Ultra Low Power Accelerometer With Digital Spi Interface*. VTI technologies.

[9] Hoja de datos de magnetómetro. *Integrated Compass Sensor HMC6052.* Honeywell. Plymouth, Minessota. 2006.

[10] Hoja de datos de Microprocesador. *dsPIC33FJ256GP506 16-Bit Digital Signal Controllers.* Microchip Technology Inc., 2009.

[11] Downs, Matthew C. *Adaptive Control Applied to the Cal Poly Spacecraft Attitude Dynamics Simulator.* San Luis Obispo, California, 2009. pp 11.

[12] Hoja de referencia MicroBlaze. *MicroBlaze Processor Reference Guide Embedded Development Kit EDK 10.1i.* Xilinx, Inc. 2008.

[13] Clarke, Peter. *European Space Agency launches free Sparc-like core*. Publicado 3/6/2000. http://www.eetimes.com/electronics-news/4151500/European-Space-Agency-launches-free-Sparc-like-core. [Consulta: 15-10-2012].

[14] Paluszek, Michael, Bhatta, Pradeep; Griesemer, Paul; Mueller, Joseph; Thomas, Stephanie. *Spacecraft attitude and orbit control*. Plainsboro, New Jersey. 2009, 2nd. Edition. Princeton Satellite Systems, Inc. pp. 242.

[15] Paluszek, Michael, Bhatta, Pradeep; Griesemer, Paul; Mueller, Joseph; Thomas, Stephanie. *Spacecraft attitude and orbit control*. Plainsboro, New Jersey. 2009, 2nd. Edition. Princeton Satellite Systems, Inc. pp. 215.

[16] Paluszek, Michael, Bhatta, Pradeep; Griesemer, Paul; Mueller, Joseph; Thomas, Stephanie. *Spacecraft attitude and orbit control*.Plainsboro, New Jersey. 2009, 2nd. Edition. Princeton Satellite Systems, Inc. pp. 215.

[17]. Chu, Pong P. *FPGA prototyping by VHDL examples*. John Wiley & Sons, Inc., 2008. pp. 42.

[18]. Chu, Pong P. *FPGA prototyping by VHDL examples*. John Wiley & Sons, Inc., 2008. pp. 42.

[19]. Chu, Pong P. *FPGA prototyping by VHDL examples*. John Wiley & Sons, Inc., 2008. pp. 42.

[20]. Chu, Pong P. *FPGA prototyping by VHDL examples*. John Wiley & Sons, Inc., 2008. pp. 42.

[21]. Chu, Pong P. *FPGA prototyping by VHDL examples*. John Wiley & Sons, Inc., 2008. pp. 42.

[22]. Xilinx Inc. *The Programmable Logic Data Book*. Xilinx Inc. April, 1998. pp. 189 – 192.

[23]. Xilinx Inc. *The Programmable Logic Data Book*. Xilinx Inc. April, 1998. pp. 197.

[24] *JTAG*. Publicado 9/3/2013. http://es.wikipedia.org/wiki/JTAG. [Consulta: 15-10-2012].

[25]. G.,Oscar. *Estándar de comunicaciones RS-232C*. http://www.euskalnet.net/shizuka/rs232.htm . [Consulta 15-10-2012].

[26] *ISE Design Suite*. http://www.xilinx.com/products/design-tools/ise-design-suite/index.htm . Xilinx, Inc. 2013.

[27] *Qsys - Altera's System Integration Tool.* http://www.altera.com/products/software/quartus-ii/subscription-edition/qsys/qts-qsys.html . Altera, Corporation. 2013.

[28] *Microsemi FPGA and SoC Development Software.* http://www.Microsemi.com/products/software/libero/ . Microsemi, Corporation. 2013.

[29] Hoja de referencia. *Spartan-3E FPGA Starter Kit Board User Guide*. Xilinx, Inc. Enero 20 de2011.

[30] Hoja de referencia. *Cyclone II DSP Development Board Reference Manual*. Altera, Corporation. Agosto de 2006.

[31] Hoja de referencia. *IGLOO nano Starter Kit User's Guide*. Microsemi, Corporation. 2012.

[32] Arenas Gaviria, Bernardo. *Dinámica De Un Cuerpo Rígido (Apuntes del Instituto de Física)*. Antioquia, Colombia. 2012. pp. 2.

[33] Arenas Gaviria, Bernardo. *Dinámica De Un Cuerpo Rígido (Apuntes del Instituto de Física)*. Antioquia, Colombia. 2012. pp. 2.

[34] Arenas Gaviria, Bernardo. *Dinámica De Un Cuerpo Rígido (Apuntes del Instituto de Física)*. Antioquia, Colombia. 2012. pp. 2.

[35] Córdova Alarcón, José Rodrigo. E*stimación y control de orientación para el nanosatéliteHumsat-México*. Distrito Federal, México, 2011. pp. 21.

[36] Córdova Alarcón, José Rodrigo. E*stimación y control de orientación para el nanosatéliteHumsat-México*. Distrito Federal, México, 2011. pp. 22.

[37] Córdova Alarcón, José Rodrigo. E*stimación y control de orientación para el nanosatélite Humsat-México*. Distrito Federal, México, 2011. pp. 25.

[38] Córdova Alarcón, José Rodrigo. E*stimación y control de orientación para el nanosatélite Humsat-México*. Distrito Federal, México, 2011. pp. 25.

[39] Hoja de referencia. *XPS IIC Bus Interface (v2.03a).* Xilinx, Inc. Junio 22, 2011.

[40] Hoja de referencia. *Fixed Interval Timer, FIT (v1.01.c).* Xilinx, Inc. Julio 25, 2012.

[41] Hoja de referencia. *LogiCORE IP XPS Interrupt Controller (v2.01a).* Xilinx, Inc. Abril 19, 2010.

[42] Hoja de referencia. *Processor Local Bus (PLB) v3.4 (v1.02a).* Xilinx, Inc. Abril 24, 2009.

[43] Hoja de referencia. *LogiCORE IP Fast SimplexLink (FSL) V20 Bus (v2.11c).* Xilinx, Inc. Abril 24, 2009.

[44] Bin Lu, Member; Xin Wu; Hernan Figueroa; Antonello Monti. *A Low-Cost Real-Time Hardware-in-the-Loop Testing Approach of Power Electronics Controls*. IEEE transactions on industrial electronics, vol. 54, no. 2, abril 2007.

[45] Hoja de referencia giróscopo. *ITG-3200 Product Specification*. InvenSense. 30 de Marzo de 2010.

[46] Hoja de referencia. *LogiCORE IP XPS Interrupt Controller (v2.01a).* Xilinx, Inc. Abril 19, 2010.

[47] Hoja de referencia. *LogiCORE IP Fixed Interval Timer (FIT) v1*.01b. Xilinx, Inc. Abril 19, 2010.

[48] Hoja de referencia. *XPS 16550 UART (v3.00a).* Xilinx, Inc. Septiembre 16, 2009.

[49] Hoja de referencia MicroBlaze. *MicroBlaze Processor Reference Guide Embedded Development Kit EDK 10.1i.* Xilinx, Inc. 2008.

[50] Córdova Alarcón, José Rodrigo. E*stimación y control de orientación para el nanosatélite Humsat-México*. Distrito Federal, México, 2011.

[51] José Gregorio Díaz, Ana María Mejías, Francisco Arteaga. Aplicación de los filtros de Kalman a sistemas de control. Revista INGENIERÍA UC, vol. 8, núm. 1, junio, 2001, pp. 1-18. Universidad de Carabobo, Venezuela.

[52] Córdova Alarcón, José Rodrigo. E*stimación y control de orientación para el nanosatélite Humsat-México*. Distrito Federal, México, 2011.

[53] Córdova Alarcón, José Rodrigo. E*stimación y control de orientación para el nanosatélite Humsat-México*. Distrito Federal, México, 2011.

[54] Domínguez Méndez, Paul. *Procesamiento de señales de sensores de navegación para satélites sobre una plataforma FPGA.* Distrito Federal, México, 2011.